On Attractor Detection and Optimal Control of Boolean Networks

Trinh Van Giang

Japan Advanced Institute of Science and Technology

Doctoral Dissertation

On Attractor Detection and Optimal Control of Boolean Networks

Trinh Van Giang

Supervisor : Professor Kunihiko Hiraishi

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
Information Science
December, 2021

Abstract

Boolean Networks (BNs) are simple but efficient mathematical formalism for modeling and analyzing complex biological systems, such as, gene regulatory networks, signal transduction networks. Beyond systems biology, BNs have widely been applied to various other areas, such as, mathematics, neural networks, social modeling, robotics, and computer science. Besides a plenty of applications, BNs are also interesting mathematical objects that have recently attracted various work in theory. Attractor detection and optimal control of BNs are two crucial but challenging issues of research on BNs. They have also become hot topics in many research communities. However, the existing theories and methods for these issues mainly focus on synchronous types of BNs and few ones focus on asynchronous types of BNs. Moreover, the existing methods are inefficient when the size of the network becomes large or the structure of the network becomes more complex. Hence, we focus in this dissertation on developing theories as well as efficient methods for attractor detection and optimal control of different types of BNs.

In theoretical aspects, we explore a number of new theoretical results that contribute to understanding the dynamics of BNs. First, we discover several relations in dynamics between different types of BNs. In addition, we also obtain several relations in dynamics between BNs and other conventional models. In particular, we demonstrate that these findings pave the potential ways to analyze different types of BNs as well as many other conventional models. Second, we discover several relations between the dynamics of a BN and its network structures. More specifically, we formally state and prove several relations between the dynamics of a BN and a feedback vertex set of its interaction graph. Notably, these relations do not depend on the updating scheme of the BN. Furthermore, we also state and prove a theorem on relations between the dynamics of an asynchronous Boolean network and a negative feedback vertex set of its interaction graph. Finally, we introduce several complexity analysis on three meaningful optimal control problems of deterministic asynchronous probabilistic Boolean networks.

In practical aspects, we develop several algorithms and methods for attractor detection and optimal control of different typical types of BNs. These algorithms and methods are mainly based on the new theoretical results obtained along with the reasonable use of formal techniques. We implement software tools for all the proposed algorithms and methods as well as conduct experiments to evaluate their performance. The experimental results on various classes of networks show that our algorithms and methods outperform the corresponding state-of-the-art ones and can handle large-scale networks. In particular, our method for finding attractors of an asynchronous Boolean network can handle very large networks with up to 1000 nodes in term of randomly generated networks and more than 300 nodes in terms of real biological networks. Notably, the principle that we propose in our algorithms and methods is general, thus enabling us to apply it to many types of BNs as well as paving potential ways to improve these algorithms and methods.

Keywords: Boolean networks, gene regulatory networks, attractor detection, optimal control, formal methods.

Acknowledgments

First and foremost, I owe my deepest gratitude to my dissertation supervisor, Prof. Kunihiko Hiraishi, for his continual encouragement and kind guidance during my doctoral research. Prof. Kunihiko Hiraishi has guided me from every aspect of research, and has been very supportive by offering me the freedom to explore while making himself available for discussions whenever needed. This dissertation would not have been completed without him being my supervisor.

I would like to express my sincere appreciation to collaboration efforts from Prof. Tatsuya Akutsu. His great collaboration had helped me to obtain several important results included in this dissertation.

I would like to thank my second supervisor, Prof. Kazuhiro Ogata, my minor research supervisor, Prof. Ryuhei Uehara, and Assoc. Prof. Nao Hirokawa for their support in my research and education life at the Japan Advanced Institute of Science and Technology (JAIST).

I wish to express my deep acknowledgements to the committee members consisting of Prof. Kunihiko Hiraishi, Prof. Tatsuya Akutsu, Assoc. Prof. Koichi Kobayashi, Prof. Mineo Kaneko, and Assoc. Prof. Daisuke Ishii, who gave valuable comments to help me improve the quality of this dissertation.

I would like to acknowledge the Japanese Government for supporting the scholarship (MEXT) for financing the study. In addition, I would like to thank the Japan Advanced Institute of Science and Technology (JAIST) for providing the financial support that enables me to present my work at international conferences.

Last, but not least, I would like to dedicate this dissertation to my family for their constant love, support, understanding, and encouragement.

Contents

List of Abbreviations and Math Notations

BN(s)	Boolean Network(s)
PBN(s)	Probabilistic Boolean Network(s)
SBN(s)	Synchronous Boolean Network(s)
ABN(s)	Asynchronous Boolean Network(s)
GABN(s)	Generalized Asynchronous Boolean Network(s)
DGABN(s)	Deterministic Generalized Asynchronous Boolean Network(s)
SPBN(s)	Synchronous Probabilistic Boolean Network(s)
DA-PBN(s)	Deterministic Asynchronous Probabilistic Boolean Network(s)
GRN(s)	Gene Regulatory Network(s)
SAT	Satisfiability
BDD(s)	Binary Decision Diagram(s)
SMT	Satisfiability Modulo Theory
STG(s)	State Transition Graph(s)
SCC(s)	Strongly Connected Component(s)
TS	Transition System
FVS(s)	Feedback Vertex Set(s)
PFVS(s)	Positive Feedback Vertex Set(s)
NFVS(s)	Negative Feedback Vertex Set(s)
PN(s)	Petri Net(s)
BMC	Bounded Model Checking
QBF	Quantified Boolean Formula
OOM	OutOfMemory

AAN(s)	Asynchronous Automata Network(s)
ROABN(s)	Random Order Asynchronous Boolean Network(s)
ESTG(s)	Extended State Transition Graph(s)
DA	Deterministic Asynchronous
DABN(s)	Deterministic Asynchronous Boolean Network(s)
BSBN(s)	Block-Sequential Boolean Network(s)
MxBN(s)	Mixed-Context Boolean Network(s)
STP	Semi-Tensor Product
PMC	Probabilistic Model Checking
SSMT	Stochastic Satisfiability Modulo Theory
POP	Polynomial Optimization Problem
MDP(s)	Markov Decision Process(es)
ILP	Integer Linear Programming
PCTL	Probabilistic Computation Tree Logic
SSAT	Stochastic Satisfiability
CNF	Conjunctive Normal Form
$\mathbb{N}$	The set of natural numbers
$\mathbb{R}$	The set of real numbers
$\mathbb{N}^+$	The set of positive natural numbers
$\mathbb{N}^+_{\leq k}$	The set of natural numbers from 1 to k
$\mathbb{B}$	The set of Boolean values
$\lvert S \rvert$	The cardinality of a set S
$A^\top$	The transpose of a matrix A
$\mathbb{N}^{m\times n}$	The set of $m \times n$ natural number matrices
$\mathbb{R}^{m\times n}$	The set of $m \times n$ real matrices
$\lceil x \rceil$	The smallest integer number that is not smaller than x
$P(\cdot)$	The probability of an event
$E[\,\cdot\,]$	The expectation of an event

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Boolean Networks (BNs) are simple but efficient mathematical formalism for modeling and analyzing complex biological systems (e.g., gene regulatory networks, signal transduction networks) [1, 2, 3]. A BN includes n nodes; each node can receive either 0 or 1, and can be associated with one Boolean function. Probabilistic Boolean Networks (PBNs) are a stochastic extension of BNs where each node can be associated with one or more Boolean functions, and each Boolean function has a probability for selection [4]. Beyond systems biology, BNs and PBNs have widely been applied to various other areas, such as, mathematics, neural networks, social modeling, robotics, and computer science (see, e.g., [5, 6, 7]). Besides a plenty of applications, BNs or PBNs are also interesting mathematical objects that have recently attracted various work in theory [6].

The updating scheme of a BN specifies the way that the nodes of this BN update their states through time evolution [5]. There are two major classes of updating schemes: *synchronous* [8] (all the nodes are updated simultaneously at each time step) and *asynchronous* [9] (not all the nodes are updated simultaneously at each time step). The asynchronous updating class is divided into several sub-classes. Three popular ones among them are *fully asynchronous* [9, 10], *generalized asynchronous* [11, 12], and *deterministic asynchronous* [9, 13]. According to these classes of updating schemes, we have four typical types of BNs: Synchronous Boolean Networks (SBNs) [8], Asynchronous Boolean Networks (ABNs) [10], Generalized Asynchronous Boolean Networks (GABNs) [14], and Deterministic Generalized Asynchronous Boolean Networks (DGABNs) [13], respectively. Similar to BNs, different updating classes can also be employed for PBNs. A Synchronous Probabilistic Boolean Network (SPBN) [4] that employs the synchronous updating class is the most popular PBN variant. When the information about the time scales of components is available, a Deterministic Asynchronous Probabilistic Boolean Network (DAPBN) [15, 16] is considered more useful. These typical types of BNs and PBNs have been widely studied as well as found various applications [6, 17, 18, 19, 20].

Attractor detection and optimal control of BNs are difficult and interesting in theory but also have a plenty of applications in many areas [21]. First, in the landscape of dynamics of a dynamical system, we can distinguish between the transient and long-run dynamics. In BNs or other qualitative models, the long-run dynamics is referred to as *attractors*. An attractor of a BN is a set of states such that the BN cannot escape from this set once entered it. In the biological context, attractors of a BN are linked to pheno-

types [22] or functional cellular states (e.g., proliferation, apoptosis, or differentiation) [23]. Thus, analysis of attractors could provide new insights into systems biology [24] (e.g., the origins of *cancers* [25, 26, 27], *SARS-CoV-2* [28, 29, 30], *HIV* [31]). Furthermore, attractors also play an important role in the development of new drugs [32, 33, 34]. Therefore, attractor detection is of great importance in analyzing biological systems modeled as BNs. In addition, attractors of BNs were also used to study various other systems, such as, multivariate systems [35], complex systems [36]. Second, optimal control of BNs is defined as the design of intervention strategies (control policies) to beneficially alter the dynamics of the considered system [16, 21]. For example, in the BN model of a Gene Regulatory Network (GRN), it is possible to control one or more genes such that the BN moves out of undesirable states (e.g., disease or cancerous states) and moves into desirable ones (e.g., healthy or normal states). Since BNs are logical dynamical and highly non-linear systems, control of BNs has become a hot topic in the control community [21, 37]. It has been found in various applications in many areas, such as, systems biology [38, 39, 40, 41], fault detection of logic circuits [42], industry [20]. Note that attractor detection also gives a starting point for many control approaches for biological systems [39, 40].

For attractor detection, many methods and tools [43, 44, 45, 46, 47, 48, 49, 50] have been proposed in the efforts to efficiently solve this problem. However, they are mainly designed for SBNs, the simplest type of BNs. Few methods and tools [43, 45, 49] have been proposed for ABNs, the more complex type of BNs but considered more suitable than SBNs in modeling biological systems [9, 51]. Whereas the SAT-based methods [44, 48] for attractor detection in SBNs are efficient and can handle very large networks, the presence of complex attractors in an ABN makes these SAT-based methods hard to extend to those for ABNs [49]. Moreover, the previous methods for attractor detection in ABNs, such as, the BDD-based methods [43], the decomposition-based methods [49], are unable to robustly handle large networks (e.g., networks with more than 100 nodes). Several efficient approximation methods [52, 53, 54] have also been proposed; however, their results may be of course not exact. In addition, there are the lacks of practical methods for other more complex types of BNs such as GABNs and DGABNs.

For optimal control, many methods and tools have been proposed in recent years. However, they are mainly designed for SBNs [55, 56], ABNs [57, 58, 59, 60, 61, 62, 63], or especially SPBNs [64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74]. There are the lacks of practical methods for more complex types of BNs and PBNs such as DGABNs and DA-PBNs, respectively. Note that DGABNs offer an interesting compromise between SBNs and ABNs, which could provide a suitable modeling formalism of various types of systems especially when the information about the time scales is known [75]; whereas, DA-PBNs are a generalization of both DGABNs and SPBNs [15]. Very few methods [76, 77] have also been proposed for optimal control of DGABNs or DA-PBNs; however, they are impractical due to they require to compute transition probability matrices of exponential size with respect to the number of nodes. Moreover, although some of the proposed methods for optimal control of SPBNs can avoid computing transition probability matrices of exponential size [68, 69, 72], they are still needed to improve due to their applicable ranges are still limited to medium problem instances [21].

Motivated by the aforementioned facts, in this dissertation, we aim to develop theories as well as efficient methods for attractor detection and optimal control of different types of BNs. A natural question is that why we need to consider different types of BNs? First, each type of BNs has its part in real life and can be suitable for modeling a specific type

of systems; the choice among them in a specific circumstance depends on the available data and application [77]. Second, relations in dynamics between different types of BNs can be exploited to efficiently analyze BNs. For example, attractors of an SBN can be used to efficiently find attractors of its ABN counterpart [43]. Finally, this consideration may provide new theoretical insights to the theory of BNs [78].

1.2 Contributions

We consider in this dissertation the attractor detection and optimal control issues of Boolean networks. We give a brief overview of the key contributions of this dissertation as follows.

In theory, we obtain a number of new theoretical results that contribute to the understanding of the dynamics of BNs. First, we discover several relations in dynamics between different types of BNs. More specifically, we formally state and prove several relations between the dynamics of a GABN and that of its SBN (or ABN) counterpart (Chapter 3). In addition, we also obtain several relations in dynamics between DGABNs and other conventional models including deterministic asynchronous models [9, 79, 80], block-sequential Boolean networks [81, 82], generalized asynchronous Boolean networks [14], and mixed-context Boolean networks [13] (Chapter 5). In particular, we demonstrate that these findings pave the potential ways to analyze different types of BNs as well as many other popular models. Second, we discover several relations between the dynamics of a BN and its network structures. More specifically, we formally state and prove several lemmas and theorems on relations between the dynamics of a BN and a feedback vertex set of its interaction graph (Chapter 4). Furthermore, we also state and prove a theorem on relations between the dynamics of an ABN and a negative feedback vertex set of its interaction graph. Finally, we discover the computational complexity of three meaningful optimal control problems of DA-PBNs (Chapter 7).

In practice, we develop several algorithms and methods for attractor detection and optimal control of different typical types of BNs. These algorithms and methods are mainly based on the new theoretical results mentioned in the previous paragraph along with the reasonable use of formal techniques (e.g., binary decision diagrams, satisfiability, satisfiability modulo theory, bounded model checking, Petri nets). First, we propose three BDD-based algorithms and one SAT-based algorithm for finding attractors of a GABN (Chapter 3). These algorithms are the first analytical and practical methods for analyzing GABNs. In particular, the experimental results on various classes of networks show that the SAT-based algorithm can handle large GABNs. As an application of the SAT-based algorithm, we propose an efficient method for approximating attractors of an ABN. Second, we propose a method that efficiently and exactly computes all the attractors of an ABN (Chapter 4). This method is then enhanced with two substantial improvements. The proposed method outperforms the state-of-the-art methods and now can handle very large networks with up to 1000 nodes in terms of randomly generated networks and more than 300 nodes in terms of real biological networks. In particular, the principle of this method can be applied to many other types of BNs and can pave potential ways to improve itself. Third, we propose one SMT-based method and two SMT-based methods for attractor detection and optimal control of DGABNs (Chapters 5 and 6), respectively. Note that there is no previous method specifically designed for DGABNs. Although

the proposed methods for DGABNs are extensions of the previous SAT-based methods for SBNs, we demonstrate that they contain substantial differences from the previous ones. Furthermore, the experimental results on randomly generated networks and artificial networks justify their efficiency. Fourth, we propose three approaches for solving the three optimal control problems of DA-PBNs (Chapter 7). One of these approaches is completely new, whereas two of these approaches are (non-trivial) extensions of the previous ones for SPBNs. These proposed approaches can handle large problem instances in terms of optimal control of DA-PBNs. Finally, we have developed software tools implementing all the proposed algorithms, methods, and approaches.

1.3 Dissertation Structure

The main content of this dissertation is composed of two parts, Part I and Part II. Part I focuses on attractor detection in BNs, and discusses computational methods for attractor detection in GABNs (Chapter 3), ABNs (Chapter 4), and DGABNs (Chapter 5). Part II focuses on optimal control of BNs, and discusses several methods for optimal control of DGABNs (Chapter 6) and DA-PBNs (Chapter 7).

In Chapter 2, we first provide preliminaries on BNs and their variants, attractors, interaction graphs, and Petri nets.

In Chapter 3, we introduce several relations in dynamics between GABNs and other types of BNs including SBNs and ABNs. Based on these relations, we propose three BDD-based algorithms for exactly computing attractors of a GABN. We then propose a near-exact algorithm for finding attractors of a GABN based on SAT-based bounded model checking. As an application of this algorithm, we propose an efficient method for approximating attractors of an ABN. This chapter is written using the content of three publications: two conference papers [83, 84] and one journal paper [85].

In Chapter 4, we introduce several relations between the dynamics of a BN and a feedback vertex set of its interaction graph. We then present the proposed method for exactly and efficiently finding attractors of an ABN as well as present its improved version. This chapter is written using the content of two publications: one journal paper [86] and one conference paper [87].

In Chapter 5, we introduce the concept of an extended state, leading to establishing the concept of an extended state transition graph to capture precisely the whole dynamics of a DGABN. We then present theoretical results on several relations in dynamics between DGABNs and other popular models as well as an SMT-based method for finding attractors of a DGABN. In Chapter 6, based on the concept of an extended state transition graph, we propose two SMT-based methods for optimal control of DGABNs under two control modes, the time-sensitive mode and the non-time-sensitive mode, respectively. Chapters 5 and 6 are written using the content of one journal paper [88].

Chapter 7 focuses on DA-PBNs, a probabilistic extension of DGABNs. We formulate three optimal control problems of DA-PBNs based on several typical aims of control. For each problem, we state and prove its hardness as well as show that it is in PSPACE. We then propose three solution approaches for solving these problems. This chapter is written using the content of one journal paper [89], which is being in preparation.

Finally, Chapter 8 summarizes the results on attractor detection and optimal control of different types of BNs and points out future research directions.

Chapter 2

Preliminaries

Let $\mathbb{N}$ and $\mathbb{R}$ denote the set of natural and real numbers, respectively. Denote by $\mathbb{N}^+$ the set $\mathbb{N}\backslash\{0\}$ and by $\mathbb{N}^+_{\leq k}$ the set $\{i \in \mathbb{N}^+ : i \leq k\}$. $\mathbb{B} := \{T \equiv 1, F \equiv 0\}$ denotes the Boolean domain. $\mathbb{R}^{m\times n}$ denotes the set of $m \times n$ real matrices. Denote by $P(\cdot)$ and $E[\cdot]$ the probability and the expectation of an event, respectively.

2.1 Boolean Networks

A Boolean Network (BN) is is a simple but efficient model that has been applied to various areas. The concept of BNs was first introduced in 1969 by Stuart Kauffman for modeling and analyzing dynamical behavior of gene regulatory networks [1]. In this section, we shall show the formal definition, dynamics, and multiple variants of Boolean networks.

Definition and Dynamics

Definition 2.1.1. *A Boolean Network (BN) is defined as a 2-tuple (V, F), where $V = \{x_1, ..., x_n\}$ $(n \geq 1)$ is the set of nodes and $F = \{f_1, ..., f_n\}$ is the set of Boolean functions. Each node x_i is identified as a Boolean variable, and is associated with a Boolean function $f_i : \mathbb{B}^{|IN(f_i)|} \to \mathbb{B}$, where $IN(f_i)$ is the set of input nodes of f_i. $x_i(t) \in \mathbb{B}$ and $x(t) = (x_1(t), ..., x_n(t))^\top$ denote the state of node x_i and the state of the BN at time t, respectively.*

Definition 2.1.1 gives the formal definition of a BN. At each time step, node x_i can update its state by

$$x_i(t + 1) = f_i(x(t)).$$

For simplicity, we use the notation $f_i(x(t))$ even if $IN(f_i) \subset V$. An updating scheme of a BN specifies the way that the nodes of the BN update their states through time evolution [5]. Following the updating scheme, the BN transits from a state to another state (possibly identical). This transition is called the *state transition*. Then, the dynamics of a BN can be represented by all possible states of the BN along with all possible state transitions from each state.

We note that, in general, a Boolean function can be formed by any combinations of any logical operators (e.g., CONJUNCTION $\wedge$, DISJUNCTION $\vee$, NEGATION $\neg$, and BI-IMPLICATION $\leftrightarrow$) on variables associated with its input nodes. Many types of BNs with special types of Boolean functions have been studied, such as, canalyzing

functions and nested canalyzing functions [90, 91], AND-OR functions [92, 93], conjunctive functions [94]. However, we here focus on *general* BNs where there is no restriction on Boolean functions.

Classification

There is an infinite number of possible updating schemes for Boolean networks [95]. However, we only consider typical updating classes that have been widely used in systems biology and many other research fields. We use the BN shown in Example 2.1.1 as a straight illustrative BN.

Example 2.1.1. *Consider a BN $\mathcal{N}$ of three nodes associated to three variables (x_1, x_2, x_3). Its Boolean functions are given by*

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

There are two major classes of updating schemes: synchronous [8] (all the nodes are updated simultaneously at each time step) and asynchronous [9] (not all the nodes are updated simultaneously at each time step). In the biological context, the asynchronous updating class, in which all genes take different time to change their expression levels, is closer to biological phenomena [10, 43, 96]. For example, very recent work [97] has explicitly backed up the necessity of asynchronous models for modeling GRNs over a realistic proof-of-concept case study. The asynchronous updating class is then divided into several sub-classes. Three popular ones among them are fully asynchronous [9, 10], generalized asynchronous [11, 12], and deterministic asynchronous [9, 13]. According to these updating classes, there are four typical types of BNs: Synchronous Boolean Networks (SBNs) [8], Asynchronous Boolean Networks (ABNs) [10], Generalized Asynchronous Boolean Networks (GABNs) [14], and Deterministic Generalized Asynchronous Boolean Networks (DGABNs) [13], respectively. Hereafter, we shall introduce each type in detail.

An SBN is the simplest BN model. At each time step, all its nodes will update their values simultaneously. Since the state transitions from a state of the SBN are time-invariant, the whole dynamics of an SBN can be captured by a State Transition Graph (STG). An STG is a directed graph in which each node corresponds to a state of the BN and each arc corresponds to a state transition between two states (possibly identical). One important property of the STG of an SBN is that each node has exactly one outgoing arc. Hence, the STG of an SBN of size n has 2^n nodes and 2^n arcs. For example, Figure 2.1a shows the STG of the SBN counterpart of the BN shown in Example 2.1.1. Herein, the SBN counterpart (also the ABN or GABN counterpart) can be seen as a more concrete object of the BN, where they share the same the structure information (the sets of nodes and Boolean functions) but the SBN is specified with an updating scheme. In addition, the STG of an SBN $\mathcal{S}$ can be seen as a Transition System (TS) with the transition formula

$$\mathcal{T}^{\mathcal{S}}(x(t), x(t+1)) := \bigwedge_{i=1}^{n} \{x_i(t+1) \leftrightarrow f_i(x(t))\}.$$

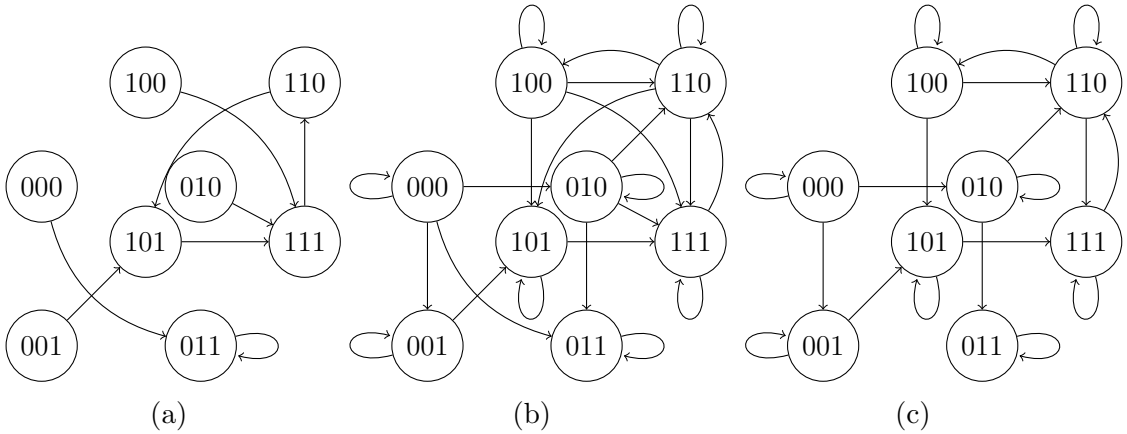


Figure 2.1: STGs of (a) the SBN counterpart, (b) the GABN counterpart, and (c) the ABN counterpart of the BN shown in Example 2.1.1.

An ABN can be seen as the most popular BN model. The updating scheme of an ABN is fully asynchronous. That is, at each time step, a single node is randomly and uniformly selected in order to be updated. By this updating scheme, a state of the ABN has n outgoing state transitions, making the dynamics of an ABN non-deterministic. Like SBNs, the whole dynamics of an ABN can be also captured by an STG. The STG of an ABN of size n has 2^n nodes and $n \times 2^n$ arcs. This property makes the analysis of an ABN more difficult. One more important property of the STG of an ABN is that two consecutive nodes (states) differ in at most one binary value. For example, Figure 2.1b shows the STG of the ABN counterpart of the BN shown in Example 2.1.1. The transition formula of an ABN $\mathcal{A}$ is

$$\mathcal{T}^{\mathcal{A}}(x(t), x(t+1)) := \bigvee_{i=1}^{n} \left\{ [x_i(t+1) \leftrightarrow f_i(x(t))] \wedge \bigwedge_{j \neq i} [x_j(t+1) \leftrightarrow x_j(t)] \right\}.$$

GABNs can be seen as a generalization of ABNs. At each time step, a GABN randomly selects any number of nodes to update synchronously. This means that the GABN can update synchronously no node, only one node, some nodes, or all the nodes. The whole dynamics of a GABN can be also captured by an STG. The STG of a GABN of size n has 2^n nodes and $2^n \times 2^n$ arcs, making its analysis more difficult. For example, Figure 2.1c shows the STG of the GABN counterpart of the BN shown in Example 2.1.1. The transition formula of a GABN $\mathcal{G}$ is

$$\mathcal{T}^{\mathcal{G}}(x(t), x(t+1)) := \bigwedge_{i=1}^{n} \{ [x_i(t+1) \leftrightarrow f_i(x(t))] \vee [x_i(t+1) \leftrightarrow x_i(t)] \}.$$

Besides the sets of nodes and Boolean functions, more information (called *context*) is added to a DGABN [13]. Then the evolution of the DGABN is specified by its context, and is time-dependent. Hence, the dynamics of the DGABN is not directly captured by an STG like SBNs, ABNs, or GABNs. Since there is no previous formal description for the dynamics of a DGABN, we shall provide detailed discussions in Section 5.2.

Useful Definitions and Notations

For convenience, we shall give several useful definitions and notations used in the rest of this dissertation. Note that some of these definitions are adjusted from [98].

Definition 2.1.2. *The successor of a state s of a BN $\mathcal{N}$ is the set of all states s' such that s has one state transition to s'.*

Definition 2.1.3. *The predecessor of a state s of a BN $\mathcal{N}$ is the set of all states s' such that s' has one state transition to s.*

Definition 2.1.4. *The forward (resp. backward) image of a set S of states of a BN $\mathcal{N}$, denoted by $FI^{\mathcal{N}}(S)$ (resp. $BI^{\mathcal{N}}(S)$), is the union of immediate successors (resp. predecessor) of the states in S.*

Definition 2.1.5. *The forward (resp. backward) reachable set of a set S of states of a BN $\mathcal{N}$, denoted by $FR^{\mathcal{N}}(S)$ (resp. $BR^{\mathcal{N}}(S)$), is the set of all the states that can be reached from (resp. can reach) at least one state in S. Specifically,*

$$FR^{\mathcal{N}}(S) = S \cup FI^{\mathcal{N}}(S) \cup FI^{\mathcal{N}}(FI^{\mathcal{N}}(S)) \ldots,$$

and

$$BR^{\mathcal{N}}(S) = S \cup BI^{\mathcal{N}}(S) \cup BI^{\mathcal{N}}(BI^{\mathcal{N}}(S)) \ldots$$

Definition 2.1.6. *The restricted backward image (resp. reachable set) of a set S of states of a BN $\mathcal{N}$ under a set S' of states, denoted by $BI^{\mathcal{N}}_{res}(S, S')$ (resp. $BR^{\mathcal{N}}_{res}(S, S')$), is defined as*

$$BI^{\mathcal{N}}_{res}(S, S') := BI^{\mathcal{N}}(S, S') \cap S'$$

(resp.

$$BR^{\mathcal{N}}_{res}(S, S') := BR^{\mathcal{N}}(S, S') \cap S'$$

).

Let us discuss how to compute these image and reachable sets of a BN. For the case that a state transition of the BN can be expressed as a propositional formula (e.g., SBNs, ABNs, GABNs), we can easily use BDDs to compute these sets. Specifically, the forward image, backward image, forward reachable set, and backward reachable set of a set of states S can be computed by using the procedures proposed in [98]. The restricted backward image and restricted backward reachable set of S under S' can also be computed with a little adjustment. The restricted backward image is computed by replacing the whole transition system of the BN by a partial transition system of S' of the BN (i.e., this partial transition system only contains state transitions starting from S'). See Section 2.2 of [98] for more details. The restricted backward reachable set is calculated by replacing forward images and backward images in Algorithm 1 of [98] by restricted forward images and restricted backward images, respectively. For the case that a state transition of the BN cannot be expressed as a propositional formula, it is difficult to directly use BDDs.

Besides the explicit method (i.e., explicitly computing the successor or the predecessor of a state), further methods may be required in this case.

Finally, we present more notations as follows. $G(\mathcal{N})$ denotes the STG of a BN $\mathcal{N}$. $FI^{\mathcal{N}}_{x_i}(S)$ denotes the forward image set of the set S by updating node x_i, i.e., the set of successors of the states in S by updating only node x_i. $\mathcal{T}^{\mathcal{N}}$ denotes the transition formula of a BN $\mathcal{N}$.

2.2 Probabilistic Boolean Networks

Various stochastic extensions of BNs have been proposed. Among them, a Probabilistic Boolean Networks (PBN) is the most extensively studied model [4]. In this section, we shall show the formal definition, dynamics, and multiple variants of Probabilistic Boolean networks.

Definition and Dynamics

Definition 2.2.1. *A Probabilistic Boolean Network (PBN) is defined as a triple* (V, F, C)*, where* $V = \{x_1, ..., x_n\}$ $(n \geq 1)$ *is the set of nodes,* $F = \{F_1, ..., F_n\}$*, and* $C = \{C_1, ..., C_n\}$*. Each node* x_i *is identified as a Boolean variable, and is associated with a non-empty set of Boolean functions,* $F_i = \{f_1^{(i)}, ..., f_{l_i}^{(i)}\}, l_i \geq 1$*. Each Boolean function* $f_j^{(i)}$ *has a probability of selection associated with it,* $c_j^{(i)}$*. Thus,* $C_i = \left\{c_1^{(i)}, ..., c_{l_i}^{(i)}\right\}$ *such that* $\sum_{j=1}^{l_i} c_j^{(i)} = 1$*. The state of a node or a PBN at time* t *is defined as same as that of a BN.*

Definition 2.2.1 gives the formal definition of a PBN. At each time step, node x_i updates its state by

$$x_i(t+1) = f_j^{(i)}(x(t)),$$

where $f_j^{(i)}$ is a Boolean function selected from F_i with the probability $c_j^{(i)}$. Similar to BNs, an updating scheme of a PBN specifies the way that the internal nodes of the PBN update their states through time evolution. Following the updating scheme, the PBN transits from a state to another state (possibly identical) with a probability. This transition is called the *probability transition*. Then, the dynamics of a PBN can be represented by all possible states of the PBN along with all possible probability transitions from each state.

Classification

There are two typical types of PBNs. The first one is Synchronous Probabilistic Boolean Networks (SPBNs) [4], where all the nodes are updated simultaneously. The second one is Deterministic Asynchronous Probabilistic Boolean Networks (DA-PBNs) [15, 16]. Roughly speaking, SPBNs and DA-PBNs are probabilistic extensions of SBNs and DGABNs, respectively. Like DGABNs, the dynamics of a DA-PBN is not well-described. Hence, we shall provide more detailed discussions in Section 7.2. Hereafter, we shall introduce SPBNs in detail.

Like SBNs, the nodes of an SPBN update their values synchronously at each time step. Then, the whole dynamics of an SPBN can be captured an STG in which a transition

probability is attached to an arc. The probability of transiting from state a to state b is

$$P\left(x(t+1)=b|x(t)=a\right) := \sum_{j_1\in\mathbb{N}^+_{\leq l_1},\ldots,j_n\in\mathbb{N}^+_{\leq l_n}} \left\{\prod_{i=1}^{n}\left[c^{(i)}_{j_i}\times P(b_i \leftrightarrow f^{(i)}_{j_i}(a))\right]\right\}.$$

See Example 2.2.1 for an SPBN and its STG. One important property of the STG of an SPBN is that the sum of the probabilities of all probability transitions from a state is equal to 1. Hence, the dynamics of an SPBN can be understood in the context of a standard Markov chain. Consequently, the techniques developed in the field of Markov chains can be applied to SPBNs.

Example 2.2.1. *Consider an SPBN $\mathcal{SP}$ [21]*

$$f^{(1)} = \begin{cases} f^{(1)}_1 = x_3, & c^{(1)}_1 = 0.8,\\ f^{(1)}_2 = \neg x_3, & c^{(1)}_2 = 0.2,\end{cases}$$
$$f^{(2)} = f^{(2)}_1 = x_1\wedge\neg x_3, \qquad c^{(2)}_1 = 1.0,$$
$$f^{(3)} = \begin{cases} f^{(3)}_1 = x_1\wedge\neg x_2, & c^{(3)}_1 = 0.7,\\ f^{(3)}_2 = x_2, & c^{(3)}_2 = 0.3.\end{cases}$$

Then, Figure 2.2 shows the STG of $\mathcal{SP}$.

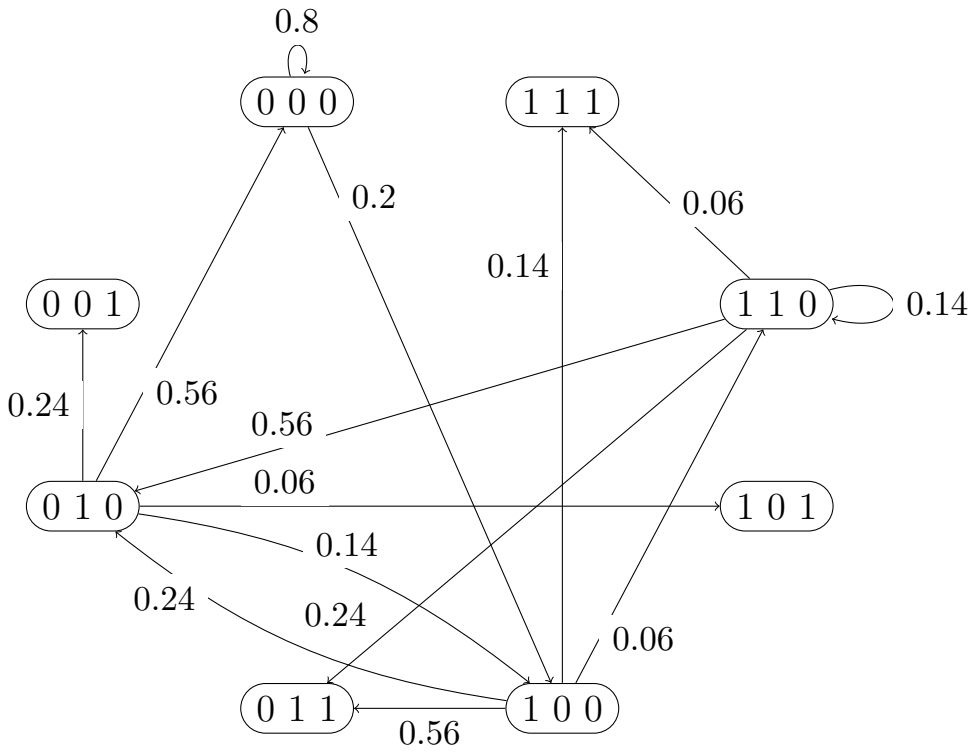


Figure 2.2: STG of the SPBN shown in Example 2.2.1. States are shown by rounded rectangles, whereas probability transitions are shown by arcs along with real numbers. For clarification, only arcs from states 000, 010, 100, and 110 are shown here.

2.3 Attractors

Attractors are key dynamical behavior of a BN. Definition 2.3.1 gives the formal definition of an attractor of a BN. This definition is general. Then, we can classify two main types of attractors: *singleton* and *cyclic* attractors. A singleton attractor (or a fixed point) has only one state. A cyclic attractor has at least two states, and is formed by overlapping one or more cycles of states. In general, an attractor of a BN is equivalent to a bottom (terminal) Strongly Connected Component (SCC) of the STG of this BN [43]. Since the STG of a BN has 2^n nodes and at least 2^n arcs, naive approaches for finding attractors (e.g, explicitly building the STG and then applying graph algorithms) are intractable when n is large.

Definition 2.3.1 ([49]). *An attractor of a BN is a set of states satisfying any state in this set can reach any state (including this state) in this set and cannot reach any state that is not in this set.*

Based on [43], we can differentiate attractors of different types of BNs. Besides singleton attractors, we classify cyclic attractors into simple attractors and complex attractors. First, a simple attractor is a cycle of at least two states such that each state has exactly one successor state (excluding itself) and may have a self transition. Simple attractors can again divided into two sub-classes: (1) type1 attractors where any two consecutive states differ in at most one single node value (or one bit in binary representation) and (2) type2 attractors where at least two consecutive states differ in more than one node value. Second, a complex attractor is formed by overlapping multiple simple attractors.

Since an SBN can have only one transition from any state, it can have three types of attractors: singleton attractors, type1 attractors, and type2 attractors. Since two consecutive states of an ABN differ in at most one node value, an ABN can have three types of attractors: singleton attractors, type1 attractors, and complex attractors. Similarly, a GABN can have three types of attractors: singleton attractor, type1 attractors, complex attractors. Note that, since GABNs allow any number of node values change between any consecutive states, they cannot have type2 attractors. If a GABN contains a type2 attractor, then it contains a state that has more than one successor state (excluding itself). This is contrary to the definition of a simple attractor.

Reconsider the BN shown in Example 2.1.1. Figures 2.1a, 2.1b, and 2.1c show the STGs of its SBN, GABN, and ABN counterparts, respectively. As we can see, all these counterparts have the same singleton attractor {011}. The SBN has a type2 attractor {101, 111, 110}. The GABN and the ABN have the same complex attractor {110, 100, 101, 111}. However, the oscillation inside this complex attractor is different between the GABN and the ABN.

2.4 Interaction Graphs

The *interaction graph* of a BN depicts the qualitative interactions between nodes and is usually represented as a signed directed graph on the set of nodes (see Definition 2.4.1). An arc from x_j to x_i indicates that the evolution of node x_i depends on the evolution of node x_j. An interaction between two nodes can be positive or negative. We first introduce some notations as follows. $IG(\mathcal{N})$ denotes the interaction graph of a BN $\mathcal{N}$. $\overline{x}^i$ denotes the state y such that $y_i = 1 - x_i$ and $y_j = x_j$ for $j \neq i$. Then, the formal definition of an

interaction graph is given in Definition 2.4.2. Note that the interaction graph can have both a positive arc and a negative arc from one vertex to another one. Furthermore, the computation of the interaction graph of a BN is often fast [78].

Definition 2.4.1 ([78]). *A signed directed graph on V is a graph G whose set of vertices is V and whose set of arcs is a subset of $V \times \{+,-\} \times V$. If (x_j, s, x_i) is an arc of G, we say that G has an arc from x_j to x_i of sign s. $s = +$ indicates a positive arc, whereas $s = -$ indicates a negative arc. A positive (resp. negative) cycle of G is an elementary directed cycle that contains an even (resp. odd) number of negative arcs. The length of a cycle is the number of arcs it involves.*

Definition 2.4.2 ([78]). *Let $\mathcal{N} = \{V, F\}$ be a BN, where $V = \{x_1, .., x_n\}$ and $F = \{f_1, ..., f_n\}$. The interaction graph of $\mathcal{N}$ is the signed directed graph on V defined by: for all $x_i, x_j \in V$, there exists a positive (resp. negative) arc from x_j to x_i if and only if there exists a state $x \in \mathbb{B}^n$ with $x_j = 0$ such that $f_i(x) < f_i(\overline{x}^j)$ (resp. $f_i(x) > f_i(\overline{x}^j)$).*

Let G be a signed directed graph. A *Feedback Vertex Set* (FVS) of G is a set of vertices U that intersects every cycle of G. In other words, G becomes acyclic after removing the vertices in U from G. A *Positive Feedback Vertex Set* (PFVS) of G is a set of vertices U^+ that intersects every positive cycle of G. A *Negative Feedback Vertex Set* (NFVS) of G is a set of vertices U^- that intersects every negative cycle of G. Equivalently, $G - U^+$ (resp. $G - U^-$) has no positive (resp. negative) cycle. By the preceding definitions, an FVS is also a PFVS or an NFVS. The problem of finding a minimum PFVS (resp. NFVS) has been proved NP-complete [99]. The problem of finding a minimum FVS has also been proved NP-complete [100].

Let us consider a BN $\mathcal{N} = \{V, F\}$, where $V = \{x_1, x_2, x_3\}$ and $F = \{f_1, f_2, f_3\}$ with

$$
\begin{aligned}
f_1 &= x_1 \wedge x_2 \wedge x_3, \\
f_2 &= x_1 \vee \neg x_3, \\
f_3 &= (x_2 \wedge \neg x_3) \vee (x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (x_1 \wedge x_2 \wedge x_3).
\end{aligned}
$$

Figure 2.3 shows the interaction graph of $\mathcal{N}$ (i.e., $IG(\mathcal{N})$). Arcs labeled with symbol "+" denote positive arcs, whereas arrows labeled with symbol "−" denote negative arcs. $IG(\mathcal{N})$ is constructed by determining arcs (positive or negative) ending at a vertex. For example, consider ending vertex x_2. Since x_1 and x_3 appear in f_2, we need to determine arcs from x_1 and x_3 to x_2. We have $f_2[x_1/0] = \neg x_3$ and $f_2[x_1/1] = 1$. If $x_3 = 1$, then $f_2[x_1/0] < f_2[x_1/1]$. In addition, there is no value of x_3 to make $f_2[x_1/0] > f_2[x_1/1]$. Hence, there is only one positive arc from x_1 to x_2. Similarly, there is only one negative arc from x_3 to x_2. Note that this procedure can be easily implemented by using BDDs.

The above interaction graph has one negative cycle of length one ($x_3 \xrightarrow{-} x_3$), one negative cycle of length two ($x_3 \xrightarrow{-} x_2 \xrightarrow{+} x_3$), one negative cycle of length three ($x_3 \xrightarrow{-} x_2 \xrightarrow{+} x_1 \xrightarrow{+} x_3$), one positive cycle of length one ($x_1 \xrightarrow{+} x_1$), two positive cycles of length two ($x_1 \xrightarrow{+} x_2 \xrightarrow{+} x_1$ and $x_1 \xrightarrow{+} x_3 \xrightarrow{+} x_1$), and one positive cycle of length three ($x_1 \xrightarrow{+} x_2 \xrightarrow{+} x_3 \xrightarrow{+} x_1$). It has two FVSs including $\{x_1, x_3\}$ (the minimum one) and $\{x_1, x_2, x_3\}$. It has four NFVSs including $\{x_3\}$ (the minimum one), $\{x_1, x_3\}$, $\{x_2, x_3\}$, and $\{x_1, x_2, x_3\}$. It also has four PFVSs including $\{x_1\}$ (the minimum one), $\{x_1, x_2\}$, $\{x_1, x_3\}$, and $\{x_1, x_2, x_3\}$.

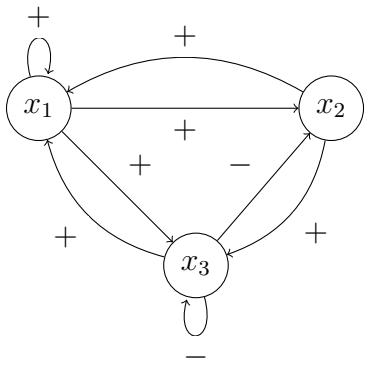


Figure 2.3: Interaction graph of the example BN.

2.5 Petri Nets and their Unfoldings

Petri nets [101] are basic modeling formalism for parallel and distributed systems. A Petri Net (PN) is a bipartite graph whose nodes are either places or transitions. In this dissertation, we only consider 1-safe Petri nets, where the number of tokens of each place is either 0 (unmarked) or 1 (marked). The set of marked places forms a marking of the PN. If the PN has an arc (p, t), then p is called an input place of t. If the PN has an arc (t, p), then p is called an output place of t. A transition is enabled if all its input places are marked. When a transition is enabled, it can fire. The firing of this transition makes all its input places unmarked and then makes all its output places marked, modifying the current marking of the PN. Note that when multiple transitions are enabled, only one transition can fire.

Formally, a 1-safe PN is a tuple $\mathcal{P} = \langle P, T, pre, post, M_0 \rangle$. P and T are sets of places and transitions, respectively. $pre \subseteq P \times T$ is the set of all arcs from places to transitions, whereas $post \subseteq T \times P$ is the set of all arcs from transitions to places. For any place p, we say *pre-set* of p is the set ${}^\bullet p = \{t \in T | (t, p) \in post\}$ and *post-set* of p is the set $p^\bullet = \{t \in T | (p, t) \in pre\}$. For any transition t, we say *pre-set* of t is the set ${}^\bullet t = \{p \in P | (p, t) \in pre\}$ and *post-set* of t is the set $t^\bullet = \{p \in P | (t, p) \in post\}$. A subset $M \subseteq P$ of the places is called a marking. M_0 is the initial marking of the PN.

A transition t of a 1-safe PN is enabled at a marking M if and only if ${}^\bullet t \subseteq M$. The firing of t leads to a new marking M' specified by $M' = (M \backslash {}^\bullet t) \cup t^\bullet$. We denote this marking transition by $M \xrightarrow{t} M'$. A marking M' is called reachable from a marking M if there exists a firing sequence $w = t_1 t_2 ...$ over T such that $M \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 ... \rightarrow M'$. A marking reachable from M_0 is called a *reachable marking* of the PN. All reachable markings of the PN and their marking transitions are represented by a directed graph called the *reachability graph*.

Figure 2.4a shows an example 1-safe PN. The places are represented by circles and the transitions are represented by squares. The arcs and the initial marking are represented by the arrows and the dots in the marked places, respectively. Herein, $M_0 = \{p_1\}$. At this marking, t_1 and t_2 are enabled. Figure 2.4b shows the reachability graph of this PN.

Petri net unfoldings [102] aim at representing the reachability graph of a 1-safe PN by exploiting concurrency between transitions to prune redundant interleavings of these transitions. The unfolding of a 1-safe PN $\mathcal{P}$ can be seen as an acyclic PN $\mathcal{U}$ that has the same behavior as $\mathcal{P}$. In general, $\mathcal{U}$ is infinite. However, there exists a finite prefix $\mathcal{P}_\mathcal{U}$ of $\mathcal{U}$

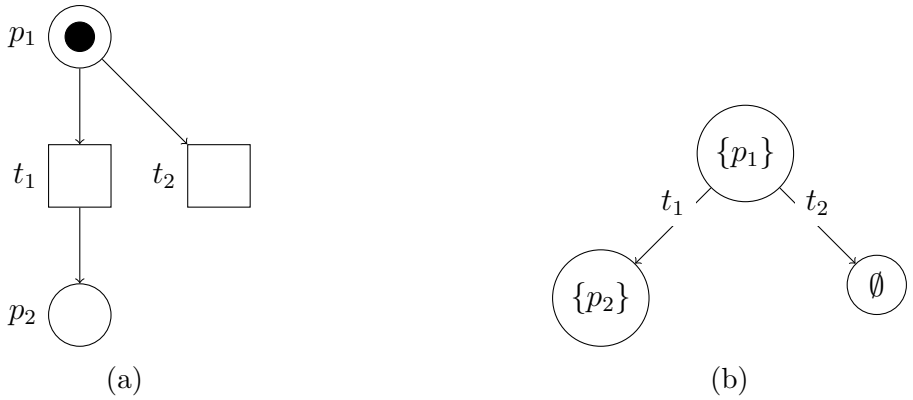


Figure 2.4: (a) A 1-safe PN and (b) its reachability graph with the initial marking $\{p_1\}$. In (b), the text above each arrow denotes the fired transition.

because $\mathcal{P}$ is safe. Generally speaking, a finite prefix is an acyclic PN whose sets of places, transitions, and arcs are finite and are subsets of the sets of places, transitions, and arcs of $\mathcal{U}$, respectively. $\mathcal{P}_\mathcal{U}$ is complete because every reachable marking of $\mathcal{P}$ has a reachable counterpart in $\mathcal{P}_\mathcal{U}$. Thus, $\mathcal{P}_\mathcal{U}$ represents the set of reachable markings of $\mathcal{P}$. See [103] for more details on finite complex prefixes. Regarding the reachability problem of 1-safe Petri nets, we can build $\mathcal{P}_\mathcal{U}$ and then easily check whether a marking M reaches a marking M'. This function is implemented in **Mole** [104], which is an efficient tool for checking reachability property of 1-safe Petri nets based on unfoldings. **Mole** also supports a function allowing that building an unfolding immediately stops when a specific transition t can be added to this unfolding.

Recently, some work has considered the link between Petri nets and Boolean networks. The authors of [105] show that ABNs have high concurrency that is the essential property of PNs. The method for encoding an ABN as a 1-safe PN has been proposed [103]. Thus, we can apply the algorithms [106] for checking the reachability in 1-safe PNs to the reachability analysis on ABNs. Conversely, the reachability property of a 1-safe PN can also be described by that of an ABN [105].

Part I

Attractor Detection

Chapter 3

Attractor Detection in Generalized Asynchronous Boolean Networks (GABNs)

3.1 Introduction

There are many theoretical studies on attractors of different types of BNs, such as, [22, 107] for attractors in SBNs, [9, 14] for attractors in ABNs, [9, 11, 14] for attractors in GABNs and DGABNs. In practice, many methods have been proposed for attractor detection (i.e., finding all possible attractors) in different types of BNs, such as, [44, 98, 108] for attractor detection in SBNs, [43, 103, 108] for attractor detection in ABNs. As mentioned in Chapter 1, asynchronous Boolean networks are considered more suitable than synchronous ones in modeling biological systems. In addition, ABNs and GABNs are conventional asynchronous models that are usually used in systems biology by the fact that precise information on time scales of components is usually missing [9]. However, to our best knowledge, there is no efficient method specifically designed for attractor detection in GABNs. This fact motivates research on attractor detection in GABNs.

In this chapter, we focus on attractors of GABNs. GABNs are more general than SBNs and ABNs because they can pick randomly any number of nodes to update synchronously at each time step. They are also interesting mathematical objects themselves. We first recall two dynamical properties of a GABN (Section 3.2). Then, we formally state and prove several relations in dynamics between a GABN and its SBN counterpart (Section 3.3). Based on these properties and relations, we propose three possible algorithms for attractor detection in GABNs (Section 3.4). All these algorithms use Binary Decision Diagrams (BDDs) [109] to represent components of the network (such as, Boolean functions, sets of states, state transitions) and manipulate operations on these components. We also formally prove the correctness of these algorithms. Furthermore, experiments are also conducted on real and artificial networks to compare the performance of the proposed algorithms.

The experimental results show that the three BDD-based algorithms still meet the inherent problems of BDDs (e.g., extremely long computational time, high memory consumption) when the network size is large (e.g., $n > 30$). Hence, we propose a new algorithm, which is based on SAT-based Bounded Model Checking (BMC) [110], to overcome these inherent problems (Section 3.5). The new algorithm is maybe not exact in

some cases because we use a bound, which is good enough but maybe not a completeness threshold [110]. However, the experimental results on real biological and artificial networks confirm the effectiveness (accuracy and efficiency) of the new algorithm.

Finally, to highlight the applications of the study on GABNs, we state and prove several relations in dynamics between a GABN and its ABN counterpart (Subsection 3.6.1). Based on these relations, we propose an efficient method for approximating attractors of an ABN (Subsection 3.6.2). The experimental results on real biological networks are promising because the approximation method outperforms the two state-of-the-art methods as well as can handle networks of size up to 101 (Subsection 3.6.3). These results also justify the accuracy of the approximation method and consolidate the observation that the number of attractors of an ABN is equal to that of its GABN counterpart in most cases. This observation is useful in studying the dynamics of Boolean networks.

3.2 Dynamical Properties

Theorem 3.2.1. *In a BN $\mathcal{N}$, state s is an attractor state if and only if $FR^{\mathcal{N}}(\{s\}) \subseteq BR^{\mathcal{N}}(\{s\})$. State s is a transient state otherwise.*

Lemma 3.2.1. *In a BN $\mathcal{N}$, if state s is transient, then states in $BR^{\mathcal{N}}(\{s\})$ are also all transient. If state s is an attractor state, then all states in $FR^{\mathcal{N}}(\{s\})$ are also attractor states. In the latter case, $FR^{\mathcal{N}}(\{s\})$ is an attractor and $BR^{\mathcal{N}}(\{s\}) - FR^{\mathcal{N}}(\{s\})$ are all transient states.*

Note that Theorem 3.2.1 and Lemma 3.2.1 are valid for all types of BNs. Their proof can be found in [111].

3.3 Relations in Dynamics between GABNs and Synchronous Boolean Networks

The following lemmas and theorems represent the relations in dynamics between a GABN and its SBN counterpart.

Lemma 3.3.1. *Let $\mathcal{G}$ be a GABN and $\mathcal{S}$ be its SBN counterpart. If s is a state of $\mathcal{S}$, then $FR^{\mathcal{S}}(\{s\}) \subseteq FR^{\mathcal{G}}(\{s\})$.*

Proof. Since s (in $\mathcal{G}$) has one state transition corresponding to the case that all nodes are updated, $FI^{\mathcal{S}}(\{s\}) \subseteq FI^{\mathcal{G}}(\{s\})$. By the definition of a forward reachable set, we obtain $FR^{\mathcal{S}}(\{s\}) \subseteq FR^{\mathcal{G}}(\{s\})$. □

Lemma 3.3.2. *Let $\mathcal{G}$ be a GABN and $\mathcal{S}$ be its SBN counterpart. $\mathcal{G}$ and $\mathcal{S}$ have the same set of singleton attractors.*

Proof. This lemma immediately holds by the finding of Gershenson [14], which is that singleton attractors are the same for any type of BNs. □

Lemma 3.3.3. *Let $\mathcal{G}$ be a GABN and and $\mathcal{S}$ be its SBN counterpart. $\mathcal{G}$ and $\mathcal{S}$ have the same set of type1 attractors.*

Proof. Let att^1 be a type1 attractor of $\mathcal{S}$. Let s be an arbitrary state in att^1 and s' be its successor state. Since s and s' differ in exactly one bit (say the ith bit), the updating of all the nodes in $\mathcal{S}$ changes only the ith bit of s. Thus, the updating of some nodes in $\mathcal{G}$ changes the ith bit (node x_i is in the set of updated nodes) or no bit (node x_i is not in the set of updated nodes) of s. This implies that $FI^{\mathcal{G}}(\{s\}) = \{s, s'\}$. Since s is arbitrary, att^1 is also a type1 attractor of $\mathcal{G}$ (*).

Let att^2 be a type1 attractor of $\mathcal{G}$. Let s be an arbitrary state in att^2 and s' be a successor state (excluding s) of s. Then, $FI^{\mathcal{G}}(\{s\}) = \{s, s'\}$. We have $FI^{\mathcal{S}}(\{s\}) \subseteq FI^{\mathcal{G}}(\{s\})$. $FI^{\mathcal{S}}(\{s\}) = \{s'\}$ because if $FI^{\mathcal{S}}(\{s\}) = \{s\}$ then $FI^{\mathcal{G}}(\{s\}) = \{s\}$. Since s is arbitrary, att^2 is also a type1 attractor of $\mathcal{S}$ (**).

From (*) and (**), we can conclude the proof. □

Theorem 3.3.1. *Let $\mathcal{G}$ be a GABN and $\mathcal{S}$ be its SBN counterpart. Any attractor of $\mathcal{G}$ always contains an attractor of $\mathcal{S}$.*

Proof. Assume that there is an attractor att of $\mathcal{G}$ that does not contain any attractor of $\mathcal{S}$ (*).

Let s be an arbitrary state in att. By Lemma 3.3.1, we have $FR^{\mathcal{S}}(\{s\}) \subseteq FR^{\mathcal{G}}(\{s\})$. Since att is an attractor of $\mathcal{G}$, $att = FR^{\mathcal{G}}(\{s\})$. Hence, $FR^{\mathcal{S}}(\{s\}) \subseteq att$.

Clearly, there is an attractor att' of $\mathcal{S}$ such that $att' \subseteq FR^{\mathcal{S}}(\{s\})$. Then $att' \subseteq att$ that is contrary to (*). □

3.4 BDD-Based Algorithms

From the properties and relations presented in Sections 3.2 and 3.3, we propose three algorithms (called **FR-BR-BDD-1**, **FR-BR-BDD-2**, and **filtBDD**, respectively) to detect all reachable attractors of a GABN. All these algorithms use BDDs to represent components of the network (such as, Boolean functions, sets of states, state transitions) and manipulate operations on these components. Before presenting the three proposed algorithms, we discuss the way of encoding a GABN using BDDs.

The STG of a GABN can be easily modeled as a TS. The authors of [98] have proposed a method to encode the STG of an ABN as a TS. This encoding can be applied similarly for GABNs. We can only replace the state transition formula of the ABN (see [98]) by the state transition formula of the GABN $\mathcal{G}$ that is expressed by

$$\mathcal{T}^{\mathcal{G}}(x, x') = \bigwedge_{i=1}^{n} \{(x'_i \leftrightarrow x_i) \vee (x'_i \leftrightarrow f_i(x))\},$$

where $x = (x_1, ..., x_n)^\top$ denotes the current state, $x' = (x'_1, ..., x'_n)^\top$ denotes the next state.

Note that $\mathcal{T}^{\mathcal{G}}(x, x')$ is a propositional formula of $2n$ Boolean variables. It can be seen as a Boolean function $f : \mathbb{B}^{2n} \to \mathbb{B}$; thus, it can be easily encoded into a BDD with the set of $2n$ BDD variables $V = \{x_1, ..., x_n, x'_1, ..., x'_n\}$. Now, the forward (reps. backward) image of a set of states of the GABN can be computed by using this BDD. Consequently, the forward (resp. backward) reachable set of a set of states of the GABN can be computed by using this BDD. The technical details are similar to those presented in [98].

3.4.1 Algorithm FR-BR-BDD-1

The idea of **FR-BR-BDD-1** is inspired by the **genYsis** tool [98] that builds the total transition system (for all states of the BN) and then calculates backward reachable sets. Algorithm 1 shows the description of **FR-BR-BDD-1**. Herein, the variables att, FS, and BS are BDDs, whereas the variables $A^{\mathcal{S}}_{type2}$ and $A^{\mathcal{G}}$ are lists of BDDs. The operator '+' is the set operator that adds a new element to a set. The operator '−' is the set operator that removes expansively elements from a set. For example, $\{\{1,2\},\{3,4\}\} - \{1,2,3,4\} = \emptyset$. Algorithm 1 will terminate when $A^{\mathcal{S}}_{type2}$ becomes empty.

Consider a running example. All running steps of **FR-BR-BDD-1** for the GABN counterpart of the BN shown in Example 2.1.1 are as follows. First, we obtain $A^{\mathcal{S}}_{type2} = \{\{110, 101, 111\}\}$ and $A^{\mathcal{G}} = \{\{011\}\}$. Next, we have $att = \{110, 101, 111\}$, $FS = \{110, 100, 111, 101\}$, and $BS = \{110, 100, 111, 010, 101, 001, 000\}$. Since $FS \subseteq BS$, we have FS is a new attractor and is added to $A^{\mathcal{G}}$. Now, $A^{\mathcal{S}}_{type2}$ becomes empty, and **FR-BR-BDD-1** terminates.

Algorithm 1 FR-BR-BDD-1

Input: A GABN $\mathcal{G}$.
Output: The set of attractors of $\mathcal{G}$.

Let $\mathcal{S}$ be the SBN counterpart of $\mathcal{G}$
Compute all attractors of $\mathcal{S}$
$A^{\mathcal{S}}_{sing} \leftarrow$ the set of singleton attractors of $\mathcal{S}$
$A^{\mathcal{S}}_{type1} \leftarrow$ the set of type1 attractors of $\mathcal{S}$
$A^{\mathcal{S}}_{type2} \leftarrow$ the set of type2 attractors of $\mathcal{S}$
$A^{\mathcal{G}} \leftarrow A^{\mathcal{S}}_{sing} \cup A^{\mathcal{S}}_{type1}$
while $A^{\mathcal{S}}_{type2} \neq \emptyset$ **do**
    Randomly remove a type2 attractor att from $A^{\mathcal{S}}_{type2}$
    $FS \leftarrow FR^{\mathcal{G}}(att)$
    $BS \leftarrow BR^{\mathcal{G}}(att)$
    if $FS \subseteq BS$ **then**
        $A^{\mathcal{G}} \leftarrow A^{\mathcal{G}} + FS$ {a new attractor}
        $A^{\mathcal{S}}_{type2} \leftarrow A^{\mathcal{S}}_{type2} - FS$
    end if
end while
return $A^{\mathcal{G}}$

Finally, we formally prove the correctness of **FR-BR-BDD-1**.

Theorem 3.4.1. *The result of **FR-BR-BDD-1** is correct, i.e., it finds correctly all attractors of a GABN $\mathcal{G}$.*

Proof. First, the set of attractors $A^{\mathcal{G}}$ is initialized the set of singleton and type1 attractors of the SBN $\mathcal{S}$. Based on Lemma 3.3.2 and Lemma 3.3.3, we have that the result of **FR-BR-BDD-1** contains *exactly* all singleton and type1 attractors of the GABN $\mathcal{G}$ (*).

Second, let s be an arbitrary state in att. By the updating scheme of a GABN, the STG of $\mathcal{G}$ contains all state transitions of the STG of its SBN counterpart (say $\mathcal{S}$). As a consequence, $att = FR^{\mathcal{S}}(\{s\}) \subseteq FR^{\mathcal{G}}(\{s\})$ and $att \subseteq BR^{\mathcal{S}}(\{s\}) \subseteq BR^{\mathcal{G}}(\{s\})$. By the definition of a forward reachable set, $FR^{\mathcal{G}}(att) = FR^{\mathcal{G}}(\{s\})$. By the definition of

a backward reachable set, $BR^{\mathcal{G}}(att) = BR^{\mathcal{G}}(\{s\})$. Then the condition $FS \subseteq BS$ is equivalent to the condition $FR^{\mathcal{G}}(\{s\}) \subseteq BR^{\mathcal{G}}(\{s\})$. Hence, following Theorem 3.2.1, a new complex attractor found **FR-BR-BDD-1** is an actual complex attractor of $\mathcal{G}$.

By Theorem 3.3.1, $A^{\mathcal{G}}$ (the result of **FR-BR-BDD-1**) will contain all complex attractors of $\mathcal{G}$. Since the found attractor is excluded from $A^{\mathcal{S}}_{type2}$ (Line 13 of Algorithm 1), $A^{\mathcal{G}}$ contains no duplicate complex attractors. Hence, the result of **FR-BR-BDD-1** contains *exactly* all complex attractors of the GABN $\mathcal{G}$ (**).

From (*) and (**), we can conclude the proof. □

3.4.2 Algorithm FR-BR-BDD-2

In case of too large backward reachable sets, the running time of **FR-BR-BDD-1** may become extremely longer, even it fails to obtain the result due to the OutOfMemory (OOM) error. With the special properties of attractors of GANs, the calculation of (total) backward reachable sets is unnecessary. We here propose a new algorithm called **FR-BR-BDD-2** that improves **FR-BR-BDD-1**.

Algorithm 2 shows the description of **FR-BR-BDD-2**. There are two main differences between **FR-BR-BDD-2** and **FR-BR-BDD-1**. First, **FR-BR-BDD-2** does not build the total transition system of the GABN. It only builds partial transition systems. Since the number of state transitions of a GABN is $\mathcal{O}(2^{2n})$, the calculation of the total transition system may lead to a context of too large BDDs, thus leading to OOM. Therefore, the range of applicable networks of **FR-BR-BDD-2** may be larger than that of **FR-BR-BDD-1**. Second, **FR-BR-BDD-2** calculates restricted backward reachable sets instead of (total) backward reachable sets. $BS \leftarrow BR^{\mathcal{G}}(att)$ in Algorithm 1 is replaced by $BS_{res} \leftarrow BR^{\mathcal{G}}_{res}(att, FS)$ (see Line 10 of Algorithm 2). When FS is not an attractor, BS may be significantly larger than both FS and BS_{res}. In this case, **FR-BR-BDD-2** may be much faster than **FR-BR-BDD-1**.

The running steps of **FR-BR-BDD-2** for the GABN counterpart of the BN shown in Example 2.1.1 are similar to those of **FR-BR-BDD-1**. However, BS_{res} $(\{110, 100, 111, 101\})$ is smaller than BS $(\{110, 100, 111, 010, 101, 001, 000\})$.

Finally, we formally prove the correctness of **FR-BR-BDD-2**.

Theorem 3.4.2. *The result of **FR-BR-BDD-2** is correct, i.e., it finds correctly all attractors of a GABN $\mathcal{G}$.*

Proof. Since **FR-BR-BDD-2** only differs **FR-BR-BDD-1** at the condition $BS_{res} \leftarrow BR^{\mathcal{G}}_{res}(att, FS)$ (Line 10 of Algorithm 2), we only need to show that a new complex attractor found in **FR-BR-BDD-2** is an actual complex attractor of $\mathcal{G}$. Let s be an arbitrary state in att.

Assume that FS is a complex attractor of $\mathcal{G}$. Then s is an attractor state of $\mathcal{G}$. We have $FS = FR^{\mathcal{G}}(\{s\}) = FR^{\mathcal{G}}(att)$ and $BR^{\mathcal{G}}(att) = BR^{\mathcal{G}}(\{s\})$. By Theorem 3.2.1, $FR^{\mathcal{G}}(\{s\}) \subseteq BR^{\mathcal{G}}(\{s\})$. Thus, $FS = FR^{\mathcal{G}}(att) \subseteq BR^{\mathcal{G}}(att)$. Then $BR^{\mathcal{G}}_{res}(att, FS) = BR^{\mathcal{G}}(att) \cap FS = FS$. Hence, $BS_{res} = FS$ holds.

Assume that $BS_{res} = FS$ holds. We have $FS = FR^{\mathcal{G}}(att) = FR^{\mathcal{G}}(\{s\})$ and $BR^{\mathcal{G}}(att) = BR^{\mathcal{G}}(\{s\})$. Since $BR^{\mathcal{G}}_{res}(att, FS) = BR^{\mathcal{G}}(att) \cap FS = BR^{\mathcal{G}}(\{s\}) \cap FS$, $FS = BR^{\mathcal{G}}(\{s\}) \cap FS$. Then $FR^{\mathcal{G}}(\{s\}) = FS \subseteq BR^{\mathcal{G}}(\{s\})$. By Theorem 3.2.1, s is an attractor state of $\mathcal{G}$. Hence, FS is a complex attractor $\mathcal{G}$.

Algorithm 2 FR-BR-BDD-2

Input: A GABN $\mathcal{G}$.
Output: The set of attractors of $\mathcal{G}$.

Let $\mathcal{S}$ be the SBN counterpart of $\mathcal{G}$
Compute all attractors of $\mathcal{S}$
$A^{\mathcal{S}}_{sing} \leftarrow$ the set of singleton attractors of $\mathcal{S}$
$A^{\mathcal{S}}_{type1} \leftarrow$ the set of type1 attractors of $\mathcal{S}$
$A^{\mathcal{S}}_{type2} \leftarrow$ the set of type2 attractors of $\mathcal{S}$
$A^{\mathcal{G}} \leftarrow A^{\mathcal{S}}_{sing} \cup A^{\mathcal{S}}_{type1}$
while $A^{\mathcal{S}}_{type2} \neq \emptyset$ **do**
  Randomly remove a type2 attractor att from $A^{\mathcal{S}}_{type2}$
  $FS \leftarrow FR^{\mathcal{G}}(att)$
  $BS_{res} \leftarrow BR^{\mathcal{G}}_{res}(att, FS)$
  if $BS_{res} = FS$ **then**
    $A^{\mathcal{G}} \leftarrow A^{\mathcal{G}} + FS$ {a new attractor}
    $A^{\mathcal{S}}_{type2} \leftarrow A^{\mathcal{S}}_{type2} - FS$
  end if
end while
return $A^{\mathcal{G}}$

Now, we can conclude that the condition $BS_{res} = FS$ is equivalent to whether FS is a complex attractor of $\mathcal{G}$. Hence, a new complex attractor found in **FR-BR-BDD-2** is an actual complex attractor of $\mathcal{G}$. □

3.4.3 Algorithm filtBDD

Both **FR-BR-BDD-1** and **FR-BR-BDD-2** rely on the calculation of backward reachable sets (total or restricted). Hence, when the backward reachable sets become too large, the algorithms may take extremely long computational even may fail to obtain the result due to OOM. We here propose a new algorithm called **filtBDD** to overcome this problem.

The intuitive idea of **filtBDD** is as follows. By Theorem 3.3.1, for any $att^{\mathcal{G}} \in A^{\mathcal{G}}$, there is an $att^{\mathcal{S}} \in A^{\mathcal{S}}$ such that $att^{\mathcal{S}} \subseteq att^{\mathcal{G}}$. We then filter out the set $A^{\mathcal{S}}$ to get the new set $A^{\mathcal{S}}_{new}$ that one-to-one corresponds to $A^{\mathcal{G}}$. The filtering process is: if $att^{\mathcal{S}}$ reaches other elements of $A^{\mathcal{S}}$ in $G(\mathcal{G})$, then $att^{\mathcal{S}}$ is filtered out $A^{\mathcal{S}}$. The reachability property is checked by calculating the forward reachable set $FR^{\mathcal{G}}(att^{\mathcal{S}})$. Now, for each $att \in A^{\mathcal{S}}_{new}$, $FR^{\mathcal{G}}(att)$ is an attractor of $\mathcal{G}$. Note that $FR^{\mathcal{G}}(att)$ has already calculated in the filtering process. Furthermore, the reachability property can be on-the-fly checked. That is, in each iteration of the algorithm for calculating $FR^{\mathcal{G}}(att)$ (see [98]), we check whether the current frontier set intersects $att^{\mathcal{S}}$. If yes, we can immediately stop the reachability checking and return reachable.

Algorithm 3 shows the description of **filtBDD**. Note that, by Lemmas 3.3.2 and 3.3.3, the filtering process can start with $A^{\mathcal{S}}_{type2}$ instead of $A^{\mathcal{S}}$. Let us see a running example of **filtBDD** for the GABN counterpart of the BN shown in Example 2.1.1. First, we obtain the set of attractors $A^{\mathcal{G}} = \{\{011\}\}$, the filtering set $A^{\mathcal{S}}_{type2} = \{\{101, 111, 110\}\}$. Next, we have $att^{\mathcal{S}} = \{101, 111, 110\}$ and $A^{\mathcal{S}}_{type2} = \emptyset$. Then, $FR^{\mathcal{G}}(att^{\mathcal{S}}) = \{110, 100, 111, 101\}$. Since this set does not contain any attractor in $A^{\mathcal{S}}_{type2} \cup A^{\mathcal{G}}$, it is a new complex attractor.

filtBDD terminates because $A^{\mathcal{S}}_{type2}$ is empty.

Algorithm 3 filtBDD

Input: A GABN $\mathcal{G}$.
Output: The set of attractors of $\mathcal{G}$.
1: Let $\mathcal{S}$ be the SBN counterpart of $\mathcal{G}$
2: Compute all attractors of $\mathcal{S}$
3: $A^{\mathcal{S}}_{sing} \leftarrow$ the set of singleton attractors of $\mathcal{S}$
4: $A^{\mathcal{S}}_{type1} \leftarrow$ the set of type1 attractors of $\mathcal{S}$
5: $A^{\mathcal{S}}_{type2} \leftarrow$ the set of type2 attractors of $\mathcal{S}$
6: $A^{\mathcal{G}} \leftarrow A^{\mathcal{S}}_{sing} \cup A^{\mathcal{S}}_{type1}$
7: **while** $A^{\mathcal{S}}_{type2} \neq \emptyset$ **do**
8: Randomly remove a type2 attractor $att^{\mathcal{S}}$ from $A^{\mathcal{S}}_{type2}$
9: **if** $att^{\mathcal{S}}$ does not reach in $G(\mathcal{G})$ any attractor in $A^{\mathcal{S}}_{type2} \cup A^{\mathcal{G}}$ **then**
10: $A^{\mathcal{G}} \leftarrow A^{\mathcal{G}} + FR^{\mathcal{G}}(att^{\mathcal{S}})$ {a new attractor}
11: **end if**
12: **end while**
13: **return** $A^{\mathcal{G}}$

Finally, we formally prove the correctness of **filtBDD**.

Theorem 3.4.3. *The result of **filtBDD** is correct, i.e., it finds correctly all attractors of a GABN $\mathcal{G}$.*

Proof. First, the set of attractors $A^{\mathcal{G}}$ is initialized the set of singleton and type1 attractors of the SBN $\mathcal{S}$. Based on Lemmas 3.3.2 and 3.3.3, we have that the result of **filtBDD** contains exactly all singleton and type1 attractors of the GABN $\mathcal{G}$ (*).

Second, in the case that $att^{\mathcal{S}}$ reaches in $G(\mathcal{G})$ an attractor in $A^{\mathcal{S}}_{type2} \cup A^{\mathcal{G}}$ (say att), $FR^{\mathcal{G}}(att^{\mathcal{S}})$ is not added to $A^{\mathcal{G}}$. However, by the definition of an attractor, if $att^{\mathcal{S}}$ belongs to an attractor of $\mathcal{G}$, then att also belongs to this attractor. By Theorem 3.3.1, when $A^{\mathcal{S}}_{type2}$ becomes empty, $A^{\mathcal{G}}$ contains all attractors of $\mathcal{G}$ (**).

In addition, let FS be a new attractor found in **filtBDD** (i.e., $FR^{\mathcal{G}}(att^{\mathcal{S}})$ in Line 10 of Algorithm 3). FS is a new attractor because FS is not already in $A^{\mathcal{G}}$. Otherwise, $FS \in A^{\mathcal{S}}_{type2} \cup A^{\mathcal{G}}$ that is contrary to the condition in Line 9 of Algorithm 3. Assume that FS is not an actual new attractor of $\mathcal{G}$. Then, FS must contain an attractor of $\mathcal{G}$. Since $A^{\mathcal{S}}_{type2} \cup A^{\mathcal{G}}$ always keeps the found attractors and remains possible attractors by Theorem 3.3.1, we have FS must contain an element of $A^{\mathcal{S}}_{type2} \cup A^{\mathcal{G}}$. That is contrary to the condition in Line 9 of Algorithm 3. Hence, FS is an actual new attractor of $\mathcal{G}$ (***).

From (*), (**), and (***), we can conclude the proof. □

3.4.4 Evaluation

We have implemented the three proposed algorithms. The implementation is in JAVA language and uses JDD library [112] for BDD manipulation. We then conducted experiments to compare the performance of these algorithms. Since the correctness of these algorithms has been proved, the evaluation metric is here computational time.

All the experiments were conducted in a computer whose environment is CPU: Intel Core i7 2.4 GHz, Memory: 16 GB, Windows 10 Home 64 bit. We used two sets of models.

The real biological networks that were obtained from the literature include the budding yeast cell cycle regulation [113], the mammalian cell cycle regulation [114], the fission yeast cell cycle regulation [113], the T-helper cell differentiation [115], and the T-cell receptor signaling pathway [116]. The artificial networks are randomly generated with **Bool Net R** package [117].

Table 3.1 shows experimental results on real biological networks. "-" denotes the case in which the algorithm failed to detect attractors due to OOM. Columns 1, 2, and 3 denote the name of the network, the number of nodes of the network, and the number and size of attractors computed by the considered algorithms, respectively. In columns 7, 8, and 9, we show the computational time (in seconds) of the three algorithms, respectively. There are two remarks obtained from this table. First, **FR-BR-BDD-2** is always better than **FR-BR-BDD-1** in four per five networks (except the T-cell receptor network in which both **FR-BR-BDD-1** and **FR-BR-BDD-2** failed to obtain the result). Second, **filtBDD** is insignificantly better than **FR-BR-BDD-1** and **FR-BR-BDD-2** in the four networks. The reason may be due to small numbers of nodes or small attractors. However, **filtBDD** outperforms both **FR-BR-BDD-1** and **FR-BR-BDD-2** in the T-cell receptor network (a network with a larger number of nodes). Specifically, **FR-BR-BDD-1** and **FR-BR-BDD-2** failed to obtain the result, whereas **filtBDD** succeeded in only 0.171 seconds.

Table 3.1: Experimental results of **FR-BR-BDD-1**, **FR-BR-BDD-2**, and **filtBDD** on real biological networks.

Name	n	Number $\times$ size of attractors	**FR-BR-BDD-1** (seconds)	**FR-BR-BDD-2** (seconds)	**filtBDD** (seconds)
Mammalian cell	10	1×1, 1×128	0.130	0.121	0.119
Fission yeast	10	13×1	0.122	0.119	0.129
Budding yeast	12	7×1	0.153	0.128	0.127
T-helper cell	23	3×1	214.361	0.190	0.197
T-cell receptor	40	8×1	-	-	0.171

Table 3.2 shows experimental results on artificial networks. Column 1 stands for the name of the network. Column 2 stands for the number and size of attractors computed by the considered algorithms. In columns 7, 8, and 9, we show the computational time (in seconds) of the three algorithms, respectively. From these results, we obtain three remarks as follows. First, **filtBDD** outperforms both **FR-BR-BDD-1** and **FR-BR-BDD-2** in all succeeded cases (i.e., these cases in which at least one algorithm succeeded to obtain the result). Second, there exist some networks in which all three algorithms failed to obtain the result, such as, the 33-node, 36-node, 37-node, and 39-node networks. The reason is that the BDD size of the forward reachable set is too large, leading to OOM. Finally, the performance of these algorithms depends on not only the number of nodes but also the structure and Boolean functions of the network. For example, **filtBDD** failed with the 36-node network but still succeeded with the 40-node network.

Table 3.2: Experimental results of **FR-BR-BDD-1**, **FR-BR-BDD-2**, and **filtBDD** on artificial networks.

Name	Number × size of attractors	**FR-BR-BDD-1** (seconds)	**FR-BR-BDD-2** (seconds)	**filtBDD** (seconds)
20-node	2 × 6144, 2 × 12288	84.430	1.321	0.480
22-node	1 × 4194304	222.753	51.836	7.431
24-node	2 × 1	-	0.332	0.136
26-node	1 × 1048576	-	46.588	3.466
28-node	6 × 1	-	1.328	1.226
30-node	4 × 1	-	-	14.674
31-node	1 × 4096, 1 × 8192	-	-	6.083
32-node	4 × 128, 2 × 256	-	-	212.927
33-node		-	-	-
34-node	5 × 1	-	-	81.797
35-node	2 × 32	-	-	127.547
36-node		-	-	-
37-node		-	-	-
38-node	1 × 32	-	0.272	0.234
39-node		-	-	-
40-node	4 × 6	-	-	0.307

3.5 Near-Exact Algorithm Using SAT-Based Bounded Model Checking

3.5.1 Algorithm filtSAT

filtBDD must first calculate forward reachable sets. This calculation may be extremely long if the forward reachable sets are too large. Moreover, this calculation is based on BDDs. Therefore, **filtBDD** may encounter OOM that is an inherent problem of BDD-based methods. To overcome these problems, we here use SAT-based BMC [110] for checking the reachability in GABNs without calculating forward reachable sets. In addition, we do not need to calculate all states of an attractor; at least one state in the attractor is enough. The reason is that we can enumerate the attractor by listing all other states reachable from one state in this attractor. Then, we propose a SAT-based algorithm called **filtSAT** for approximating attractors of a GABN.

Since **filtSAT** uses SAT-based BMC to check the reachability property, a good bound d is important. The *diameter* of a graph is the length of the longest shortest path between two states. In the field of model checking, it represents the smallest number of steps that are needed to reach all reachable states. We here define a new concept called *diameter of attraction* inspired by the concept of diameter in model checking. The diameter of attraction of an STG is defined as the length of the longest shortest path from a state to an attractor of the STG. Let $d_{\mathcal{N}}$ denote the diameter of attractions of the STG of a BN $\mathcal{N}$. Consider the BN $\mathcal{N}$ shown in Example 2.1.1. Let $\mathcal{S}$ and $\mathcal{G}$ be the SBN and GABN counterparts of $\mathcal{N}$, respectively. Then, we have $d_{\mathcal{S}} = d_{\mathcal{G}} = 1$ (see Figure 2.1).

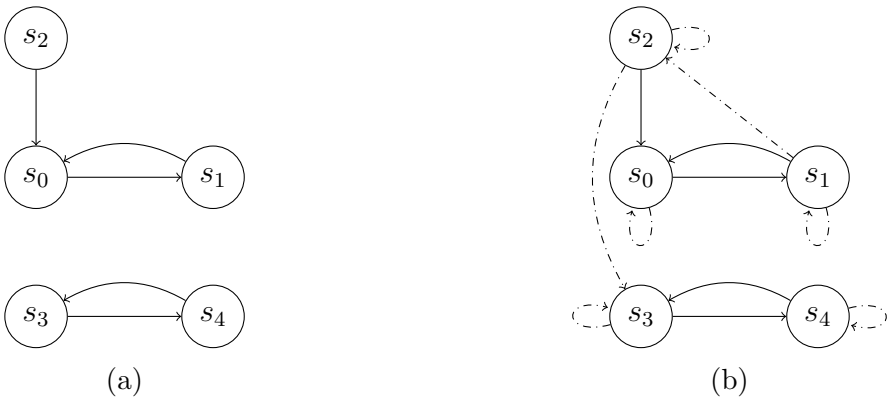


Figure 3.1: A counter example for the claim that $d_{\mathcal{S}} \geq d_{\mathcal{G}}$. (a) and (b) show parts of the STGs of the SBN $\mathcal{S}$ and the GABN $\mathcal{G}$, respectively. Herein, $d_{\mathcal{S}} = 1$ (the longest shortest path is, for example, $s_2 \rightarrow s_0$), whereas $d_{\mathcal{G}} = 3$ (the longest shortest path is, for example, $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3$).

Obviously, d must be at least $d_{\mathcal{G}}$ to ensure the completeness of checking the reachability in the GABN $\mathcal{G}$. However, $d_{\mathcal{G}}$ is very hard to compute. Even to compute its upper bound (the diameter of the STG of $\mathcal{G}$), we need to use a graph algorithm that is polynomial in the size of the STG or a Quantified Boolean Formula (QBF) [110]. Unfortunately, the size of the STG of a GABN is exponential in its number of nodes and QBF is PSPACE-complete. Hence, we need a bound that is good enough and easy to compute. Although $d_{\mathcal{S}} \geq d_{\mathcal{G}}$ does not always hold (see Figure 3.1), $d_{\mathcal{S}}$ may be used as an efficient bound because of the following reasons. First, since the outdegree of a state in a GABN is high and the STG of the GABN contains all arcs of the STG of its SBN counterpart, if the reachability property actually holds then there is a high probability that the reachability property holds with the bound $d_{\mathcal{S}}$. Second, $d_{\mathcal{S}}$ can be easily computed in the process of finding SBN attractors. This reason is justified in the last paragraph of this subsection.

Algorithm 4 shows the description of **filtSAT**. $\chi(F, s^i)$ is the characteristic formula representing the set F of states in term of variables of state s^i. The characteristic formula of a set of states is defined based on the characteristic formula of a state: $\chi(F, s^i) = \bigvee_{s \in F} \chi(s, s^i)$. The characteristic formula of a state s in term of variables of state s^i is defined as $\chi(s, s^i) = \bigwedge_{j=1}^{n} (s_j^i \leftrightarrow s_j)$. For example, if $att = \{00, 11\}$, then $\chi(att, s^0) = (s_1^0 \leftrightarrow 0 \wedge s_2^0 \leftrightarrow 0) \vee (s_1^0 \leftrightarrow 1 \wedge s_2^0 \leftrightarrow 1)$. The formula ϕ represents a path with length d from a state in $att^{\mathcal{S}}$ to a state in $flatten(A \cup A_{type2}^{\mathcal{S}})$. Since $A \cup A_{type2}^{\mathcal{S}}$ is a set of sets of states, we need to flatten it into a set of states. If the path does not exist (i.e., SAT(ϕ) = false), $att^{\mathcal{S}}$ corresponds to an attractor of the GABN $\mathcal{G}$ and is added to A. Correspondingly, the state s randomly picked from $att^{\mathcal{S}}$ is added to $F_{comp}^{\mathcal{G}}$. At Line 16 of Algorithm 4, we use $att^{\mathcal{S}}$ instead of $FR^{\mathcal{G}}(att^{\mathcal{S}})$ because (1) $FR^{\mathcal{G}}(att^{\mathcal{S}})$ may be very large and we may take much time/memory for calculating it; (2) when known at least one state of an attractor, we can easily calculate all its states if needed. This is also one of the differences between **filtSAT** and **filtBDD**. Note that, by Lemmas 3.3.2 and 3.3.3, A can start with the set of singleton and type1 attractors of the SBN $\mathcal{S}$. This can improve the efficiency of **filtSAT** by two reasons: (1) singleton and type1 attractors can be easily classified in the process of finding SBN attractors; (2) we do not need to take time for checking the reachability property for these attractors by solving ϕ.

The result of **filtSAT** includes $A_{sing}^{\mathcal{G}}$ (the set of singleton attractors of $\mathcal{G}$), $A_{type1}^{\mathcal{G}}$ (the

Algorithm 4 filtSAT

Input: A GABN $\mathcal{G}$.
Output: $A^{\mathcal{G}}_{sing}$, $A^{\mathcal{G}}_{type1}$, $F^{\mathcal{G}}_{comp}$.

1: Let $\mathcal{S}$ be the SBN counterpart of $\mathcal{G}$
2: Compute all attractors of $\mathcal{S}$
3: $A^{\mathcal{S}}_{sing} \leftarrow$ the set of singleton attractors of $\mathcal{S}$
4: $A^{\mathcal{S}}_{type1} \leftarrow$ the set of type1 attractors of $\mathcal{S}$
5: $A^{\mathcal{S}}_{type2} \leftarrow$ the set of type2 attractors of $\mathcal{S}$
6: $A^{\mathcal{G}}_{sing} \leftarrow A^{\mathcal{S}}_{sing}$
7: $A^{\mathcal{G}}_{type1} \leftarrow A^{\mathcal{S}}_{type1}$
8: $F^{\mathcal{G}}_{comp} \leftarrow \emptyset$
9: $A \leftarrow A^{\mathcal{S}}_{self} \cup A^{\mathcal{S}}_{type1}$
10: $d \leftarrow$ the diameter of attraction of $\mathcal{S}$
11: **while** $A^{\mathcal{S}}_{type2} \neq \emptyset$ **do**
12: Randomly remove a type2 attractor $att^{\mathcal{S}}$ from $A^{\mathcal{S}}_{type2}$
13: $\phi_{path} \leftarrow \bigwedge_{i=0}^{d-1} \mathcal{T}(s^i, s^{i+1})$
14: $\phi \leftarrow \chi(att^{\mathcal{S}}, s^0) \wedge \phi_{path} \wedge \chi(flatten(A \cup A^{\mathcal{S}}_{type2}), s^d)$
15: **if** SAT(ϕ) = false **then**
16: $A \leftarrow A \cup att^{\mathcal{S}}$
17: Randomly pick a state s in $att^{\mathcal{S}}$
18: $F^{\mathcal{G}}_{comp} \leftarrow F^{\mathcal{G}}_{comp} \cup \{s\}$
19: **end if**
20: **end while**
21: **return** $A^{\mathcal{G}}_{sing}$, $A^{\mathcal{G}}_{type1}$, $F^{\mathcal{G}}_{comp}$

set of type1 attractors of $\mathcal{G}$), and $F^{\mathcal{G}}_{comp}$ (a set of states of $\mathcal{G}$). We say that $F^{\mathcal{G}}_{comp}$ covers an attractor att if there is a state in $F^{\mathcal{G}}_{comp}$ such that this state belongs to att (formally, $F^{\mathcal{G}}_{comp} \cap att \neq \emptyset$). $F^{\mathcal{G}}_{comp}$ is expected to cover exactly all the complex attractors of $\mathcal{G}$ (i.e., $\forall att \in A^{\mathcal{G}}_{comp}, F^{\mathcal{G}}_{comp} \cap att \neq \emptyset$ and $|A^{\mathcal{G}}_{comp}| = |F^{\mathcal{G}}_{comp}|$). Theorem 3.5.1 guarantees that all the attractors of a GABN are always found by **filtSAT**. *However, the result of **filtSAT** may contain some redundant or spurious attractors* (see Case 1-A and Case 1-B in the proof of Theorem 3.5.1, respectively). Case 1-A and Case 1-B occur when $att^{\mathcal{S}}$ reaches $A \cup A^{\mathcal{S}}_{type2}$ but SAT(ϕ) = false holds because the bound d is not high enough. If Case 1-A and Case 1-B never occur, **filtSAT** can find exactly all attractors of the GABN. In this case, we have a one-to-one correspondence between $F^{\mathcal{G}}_{comp}$ and $A^{\mathcal{G}}_{comp}$ (i.e., the set of complex attractors of $\mathcal{G}$).

Theorem 3.5.1. *Let $\mathcal{G}$ be a GABN. Then, all the attractors of $\mathcal{G}$ are covered by the result obtained by applying **filtSAT** to $\mathcal{G}$.*

Proof. By Lemmas 3.3.2 and 3.3.3, the result of **filtSAT** contains all singleton and type1 attractors of $\mathcal{G}$. Hereafter, we only consider complex attractors of $\mathcal{G}$.

After finishing Lines 1-10 of Algorithm 4, any complex attractor of $\mathcal{G}$ contains at least one SBN attractor in $A \cup A^{\mathcal{S}}_{type2}$ by Theorem 3.3.1 (*). In the first iteration of the while loop (Lines 11-20 of Algorithm 4), we have all the possible cases as follows.
Case 1: SAT(ϕ) = false. Obviously, (*) still holds because $A \cup A^{\mathcal{S}}_{type2}$ is not changed. However, there are two possible subcases that may make the result of **filtSAT** incorrect.

Case 1-A: $att^{\mathcal{S}}$ is contained in an attractor $att^{\mathcal{G}}$ of $\mathcal{G}$ and there is an $att' \in A$ contained in $att^{\mathcal{G}}$. Then, $att^{\mathcal{S}}$ will be a redundant attractor.
Case 1-B: $att^{\mathcal{S}}$ is not contained in any attractor of $\mathcal{G}$. Then, att will be a spurious attractor.
Case 2: SAT(ϕ) = true. By the theory of bounded model checking, $att^{\mathcal{S}}$ reaches $A \cup A^{\mathcal{S}}_{type2}$ in the STG of $\mathcal{G}$. There are all two possible subcases.
Case 2-A: $att^{\mathcal{S}}$ is contained in an attractor $att^{\mathcal{G}}$ of $\mathcal{G}$. There is an $att' \in A \cup A^{\mathcal{S}}_{type2}$ such that $att' \subseteq FR^{\mathcal{G}}(att^{\mathcal{S}})$. $att' \subseteq att^{\mathcal{G}}$ because $FR^{\mathcal{G}}(att^{\mathcal{S}}) = att^{\mathcal{G}}$. Thus, (*) still holds.
Case 2-B: $att^{\mathcal{S}}$ is not contained in any attractor of $\mathcal{G}$. Obviously, (*) still holds.

We obtain that (*) is preserved after finishing the first iteration. Similarly, (*) is also preserved after finishing the last iteration. Now, $A^{\mathcal{S}}_{type2} = \emptyset$, thus any complex attractor of $\mathcal{G}$ contains at least one attractor in A. Since the attractors of $\mathcal{G}$ are pairwise disjoint, an attractor in A cannot be contained in two attractors of $\mathcal{G}$. Hence, A covers all the complex attractors of $\mathcal{G}$. Correspondingly, $F^{\mathcal{G}}_{comp}$ covers all the complex attractors of $\mathcal{G}$. □

Let l be the number of type2 attractors of the SBN $\mathcal{S}$ (i.e., $l = |A^{\mathcal{S}}_{type2}|$). Obviously, the number of iterations of **filtSAT** is always l. Since **filtSAT** is an SAT-based algorithm, its efficiency obviously depends largely on the number of variables and the number of clauses of the formula ϕ. From the description of **filtSAT**, the number of variables is $n \times d$. It is hard to estimate exactly the number of clauses because we cannot estimate exactly the characteristic formulas of $att^{\mathcal{S}}$ and $flatten(A \cup A^{\mathcal{S}}_{type2})$. However, we can realize that the number of clauses is proportional to $n \times l \times d$. Hence, the efficiency of **filtSAT** depends on not only the network size but also the number of SBN type2 attractors and the diameter of attraction. Note that, as many previous methods for BNs, the efficiency of **filtSAT** also depends on the Boolean functions.

Finally, we discuss two possible methods for calculating $d_{\mathcal{S}}$. Let ALG be the algorithm for finding SBN attractors. $d_{\mathcal{S}}$ can be easily calculated in the running of ALG. If ALG is a BDD-based method (e.g., [43]) then $d_{\mathcal{S}} = max\{t|t$ is the number of iterations in a call of the *backward_set* funtion$\}$. If ALG is an SAT-based method (e.g., [44]) then $d_{\mathcal{S}}$ can be calculated by a simple procedure as follows. First, $p \leftarrow p_{start}$ and $atts \leftarrow$ the set of states flattened from all the attractors of $\mathcal{S}$. Second, if there exists a path of length p from s_0 to s_p such that $s_p \notin atts$, then $p \leftarrow p + step$ and go back to the first step. Otherwise, $d_{\mathcal{S}} \leftarrow p$ and the procedure terminates. In this dissertation, we use the SAT-based method by [44] for finding SBN attractors. If we set $p_{start} \leftarrow 1$ and $step \leftarrow 1$, we can get exactly the value of $d_{\mathcal{S}}$. However, the computational time of this procedure may be long because the number of iterations is $d_{\mathcal{S}}$. Therefore, we empirically set $p_{start} \leftarrow min(p_d/2, 20)$, where p_d is the depth of unfolding (see [44]) and $step \leftarrow 2$. Herein, an upper bound of $d_{\mathcal{S}}$ is obtained.

3.5.2 Evaluation

We have implemented the proposed SAT-based method **filtSAT**. The implementation is in JAVA language, uses JDD library [112] for BDD manipulation, and uses Z3 [118] as the SAT solver. We then conducted experiments to evaluate the performance of **filtBDD** and **filtSAT**. Note that each of these algorithms includes two steps: the first step is the computation of SBN attractors and the second one is the computation of GABN attractors. The evaluation metric is only the computational time of the second step because **filtBDD** and **filtSAT** use the same algorithm for the first step. An SBN is

much simpler than its GABN counterpart. Therefore, the time limit for the overall was set to three hours, whereas the time limit for the fist step was set to two hours.

All the experiments were run on a computer whose environment is CPU: Intel Core i7 2.4 GHz, Memory: 16 GB, Windows 10 Home 64 bit. We used two sets of Boolean networks. The first set includes 25 BNs of 25 real biological networks obtained from [119]. The second set includes random BNs generated with **Bool Net R** package [117]. Note that a JAVA program will release OOM when it tries to allocate amount of memory exceeding the available heap space. In our experiments, when we set the heap size to 4 GB, **filtSAT** never met OOM, whereas **filtBDD** met that before exceeding the time limit in some of the BNs. Therefore, we here set the heap size to 6 GB. With this heap size, both **filtBDD** and **filtSAT** never met OOM before exceeding the time limit.

Table 3.3 shows experimental results on the BNs of the real biological networks. Columns 1 and 2 denote the name and the number of nodes of the network (i.e., n), respectively. Column 3 denotes the number of singleton and type1 attractors (a_1). Columns 4 and 6 denote the numbers of GABN attractors computed by **filtBDD** and **filtSAT**, respectively. Columns 5 and 7 denote the computational time (in seconds) of **filtBDD** and **filtSAT**, respectively. "-" denotes the case of timeout. We here omit the 12/25 networks in which both **filtBDD** and **filtSAT** failed to compute SBN attractors within two hours. These 12 networks include Signaling pathway for Butanol Production ($n = 66$), CD4 T Cell Signaling ($n = 188$), EGFR & ErbB Signaling ($n = 104$), Glucose Repression Signaling 2009 ($n = 73$), HGF Signaling in Keratinocytes ($n = 68$), IL-1 Signaling ($n = 118$), IL-6 Signaling ($n = 86$), Influenza A Virus Replication Cycle ($n = 131$), Lymphopoiesis Regulatory Network ($n = 81$), Signal Transduction in Fibroblasts ($n = 139$), Signaling in Macrophage Activation ($n = 321$), and Yeast Apoptosis ($n = 73$).

From Table 3.3, we report two observations as follows. First, in the 7/13 networks in which both **filtBDD** and **filtSAT** succeed to obtain the result within three hours, the numbers of GABN attractors computed by **filtBDD** and **filtSAT** are the same. In addition, the computational time of **filtSAT** and **filtBDD** is comparable. Second, in all the 6/13 remaining networks, **filtSAT** succeeded to obtain the result within three hours, whereas **filtBDD** failed. In particular, for Bordetella Bronchiseptica, we can use some theoretical results previously presented to reason the number of GABN attractors. Let a_3 be the number of GABN attractors. Then, $a_3 \geq a_1$ by Lemmas 3.3.2 and 3.3.3. Let a_{SAT} be the number of attractors computed by **filtSAT**. Then, $a_3 \leq a_{SAT}$ by Theorem 3.5.1. If $a_1 = a_{SAT}$, we can imply that $a_3 = a_{SAT}$. Herein, $a_1 = a_{SAT} = 3$, hence $a_3 = 3$, i.e., the number of attractors computed by **filtSAT** is correct. These observations are evidence for the effectiveness of **filtSAT** as compared to **filtBDD**.

Next, we randomly generated a set of BNs with network size n in the set {40, 50, 60, 70, 80, 90, 100, 200, 300, 400}. For each network size, five instances of N-K BNs [22] and five instances of scale-free BNs [120] were generated. N-K and scale-free BNs are two popular topology-based variations of BNs. In an N-K BN, each node has exactly K input nodes. In a scale-free BN, the number of input nodes for each node follows the scale-free Zeta distribution [120]. We set the maximum number of input nodes for each node to three for scale-free BNs and $K = 2$ for N-K BNs. In total, we have 100 random BNs.

Tables 3.4 and 3.5 show the experimental results on the randomly generated N-K and scale-free BNs, respectively. We omit the 38/100 BNs in which both **filtBDD** and **filtSAT** failed to compute SBN attractors within two hours. Almost of these BNs are N-K BNs

Table 3.3: Experimental results of **filtBDD** and **filtSAT** on real biological networks.

			filtBDD		**filtSAT**	
Network name	n	a_1	# atts	time (s)	# atts	time (s)
Bordetella Bronchiseptica	33	3	-	-	3	0.5
T-Cell Signaling 2006	40	7	-	-	8	0.8
Apoptosis Network	41	0	8	3.3	8	8.9
Guard Cell Abscisic Acid Signaling	44	16	28	6.9	28	19.3
Stomatal Opening Model	49	48	48	0.2	48	0.5
Senescence Associated Secretory Phenotype	51	15	17	0.8	17	3.0
B Bronchiseptica and T Retortaeformis Coinfection	53	30	30	34.8	30	8.8
MAPK Cancer Cell Fate Network	53	12	-	-	18	13.6
T-LGL Survival Network 2011	60	118	-	-	142	43.3
PC12 Cell Differentiation	62	3	3	0.2	3	0.4
Bortezomib Responses in U266 Human Myeloma Cells	67	83	83	0.4	83	26.4
Colitis-Associated Colon Cancer	70	2	-	-	5	113.7
T Cell Receptor Signaling	101	112	-	-	128	21.5

(even with $n = 100$). It is reasonable because the method for finding SBN attractor [44] is known inefficient for large-scale BNs with a relatively large average degree (i.e., average number of input nodes). Column "network name" denotes the name of the BN following the format n-i-instance number for N-K BNs or n-s-instance number for scale-free BNs. Columns "l" and "a_1" denote the number of type2 attractors and the number of singleton and type1 attractors of the SBN, respectively. Column "# attractors" denotes the number of GABN attractors, whereas Column "time (s)" denotes the computational time (in seconds). We classify the experimental results into four cases including Case 1: both **filtBDD** and **filtSAT** succeeded to detect all GABN attractors within three hours; Case 2: **filtBDD** failed to detect all GABN attractors, whereas **filtSAT** succeeded to detect those within three hours; Case 3: **filtSAT** failed to detect all GABN attractors, whereas **filtBDD** succeeded to detect those within three hours; and Case 4: both **filtBDD** and **filtSAT** failed to detect all GABN attractors within three hours. In total, we have 19/62 BNs of Case 1, 43/62 BNs of Case 2, 0/62 BN of Case 3, and 0/62 BNs of Case 4. From these results, we report two observations as follows. These observations are evidence for the effectiveness of **filtSAT** as compared to **filtBDD**.

First, in each BN of Case 1, the numbers of GABN attractors computed by **filtSAT** and **filtBDD** are the same. This is evidence for the accuracy of **filtSAT**. In addition, the computational time of **filtSAT** and **filtBDD** is comparable. Note that, in some large-scale BNs (e.g., 90-i-4, 90-s-4, 300-s-3), $l = 0$ means that the GABN has no complex attractors and **filtBDD** did not need to compute any forward reachable sets.

Second, in the 43 BNs of Case 2, **filtSAT** can handle large-scale networks (e.g., 300-s-4, 400-s-1) and even large-scale networks with a relatively large average degree (e.g., 80-i-5, 90-i-1). Furthermore, in the 9/43 networks (50-i-3, 60-i-1, 70-i-3, 80-i-5, 50-s-4, 70-s-3, 80-s-4, 200-s-3, 400-s-5), we can show that the result of **filtSAT** is correct by using the reasoning that has already presented in the analysis of the experimental results on real biological networks.

Table 3.4: Experimental results of **filtBDD** and **filtSAT** on N-K networks.

			filtBDD		**filtSAT**	
network name	l	a_1	# attractors	time (s)	# attractors	time (s)
40-i-2	1	0	1	0.2	1	0.3
40-i-5	1	1	1	2.0	1	0.3
40-i-4	2	1	3	8.2	3	0.5
40-i-3	5	0	-	-	1	1.0
40-i-1	10	2	2	0.8	2	3.1
50-i-2	4	1	1	36.4	1	2.0
50-i-4	6	0	4	0.3	4	3.7
50-i-5	6	0	-	-	1	4.6
50-i-3	7	2	-	-	2	3.1
50-i-1	32	0	2	221.1	2	28.8
60-i-3	3	0	-	-	1	3.3
60-i-5	7	1	-	-	2	11.1
60-i-2	8	4	4	4.2	4	3.8
60-i-4	13	0	-	-	2	9.9
60-i-1	16	2	-	-	2	12.7
70-i-4	0	1	1	0.2	1	0.2
70-i-3	3	1	-	-	1	3.1
70-i-2	9	1	-	-	2	9.1
80-i-3	4	0	-	-	1	12.6
80-i-2	7	1	-	-	2	9.4
80-i-5	14	3	-	-	3	16.5
80-i-4	18	0	-	-	3	22.4
90-i-4	0	3	3	0.3	3	0.4
90-i-3	2	0	-	-	1	1.8
90-i-1	5	0	-	-	1	2615.7

Moreover, as previously mentioned, there are some variables that may affect the efficiency of **filtSAT** such as the network size (i.e., n) and the number of SBN type2 attractors (i.e., l). From the experimental results, we can obtain the correlation between l and the computational time of **filtSAT**. We observed the computational time of **filtSAT** for the BNs with the same n and the same form of network topology (scale-free or N-K). In general, the computational time of **filtSAT** increases as l increases (see Tables 3.4 and 3.5). It is reasonable because the number of iterations of **filtSAT** is equal to l. This correlation suggests us that if we can early exclude from the filtering set (i.e., $A^{\mathcal{S}}_{type2}$) the SBN attractors that cannot be in any GABN attractor, we can significantly reduce the computational time of **filtSAT**. This is one of our future work.

Table 3.5: Experimental results of **filtBDD** and **filtSAT** on scale-free networks.

			filtBDD		**filtSAT**	
network name	l	a_1	# attractors	time (s)	# attractors	time (s)
40-s-2	8	0	-	-	1	2.4
40-s-1	13	7	7	0.2	7	2.1
40-s-4	28	5	6	728.0	6	5.6
40-s-5	40	0	-	-	4	5.4
40-s-3	208	0	-	-	2	104.0
50-s-2	2	0	-	-	2	0.3
50-s-5	2	0	-	-	1	0.4
50-s-3	11	1	-	-	2	2.5
50-s-4	24	4	-	-	4	4.9
50-s-1	240	0	8	3004.6	8	2314.4
60-s-3	3	0	-	-	1	0.7
60-s-4	8	0	-	-	2	1.8
60-s-5	11	0	-	-	2	11.2
60-s-1	152	0	-	-	1	351.9
60-s-2	422	5	-	-	6	991.8
70-s-1	8	8	8	0.2	8	4.4
70-s-4	10	4	4	0.4	4	3.5
70-s-3	38	4	-	-	4	27.6
70-s-2	265	1	-	-	3	742.4
80-s-1	8	0	1	212.9	1	6.5
80-s-2	48	0	-	-	2	14.3
80-s-4	212	4	-	-	4	568.2
90-s-4	0	1	1	0.2	1	0.2
90-s-5	2	0	-	-	1	1.3
90-s-1	24	0	-	-	2	29.8
90-s-3	24	0	2	0.5	2	30.7
90-s-2	675	1	-	-	2	8565.0
100-s-1	8	0	-	-	1	14.1
100-s-2	24	2	-	-	4	31.0
100-s-5	24	0	-	-	1	31.7
100-s-3	486	12	-	-	16	2400.5
200-s-3	3	1	-	-	1	31.4
300-s-3	0	2	2	0.2	2	0.6
300-s-4	24	0	-	-	2	727.3
400-s-2	32	0	-	-	1	2275.6
400-s-5	36	4	-	-	4	2247.4
400-s-1	204	0	-	-	8	10034.0

3.6 Relations in Dynamics between GABNs and Asynchronous Boolean Networks

3.6.1 Relations

Proposition 3.6.1. *Let $\mathcal{A}$ be an ABN and $\mathcal{G}$ be its GABN counterpart. Then $\mathcal{A}$ and $\mathcal{G}$ have the same set of singleton attractors.*

Proof. This proposition immediately holds because any type of BNs has the same set of singleton attractors [14]. □

Proposition 3.6.2. *Let $\mathcal{A}$ be an ABN and $\mathcal{G}$ be its GABN counterpart. Then $\mathcal{A}$ and $\mathcal{G}$ have the same set of type1 attractors.*

Proof. Let att^1 be a type1 attractor of $\mathcal{A}$. Let s be an arbitrary state in att^1 and s' be its successor (excluding s). Since s and s' differ in exactly one bit (say the ith bit), only node x_i changes its value. Thus, the updating of some nodes in G changes the ith bit (node x_i is in the set of updated nodes) or no bit (node x_i is not in the set of updated nodes) of s. This implies that $FI^{\mathcal{G}}(\{s\}) = \{s, s'\}$. Since s is arbitrary, att^1 is also a type1 attractor of $\mathcal{G}$ (*).

Let att^2 be a type1 attractor of $\mathcal{G}$. Let s be an arbitrary state in att^2 and s' be a successor state (excluding s) of s. Then $FI^{\mathcal{G}}(\{s\}) = \{s, s'\}$. Since $\mathcal{G}$ can update any number of nodes synchronously, we have $FI^{\mathcal{A}}(\{s\}) \subseteq FI^{\mathcal{G}}(\{s\})$. $FI^{\mathcal{A}}(\{s\}) = \{s'\}$ or $FI^{\mathcal{A}}(\{s\}) = \{s, s'\}$ because if $FI^{\mathcal{A}}(\{s\}) = \{s\}$ then $FI^{\mathcal{G}}(\{s\}) = \{s\}$. Since s is arbitrary, att^2 is also a type1 attractor of $\mathcal{A}$ (**).

From (*) and (**), we can conclude the proof. □

Proposition 3.6.3. *Let $\mathcal{A}$ be an ABN and $\mathcal{G}$ be its GABN counterpart. Then $FR^{\mathcal{A}}(\{s\}) \subseteq FR^{\mathcal{G}}(\{s\})$ holds for any state s of $\mathcal{A}$.*

Proof. Since $\mathcal{G}$ can update any number of nodes synchronously, we have $FI^{\mathcal{A}}(\{s\}) \subseteq FI^{\mathcal{G}}(\{s\})$ for any state s. By the definition of a forward reachable set, we obtain $FR^{\mathcal{A}}(\{s\}) \subseteq FR^{\mathcal{G}}(\{s\})$. □

Theorem 3.6.1. *Let $\mathcal{A}$ be an ABN and $\mathcal{G}$ be its GABN counterpart. Then there exists a mapping $m : A^{\mathcal{G}} \to A^{\mathcal{A}}$ with $m(att) \subseteq att$ for all $att \in A^{\mathcal{G}}$ and $m(att_1) \neq m(att_2)$ for all $att_1, att_2 \in A^{\mathcal{G}}, att_1 \neq att_2$. Intuitively, every attractor of $\mathcal{G}$ contains at least one attractor of $\mathcal{A}$.*

Proof. Let att be an attractor in $A^{\mathcal{G}}$ and $s \in att$. Clearly, $FR^{\mathcal{G}}(\{s\}) = att$. By Proposition 3.6.3, $FR^{\mathcal{A}}(\{s\}) \subseteq FR^{\mathcal{G}}(\{s\})$. Thus, $FR^{\mathcal{A}}(\{s\}) \subseteq att$.

Clearly, there is an attractor $att' \in A^{\mathcal{A}}$ such that $att' \subseteq FR^{\mathcal{A}}(\{s\})$. Thus, $att' \subseteq att$. Now, we can choose the mapping $m : A^{\mathcal{G}} \to A^{\mathcal{A}}$ such that $m(att) = att'$. Note that since att' may be not unique, the mapping m may not be uniquely determined.

Since attractors are pairwise disjoint, $m(att_1) \neq m(att_2)$ for all $att_1, att_2 \in A^{\mathcal{G}}$, $att_1 \neq att_2$. Therefore, we can conclude the proof. □

Corollary 3.6.1. *The number of attractors of an ABN is greater than or equal to that of its GABN counterpart.*

Proof. Let $\mathcal{A}$ be an ABN and $\mathcal{G}$ be its GABN counterpart. We show that $|A^{\mathcal{A}}| \geq |A^{\mathcal{G}}|$.

By Theorem 3.6.1, there is a mapping $m : A^{\mathcal{G}} \to A^{\mathcal{A}}$ with $m(att) \subseteq att$ for all $att \in A^{\mathcal{G}}$ and $m(att_1) \neq m(att_2)$ for all $att_1, att_2 \in A^{\mathcal{G}}, att_1 \neq att_2$. Obviously, m is an injection. Hence, $|A^{\mathcal{A}}| \geq |A^{\mathcal{G}}|$. □

We here report an example (Example 3.6.1) for the inequality case of Corollary 3.6.1. However, we observed that the equality case of Corollary 3.6.1 happens in most cases (see, e.g., [5]). This suggests us to propose an efficient method for approximating the attractors of an ABN based on the attractors of its GABN counterpart.

Example 3.6.1. *Consider a BN of three nodes associated to three variables (x_1, x_2, x_3). Its Boolean functions are given by*

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

The STGs of the GABN and ABN counterparts of this BN are given in Figure 3.2a and Figure 3.2b, respectively. As we can see, the ABN has two attractors ($\{011\}$ and $\{001, 101, 111, 110, 100\}$), whereas the GABN has only one attractor ($\{011\}$).

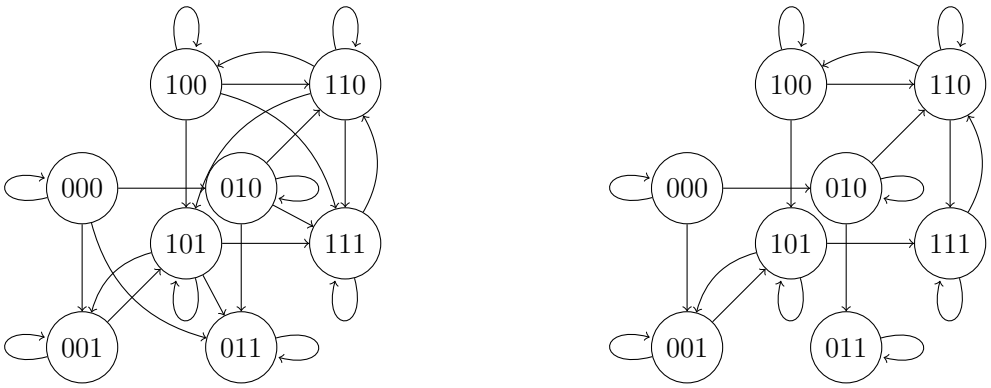


Figure 3.2: STGs of (a) the GABN counterpart and (b) the ABN counterpart of the BN shown in Example 3.6.1.

3.6.2 Application

As presented in Section 3.5, we have proposed an efficient method called **filtSAT** for finding attractors of GABNs. This method relies on the relations in dynamics between a GABN and its SBN counterpart. Although the result of **filtSAT** may be incorrect, its accuracy has been justified by the experiments on real biological and randomly generated networks. In particular, **filtSAT** can handle large-scale networks (e.g., the T_Cell_Receptor_Signaling network with $n = 101$ [119]).

Inspired by the effectiveness of **filtSAT** and the relations in dynamics between ABNs and GABNs presented in the previous subsection, we propose an efficient method called

ApproABN for approximating attractors of ABNs. The description of **ApproABN** is given in Algorithm 5. $A^{\mathcal{A}}_{sing}$ and $A^{\mathcal{A}}_{type1}$ are simply equal to $A^{\mathcal{G}}_{sing}$ and $A^{\mathcal{G}}_{type1}$ by Proposition 3.6.1 and Proposition 3.6.2, respectively. We now make a long random walk on the set $F^{\mathcal{G}}_{comp}$. A random step of this random walk is a state transition corresponding to the updating of a random node in the STG of $\mathcal{A}$. By Proposition 3.6.3, each state of $F^{\mathcal{G}}_{comp}$ will reach an attractor of $\mathcal{A}$. In addition, if $s, s' \in F^{\mathcal{G}}_{comp}$ belong to two different attractors of $\mathcal{G}$, then they reach two different attractors of $\mathcal{A}$ because attractors of a BN are pairwise disjoint. Making random walk aims at making $F^{\mathcal{G}}_{comp}$ approach attractor states of $\mathcal{A}$. In practice, we empirically used $I_{max} = n \times 20$ and found that it is enough to reach an attractor of $\mathcal{A}$. Finally, we obtain $F^{\mathcal{A}}_{comp}$ as the set of states of $\mathcal{A}$, and each state of $F^{\mathcal{A}}_{comp}$ is expected to represent (i.e., belong to) a complex attractor of $\mathcal{A}$. $F^{\mathcal{A}}_{comp}$ is sufficient because starting from a state s of an attractor of $\mathcal{A}$, we can enumerate another states of this attractor by listing all other states reachable from s (i.e., $FR^{\mathcal{A}}(\{s\})$, which can be easily implemented by using BDDs [43]).

Algorithm 5 ApproABN

Input: An ABN $\mathcal{A}$.
Output: $A^{\mathcal{A}}_{sing}$, $A^{\mathcal{A}}_{type1}$, $F^{\mathcal{A}}_{comp}$ is a set of states of $\mathcal{A}$.

$\mathcal{G} \leftarrow$ the GABN counterpart of $\mathcal{A}$
Apply **filtSAT** to $\mathcal{G}$
$A^{\mathcal{A}}_{sing} \leftarrow A^{\mathcal{G}}_{sing}$
$A^{\mathcal{A}}_{type1} \leftarrow A^{\mathcal{G}}_{type1}$
$F^{\mathcal{A}}_{comp} \leftarrow F^{\mathcal{G}}_{comp}$
$i \leftarrow 1$
while $i \leq I_{max}$ **do**
    Choose randomly a node $x_j \in \{x_1, ..., x_n\}$
    $F^{\mathcal{A}}_{comp} \leftarrow FI^{\mathcal{A}}_{x_j}(F^{\mathcal{A}}_{comp})$
    $i \leftarrow i + 1$
end while
return $A^{\mathcal{A}}_{sing}$, $A^{\mathcal{A}}_{typ1}$, $F^{\mathcal{A}}_{comp}$

Hereafter, we shall discuss the correctness of **ApproABN**. The result of **ApproABN** may be incorrect in some cases as follows. First, $F^{\mathcal{A}}_{comp}$ may contain redundant states belonging to the same attractor of $\mathcal{A}$ because $F^{\mathcal{G}}_{comp}$ may contain redundant states belonging to the same attractor of $\mathcal{G}$ (see Section 3.5). Second, $F^{\mathcal{A}}_{comp}$ also may contain spurious states that do not belong to any attractor of $\mathcal{A}$ because the random walk does not guarantee always reaching attractor states. Last, some attractors of $\mathcal{A}$ may not appear in $F^{\mathcal{A}}_{comp}$ because the number of attractors of $\mathcal{A}$ may be larger than that of $\mathcal{G}$ by Corollary 3.6.1.

The analysis in the previous paragraph is negative. However, the following reasons convince the accuracy of **ApproABN**. First, the accuracy of **filtSAT** has been justified by the experiments on many real biological and randomly generated networks (see Section 3.5). This means that the result of **filtSAT** does not contain spurious or redundant states in most cases. Consequently, **ApproABN** will not contain redundant states in most cases. Second, a long random walk with a sufficient I_{max} usually makes a state in $F^{\mathcal{G}}_{comp}$ reach an attractor of $\mathcal{A}$. Even a state in $F^{\mathcal{G}}_{comp}$ has already belonged to an attractor of $\mathcal{A}$. Last, the number of attractors of an ABN is equal to that of its GABN counterpart in most cases. Consequently, **ApproABN** will not miss any complex attractor of $\mathcal{A}$ in

most cases.

Finally, we discuss the time complexity of Algorithm 5. Since there are 2^{2^n} different Boolean functions with n variables, the representation of a Boolean function may need $\mathcal{O}(2^n)$ space. Herein, we follow two assumptions on Boolean functions used in BNs including (i) each Boolean function can be represented in a polynomial amount of space and (ii) evaluation of each Boolean function can be done in polynomial time. Algorithm 5 includes two parts. The first part is the **filtSAT** method. The second part is the long random walk whose time complexity is obviously polynomial with respect to n. The time complexity of **filtSAT** must be not polynomial with respect to n because the problem of finding a singleton attractor was proved NP-hard [121]. Thus, the time complexity of Algorithm 5 is equivalent to the time complexity of **filtSAT**. **filtSAT** uses Z3 [118] and some other tools as subroutines. Since the time complexity of such tools is unclear, it is difficult to analyze the time complexity of **filtSAT**. We leave the analysis of theoretical or practical computational complexity of **ApproABN** as future work.

3.6.3 Evaluation

We have implemented **ApproABN** in a JAVA tool. This tool uses the JDD library [112] for BDD manipulation and uses Z3 [118] as the SAT solver. We then conducted experiments on real biological networks obtained from the literature to evaluate the effectiveness of **ApproABN**. All the experiments were run on a virtual machine whose environment is CPU: Intel(R) Xeon(R) Silver 4116 4x2.10GHz, Memory: 24 GB, CentOS 7 64 bit.

We applied **ApproABN**, **genYsis** [43], and **CABEAN** [49] to BNs of 32 real biological networks whose sizes range from 19 to 101. The Boolean functions of these 32 BNs only include CONJUNCTION, DISJUNCTION, and NEGATION operators. We chose **genYsis** and **CABEAN** because they are exact and famous tools for finding attractors of ABNs. We used the executable files of **genYsis** and **CABEAN** that are online available. Note that the result of **genYsis** or **CABEAN** is a set of attractors (i.e., $A^{\mathcal{A}}$), whereas the result of **ApproABN** includes a set of singleton attractors (i.e., $A^{\mathcal{A}}_{sing}$), a set of type1 attractors (i.e., $A^{\mathcal{A}}_{type1}$), and a set of states (i.e., $F^{\mathcal{A}}_{comp}$). **CABEAN** requires to use a network reduction technique that removes all the *leaf nodes* [49] of a BN. This reduction technique fully conserves the attractors of an ABN [122]. To ensure the fairness of the experiments, we also used this reduction technique for all the considered methods. Let n' be the number of nodes of the reduced network. For finding SBN attractors, **ApproABN** uses the SAT-based method by [44] if $n' > 65$ and uses the decomposition-based method by [50] otherwise. Finally, the time limit for each BN was 10 hours because the running time may be very long for large-scale networks.

Table 3.6 shows the obtained results. Columns "name" and "n" denote the name and the number of nodes of a BN, respectively. Column "c" denotes the number of type2 attractors of the SBN counterpart of a BN. For each method (i.e., **ApproABN**, **genYsis**, or **CABEAN**), $|A|$ denotes the numbers of attractors obtained, "time" denotes the running time (in seconds). For **ApproABN**, Column "SBN time" represents the running time (in seconds) for finding SBN attractors of **filtSAT** and Column "Correct?" indicates whether **ApproABN** did or did not find exactly all the attractors of the ABN. Hereafter, we present the way for validating the result of **ApproABN** based on the result of **CABEAN** or **genYsis** (i.e., $A^{\mathcal{A}}$). If **CABEAN** succeeded to find attractors within the time limit, we would use its result. Otherwise, we would use the result of **genYsis**

if it succeeded to find attractors within the time limit. In the case that both **CABEAN** and **genYsis** failed to find attractors within the time limit, we cannot determine the correctness of **ApproABN** (i.e., N/A). Let $A^{\mathcal{A}}_{comp} = A^{\mathcal{A}} \backslash (A^{\mathcal{A}}_{sing} \cup A^{\mathcal{A}}_{type1})$. We define a number m as $m = |\{att \in A^{\mathcal{A}}_{comp} | att \cap F^{\mathcal{A}}_{comp} \neq \emptyset\}|$. If $m = |F^{\mathcal{A}}_{comp}|$ and $m = |A^{\mathcal{A}}_{comp}|$, then we can conclude that the result of **ApproABN** is correct (i.e., Yes). Otherwise, the result of **ApproABN** is incorrect (i.e., No). From the obtained results, we here report some remarks as follows.

First, there are 26/32 networks in which **CABEAN** or **genYsis** succeeded to find attractors within the time limit. In all these networks (except the T_Cell_Receptor_Signaling network), the result of **ApproABN** is always correct. In the T_Cell_Receptor_Signaling network, **genYsis** only returned the number of attractors, $A^{\mathcal{A}}$ was not obtained because there are some unmanageable large attractors. Hence, we cannot determine the correctness of **ApproABN** for this network. However, $|A|$ of **genYsis** is equal to $|A|$ of **ApproABN**. This result justifies the accuracy of **ApproABN**. Furthermore, since the accuracy of **filtSAT** has been justified (see Section 3.5), $|A|$ of **ApproABN** seems to be equal to the number of attractors of the GABN in most cases. This consolidates the observation that the number of attractors of an ABN is equal to that of its GABN counterpart in most cases [14].

Second, in the ApoptosisNetwork, HumanMyelomaCells, remy_tumorigenesis, and T_Cell_Receptor_Signaling networks, **CABEAN** failed to find attractors within the time limit, whereas **genYis** and **ApproABN** succeeded. In these networks, the running time of **ApproABN** is much less than that of **genYsis**. Especially in the T_Cell_Receptor_Signaling network with $n = 101$, the running time of **ApproABN** is even less than one minute. In the 28/32 remaining networks, **CABEAN** almost outperforms **genYsis**. Thus, we only compare **CABEAN** with **ApproABN**. In the IL_6_Signalling networks, both **CABEAN** and **ApproABN** failed to find attractors within the time limit. There is room for improvement. In the 5/28 networks (e.g., the ButanolProduction network with $n = 66$, the Colitis_associated_colon_cancer network with $n = 70$), **CABEAN** failed to find attractors within the time limit, whereas **ApproABN** succeeded. In particular, the running time of **ApproABN** for the Colitis_associated_colon_cancer network is even less than 15 minutes. In the 22/28 remaining networks, both **CABEAN** and **ApproABN** succeeded to find attractors within the time limit. In these networks, the running time of **ApproABN** is comparable to that of **CABEAN**. To sum up, these results are evidence for the time efficiency of **ApproABN** as compared to **genYsis** and **CABEAN**.

Third, in some networks (e.g., the Colitis_associated_colon_cancer and InflammatoryBowelDisease networks), we can see that most of the running time of **ApproABN** was spent for finding SBN attractors and the time for finding SBN attractors is large. Even in the IL_6_Signalling network, **ApproABN** failed to find SBN attractors within the time limit. We can easily see the high impact of finding SBN attractors on the performance of **ApproABN**. **ApproABN** uses the SAT-based method by [44] or the decomposition-based method by [50] for finding SBN attractors. These methods may be inefficient for the above networks. Using a more efficient method or efficiently combining multiple methods for finding SBN attractors may be a potential improvement.

Finally, in some networks (e.g., the Bcell, HGF_Signaling_in_Keratinocytes, Septation_Initiation_Network, and YeastApoptosis networks), the running time of **ApproABN** is much longer than that of **CABEAN**. In the ButanolProduction network, **ApproABN**

succeeded to obtain the result within the time limit, whereas both **genYsis** and **CABEAN** failed. However, the running time is quite long (more than seven hours). We can see that in these networks, the number of type2 attractors of the SBN counterpart is large. As mentioned in Section 3.5, the number of iterations of **filtSAT** is equal to the number of type2 attractors of the SBN counterpart. When this number is large, **ApproABN** may be inefficient. Hence, excluding redundant SBN attractors before the filtering of **ApproABN** may be a potential improvement.

3.7 Discussion

In this chapter, we have studied on several relations in dynamics between a GABN and its SBN counterpart. Based on these relations as well as the dynamical properties previously discovered, we have proposed three BDD-based algorithms for finding all attractors of a GABN by using attractors of its SBN counterpart. The first algorithm (called **FR-BR-BDD-1**) is inspired by the **genYsis** tool [98] that builds the total transition system and then calculates backward reachable sets to find attractors of an ABN. The second algorithm (called **FR-BR-BDD-2**) improves **FR-BR-BDD-1** by building only partial transition systems and calculating restricted backward reachable sets. The third algorithm (called **filtBDD**) also builds partial transition systems like **FR-BR-BDD-2**, but it does not calculate restricted backward reachable sets. This algorithm uses a filtering process. Experiments were also conducted to compare the performance of these algorithms. The experimental results show that **filtBDD** outperforms both **FR-BR-BDD-1** and **FR-BR-BDD-2**.

All **FR-BR-BDD-1**, **FR-BR-BDD-2**, and **filtBDD** must first calculate the forward reachable set. The computation of the forward reachable set may be extremely long even encounter OOM when the set becomes too large (usually when n is large). Hence, we have then proposed a new method (called **filtSAT**) to overcome this problem. **filtSAT** improves **filtBDD** by using SAT-based Bounded Model Checking (BMC) for checking the reachability in GABNs. A new concept called diameter of attraction has been proposed and used as a bound for the SAT-based BMC. Note that the result **filtSAT** is complete but not sound, whereas the result of **filtBDD** is exact. That is, any attractor of an GABN will be found by **filtSAT** but the result of **filtSAT** may contain spurious attractors. However, the experimental results on both randomly generated and real biological networks confirm the effectiveness (accuracy and efficiency) of **filtSAT** as compared to **filtBDD**.

Inspired by the above results, we have continued to explore relations in dynamics between GABNs and another popular type of BNs, ABNs. We have also obtained several results that highlight the applications of the study on GABNs. First, we have formally stated and proved several relations in dynamics between a GABN and its ABN counterpart. These theoretical results not only contribute to the understanding of dynamics of BNs but also can pave potential ways to analyze ABNs. We have developed a method called **ApproABN** for efficiently approximating attractors of large-scale ABNs. We have also conducted experiments on real biological networks obtained from the literature. The experimental results justify the accuracy of **ApproABN** and the efficiency of **ApproABN** as compared to the two state-of-the-art methods, **genYsis** and **CABEAN**. Furthermore, these results also consolidate the observation that the number of attractors of an ABN is equal to that of its GABN counterpart in most cases. This observation is

useful in studying dynamics of Boolean networks.

In many BNs of our experiments, most of the running time is spent for finding SBN attractors. Therefore, we can improve the proposed methods by using a more efficient method or efficiently combining multiple methods for finding SBN attractors. It is potentially possible because there have recently been many efficient methods (e.g., [48, 50]) for attractor detection in SBNs. Moreover, the number of iterations of the filtering process of **filtSAT** is equal to the number of type2 attractors of the SBN counterpart. **filtSAT** may be inefficient when this number is large. Hence, excluding redundant SBN attractors before the filtering process of **filtSAT** may be a potential improvement to **filtSAT** (consequently **ApproABN**). It is also interesting to analyze the theoretical or practical computational complexity of the proposed methods.

Roughly speaking, a GABN can be an intermediate model between its SBN and ABN counterparts. The relations in dynamics between GABNs and SBNs as well as between GABNs and ABNs have been explored in this chapter. However, to our best knowledge, there is no existing work linking the cyclic attractors of an SBN and those of its ABN counterpart. Hence, exploring the connection between SBNs and ABNs is theoretically interesting and one of our future work. Furthermore, we also plan to extend our obtained results to those for multi-valued networks that are an extension of BNs where each node can receive more than two values.

Table 3.6: Experimental results of **ApproABN**, **genYsis**, and **CABEAN** on real biological networks. "-" stands for the case of timeout.

			genYsis		**CABEAN**		**ApproABN**			
name	n	c	$\|A\|$	time	$\|A\|$	time	SBN time	$\|A\|$	time	Correct?
ApoptosisNetwork [119]	41	25	8	550.57	-	-	1.12	8	6.52	Yes
AuroraKinaseA [119]	23	36	32	2.69	32	0.48	0.56	32	5.29	Yes
B_bronchiseptica_T_retortaeformis [119]	53	18	30	3468.22	30	421.04	57.99	30	60.55	Yes
Bcell [123]	72	808	72	8543.63	72	29.35	85.50	72	1115.82	Yes
Bordetella_bronchiseptica [119]	33	1	3	1.88	3	1.73	0.87	3	1.41	Yes
ButanolProduction [119]	66	13312	-	-	-	-	100.02	8192	25711.04	N/A
CholesterolRegulatoryPathway [119]	34	17	4	4.78	4	0.47	1.17	4	2.82	Yes
Colitis_associated_colon_cancer [119]	70	84	-	-	-	-	701.90	10	873.26	N/A
dahlhaus_neuroplastoma [124]	23	34	32	2.69	32	0.51	0.54	32	4.93	Yes
Differentiation_of_T_lymphocytes [119]	50	240	-	-	2050	85.18	14.30	2050	251.97	Yes
Drosophila [125]	52	0	-	-	128	1980.41	60.55	128	66.33	Yes
FA_BRCA_pathway [119]	28	0	1	15.14	1	4.15	1.02	1	1.41	Yes
GuardCellAbscisicAcidSignaling [119]	44	92	28	7.60	28	0.82	0.55	28	7.71	Yes
HGF_Signaling_in_Keratinocytes [119]	68	2757	72	1158.00	72	8.47	26.20	72	8685.93	Yes
HumanMyelomaCells [119]	67	139	83	12656.01	-	-	9.18	83	16.81	Yes
IL_6_Signalling [119]	86	-	-	-	-	-	-	-	-	-
InflammatoryBowelDisease [119]	47	1	-	-	-	-	11944.1	1	11948.98	N/A
Lymphoid_myeloid_cell_specification [119]	33	0	21	21.45	21	3.32	1.11	21	1.94	Yes
MAPK [119]	53	28	-	-	-	-	17.16	18	25.15	N/A
OxidativeStressPathway [119]	19	1	2	1.43	2	0.72	0.07	2	0.32	Yes
PC12CellDifferentiation [119]	62	0	3	4.78	3	0.59	0.83	3	2.44	Yes
remy_tumorigenesis [126]	35	42	25	15.36	-	-	0.90	25	4.86	Yes
Senescence [119]	51	4	17	17.73	17	2.98	0.66	17	3.07	Yes
Septation_Initiation_Network [119]	31	1600	640	49.58	640	1.21	1.47	640	225.30	Yes
Stomatal_Opening_Model [119]	49	12	48	29.36	48	2.35	0.83	48	5.01	Yes
T_Cell_Receptor_Signaling [119]	101	48	128	3369.89	-	-	3.75	128	20.84	N/A
TcellLGL [119]	60	98	142	20987.68	142	890.47	54.75	142	81.18	Yes
TCellSignaling [119]	40	3	8	0.13	8	0.03	0.02	8	0.25	Yes
TLGLSurvival [119]	61	268	-	-	-	-	90.16	318	199.96	N/A
Treatment_of_Castration_Resistant [119]	42	0	16384	18.53	16384	0.71	4.28	16384	4.41	Yes
TumourCell [119]	32	6	9	2.06	9	0.51	0.54	9	0.85	Yes
YeastApoptosis [119]	73	9728	8448	45.45	8448	1.15	14.44	8448	1366.37	Yes

Chapter 4

Attractor Detection in Large-Scale Asynchronous Boolean Networks

4.1 Introduction

There are two main types of BNs usually used for modeling biological networks: Synchronous BNs (SBNs) and Asynchronous BNs (ABNs). The updating scheme of SBNs is that all the nodes are updated simultaneously at each time step [43]. The updating scheme of ABNs is that only one node is randomly and uniformly selected in order to be updated at each time step [43]. In biology, the updating process of each gene may spend various time from fractions of a second to hours [9]. Moreover, the information on time scales of components is usually lacking [9]. Hence, ABNs are considered more suitable [9, 51] for representing various time scales as well as dealing with the lack of knowledge on time scales. However, whereas many efficient algorithms and tools (see, e.g., [43, 44, 47, 48, 50]) have been developed for attractor detection in SBNs, few methods (see, e.g., [43, 45, 49]) have been proposed for attractor detection in ABNs due to the high complexity of the STG of an ABN. A more detailed literature review on computational methods for attractor detection in BNs is provided in Section 4.2. Moreover, the efficiency of these few methods is strictly prevented when the ABN becomes large, e.g., the number of nodes is over 100. Therefore, it is important and interesting to develop efficient methods that can handle larger ABNs.

Inspired by the principle of reducing dynamics by [45], we propose a new method to handle attractor detection in ABNs, especially for large ones (e.g., networks of over 100 nodes). The main idea of our method is similar to the idea of [45]. However, we here present several generalized and improved results in both theoretical and practical aspects. We first state and prove several relations between a Feedback Vertex Set (FVS) of the interaction graph of a BN and the dynamics of the BN (Section 4.3). From these relations, we propose an FVS-based method for detecting all possible attractors of an ABN (Section 4.4). Our approach relies on an FVS of this ABN to get a candidate set of states such that each attractor of the ABN contains at least one state of this set. We then filter out the candidate set by performing the reachability analysis on the ABN. Our method includes several constituent steps. For each step, we formalize the corresponding problems, analyze them, and propose efficient solutions for them. The correctness of our method is formally proved. We also propose several preprocessing procedures to reduce the computational burden but the correctness of our method is still preserved. The

experimental results confirm the usefulness of the preprocessing procedures and are very promising (Section 4.5).

Furthermore, we continue to propose an improved method that contains two substantial enhancements to our method (Section 4.6). The first enhancement is a reasonable combination of multiple previous techniques for checking the reachability in ABNs. The second enhancement is to use an NFVS instead of an FVS to get the candidate set of states. The theoretical foundation of this enhancement is a new theorem on a relation between an NFVS of the interaction graph of an ABN and the dynamics of this ABN. We also conduct experiments on various types of networks to evaluate the efficiency of the two proposed enhancements. The experimental results show that the two enhancements are effective and the improved method outperforms the original method. In particular, the improved method can handle large networks with up to 1000 nodes in terms of randomly generated networks and 321 nodes in terms of real biological networks.

4.2 Related Work

Attractor detection in BNs is a challenging problem that has been attracted much attention from researchers in various fields, such as, systems biology, physics, mathematics, formal methods. In this section, we briefly review several notably previous work related to this problem.

The set of fixed points of a BN is invariant with respect to the updating scheme [14]. In other words, a BN and any its counterpart (e.g., ABN or GABN counterpart) have the same set of fixed points. The computation of fixed points of a BN is also the simplest case of the attractor detection in BNs. Many efficient methods [53, 127, 128, 129, 130] have been proposed for this simplest case. Hereafter, we consider the general case of the attractor detection in BNs, i.e., finding all possible attractors (fixed points and cyclic attractors) of a BN.

Many algorithms and tools have been developed in the efforts to efficiently solve attractor detection in BNs. For small networks (e.g., $n \leq 50$), attractors can be simply detected by various enumeration and simulation methods [47, 127, 131]. For larger networks, attractors can be efficiently detected with two techniques including BDDs and SAT. In BDD-based methods [43, 98], the transition relation of a BN is encoded with BDDs and the calculation of attractors exploits advantages of the efficient BDD operations. However, these methods still rely on the exhaustive traversal of the whole state space, making their efficiency strictly prevented when the BN becomes large, e.g., n is over 100. SAT-based methods [44, 93, 132] encode the attractor detection problem as a satisfiability problem, then exploit the efficient implementation of SAT solvers. They can handle larger networks within shorter time as compared to BDD-based methods. There are also some methods [46, 47, 50, 127, 133] exploiting the relations between the network structure and the network dynamics.

The above-mentioned methods are mainly designed for SBNs. Few methods have been proposed in the efforts to efficiently solve attractor detection in ABNs. The first effort is the BDD-based method [98] as an extension of that for SBNs. The authors then improved this method by exploiting the relations between attractors of an ABN and attractors of its SBN counterpart [43]. Their implemented tool (called **genYsis**) has also been used widely in many research communities. There is also work [108] that

slightly improves **genYsis**. In the field of formal methods, there are also some recent work [134, 135] enhancing the BDD-based approach by symbolic state-space reduction techniques. However, these methods [134, 135] are still basically based on the computation of bottom SCCs of the STG of an ABN. This characteristic limits their efficiency. In 2011, Skodawessely proposed a method [45] to detect ABN attractors based on FVSs and the principle of reducing dynamics. This method seems to be promising but it can only handle networks of up to 38 nodes. Recently, a decomposition-based method [49] was proposed to deal with large ABNs. This method decomposes a large ABN into smaller components (called *blocks*) based on the network structure, detects attractors in these blocks, and then recovers the attractors of the original ABN. This method then was enhanced by a technique for optimizing the decomposition of an ABN [136]. However, the efficiency of the decomposition-based method largely depends on the sizes of the decomposed blocks. Finally, we note that there are also several methods [52, 53, 54] for approximating attractors of an ABN. Obviously, they however cannot guarantee finding exactly all the attractors of an ABN.

4.3 Feedback Vertex Sets and Boolean Networks

We formally state and prove the below lemmas and theorems on relations between the dynamics of a BN and its feedback vertex sets. Note that these lemmas and theorems do not depend on the updating scheme of the BN.

Lemma 4.3.1. *Let $\mathcal{N}$ be a BN whose interaction graph is acyclic. Then the STG of $\mathcal{N}$ has no cycles.*

Proof. We prove this lemma by using induction on the size n of $\mathcal{N}$.

Let G be the STG of $\mathcal{N}$. The case $n = 1$ is trivial since G has clearly only fixed points. Assume that G has no cycles with $n = k$.

We consider the case $n = k + 1$. There exists a node x_i without incoming arcs since $IG(\mathcal{N})$ is acyclic. Then, x_i is fixed, i.e., $f_i = a_i, a_i \in \mathbb{B}$. In a cycle of G, the value of x_i must not be changed since x_i is always a_i once its value receives a_i. x_i can be either 0 or 1. Let $f_1^{\mathcal{N}}, ..., f_{k+1}^{\mathcal{N}}$ be Boolean functions of $\mathcal{N}$. Then, $\mathcal{N}_1 = (V^{\mathcal{N}_1}, F^{\mathcal{N}_1})$ and $\mathcal{N}_2 = (V^{\mathcal{N}_2}, F^{\mathcal{N}_2})$ be two BNs of k nodes where $V^{\mathcal{N}_1} = V^{\mathcal{N}_2} = V^{\mathcal{N}} \backslash \{x_i\}$, $f_j^{\mathcal{N}_1} = f_j^{\mathcal{N}}(x_1, ..., x_i/a_i, x_{i+1}, ..., x_{k+1})$, $f_j^{\mathcal{N}_2} = f_j^{N}(x_1, ..., x_i/(1 - a_i), x_{i+1}, ..., x_{k+1})$ $(j \in \{1, ..., k + 1\}, j \neq i)$. Let G_1 and G_2 be the STGs of $\mathcal{N}_1$ and $\mathcal{N}_2$, respectively. Obviously, a cycle of G corresponds to a cycle of either $\mathcal{N}_1$ or $\mathcal{N}_2$. Since the interaction graphs of $\mathcal{N}_1$ and $\mathcal{N}_2$ are acyclic and have k nodes, G_1 and G_2 have no cycles by the induction hypothesis. Therefore, G has no cycles. □

Lemma 4.3.2. *Let $\mathcal{N}$ be a BN and its STG be G. Let U be an FVS of the interaction graph of $\mathcal{N}$. Then G has no cycles such that the values of the nodes in U do not change through these cycles.*

Proof. We prove this lemma by contradiction and using Lemma 4.3.1.

Let n be the number of nodes of $\mathcal{N}$. Without loss of generality, we reorder the nodes of $\mathcal{N}$ such that $U = \{x_1, ..., x_k\}$ and $V^{\mathcal{N}} \backslash U = \{x_{k+1}, ..., x_n\}$.

Assume that G has a cycle such that the values of the nodes in U do not change through this cycle (*). That is, the values of x_i $(i \in \{1, ..., k\})$ are fixed. Let $x_i = a_i, a_i \in \mathbb{B}$

$(i \in \{1, ..., k\})$. Then, $\mathcal{N}' = (V^{\mathcal{N}'}, F^{\mathcal{N}'})$ be a BN of $n-k$ nodes, where $V^{\mathcal{N}'} = V^{\mathcal{N}} \backslash U$, $f_j^{\mathcal{N}'} = f_j^{\mathcal{N}}(x_1/a_1, ..., x_k/a_k, x_{k+1}, ..., x_n)$ $(j \in \{k+1, ..., n\})$. Let G' be the STG of $\mathcal{N}'$. Obviously, G' has a cycle by (*). $IG(\mathcal{N}')$ is acyclic because U is an FVS. Hence, G' has no cycles by Lemma 4.3.1. This is a contradiction. Therefore, G has no cycles such that the values of the nodes in U do not change through these cycles. □

Theorem 4.3.1. *Let $\mathcal{N}$ be a BN and its STG be G. Let $U = \{x_{i_1}, ..., x_{i_k}\}$ be an FVS of $\mathcal{N}$. Let $B = \{b_{i_1}, ..., b_{i_k}\}$ be a set of Boolean values corresponding to the nodes of U. G' is the graph obtained by removing all arcs (x, x') from G where $\bigvee_{j=1}^{k}(x_{i_j} \leftrightarrow b_{i_j} \wedge x'_{i_j} \leftrightarrow 1 - b_{i_j})$ (*) holds. This means an arc (x, x') will be removed if it changes at least one node $x_{i_j} \in U$ from b_{i_j} to $1 - b_{i_j}$. In other words, the value of a node $x_{i_j} \in U$ in a state in G' is retained if this value is equal to b_{i_j}. With this meaning, we call B as a set of "retained" values. Then G' has no cycles.*

Proof. We prove this theorem by using Lemma 4.3.2 to show that all cycles of G disappear in G'.

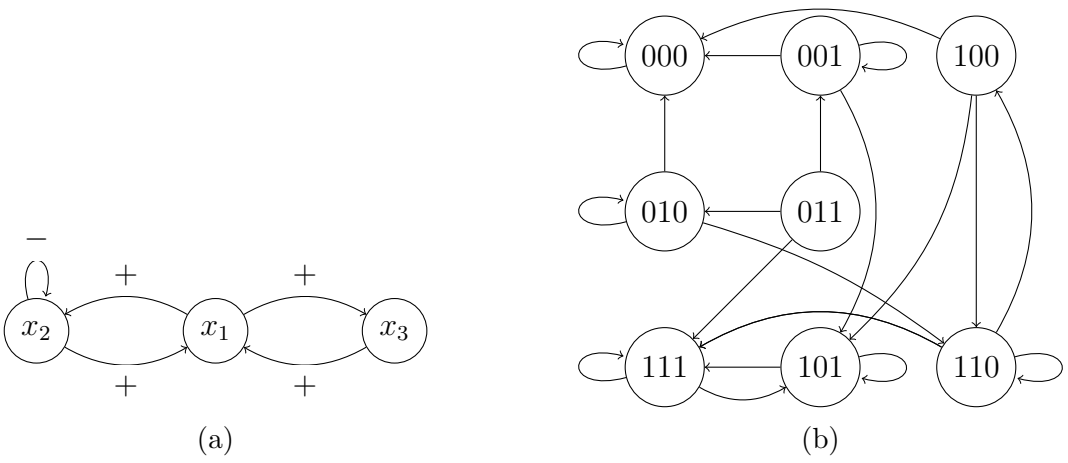
Let c be an arbitrary cycle of G. By Lemma 4.3.2, there is a node $x_i \in \{x_{i_1}, ..., x_{i_k}\}$ such that x_i changes its value through this cycle. Since c is a cycle of states, it must contain an arc (x, x') such that $x_i = b_i$ and $x'_i = 1 - b_i$. (x, x') satisfies (*) and will be removed. Then, c will disappear in G'.

Since c is arbitrary, all cycles of G will disappear in G'. Hence, G' has no cycles. In other words, G' has only fixed points. □

Example 4.3.1. *Consider a BN $\mathcal{N}$ of three nodes associated to three variables (x_1, x_2, x_3). Its Boolean functions are given by*

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

Let $\mathcal{A}$ be the ABN counterpart of $\mathcal{N}$. Figured 4.1a and 4.1 show the interaction graph of $\mathcal{N}$ and the STG of $\mathcal{A}$, respectively.



(a) (b)

Figure 4.1: (a) Interaction graph of the BN shown in Example 4.3.1. (b) STG of the ABN counterpart of the BN shown in Example 4.3.1.

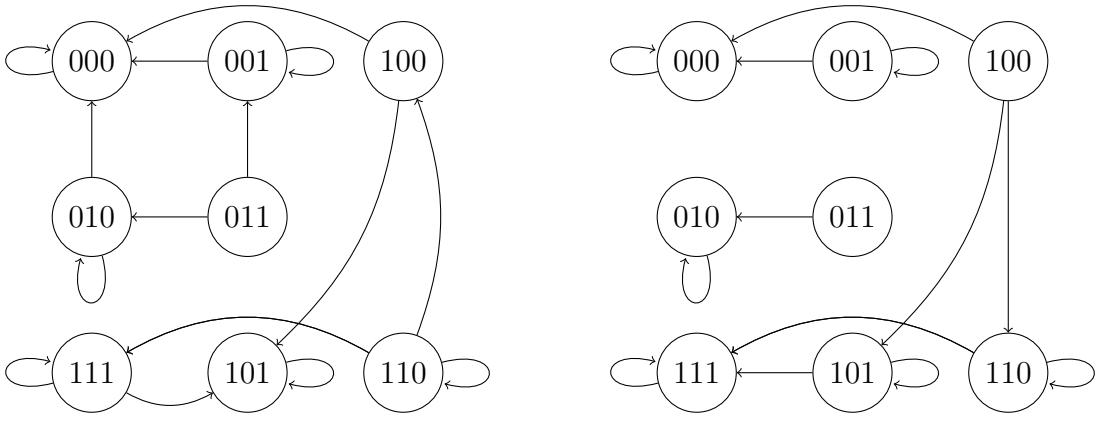


Figure 4.2: Reduced STGs of the ABN counterpart of the BN shown in Example 4.3.1 corresponding to (a) $U = \{x_1, x_2\}, b_1 = 0, b_2 = 0$ and (b) $U = \{x_1, x_2\}, b_1 = 0, b_2 = 1$.

Theorem 4.3.1 shows that the reduced STG of a BN has only fixed points. For example, let us consider the ABN $\mathcal{A}$ shown in Example 4.3.1. Suppose that U is $\{x_1, x_2\}$. If $B = \{b_1, b_2\} = \{0, 0\}$, then G' is as in Figure 4.2a. The removed arcs are $(001, 101)$, $(010, 110)$, $(011, 111)$, $(100, 110)$, and $(101, 111)$. Obviously, G' has only fixed points.

Intuitively, the removal of arcs of the STG of the BN may only increase the number of attractors; each attractor of the original STG contains at least one attractor of the reduced STG. The relations between the original STG and the reduced STG are given in Lemma 4.3.3 and Theorem 4.3.2.

Lemma 4.3.3. *Let $\mathcal{N}$ be a BN and its STG be G. G' is the graph obtained by removing an arc (x, y) from G. Let A and A' be the sets of attractors of G and G', respectively. Then, there exists a mapping $m : A \to A'$ with $m(att) \subseteq att$ for all $att \in A$ and $m(att_1) \neq m(att_2)$ for all $att_1, att_2 \in A, att_1 \neq att_2$ (*).*

Proof. We consider all two cases as follows.

Case 1: (x, y) is not in any attractor of G. Obviously, all attractors of G are still in G'. Choose the mapping m such that $m(att) = att$ for all $att \in A$.

Case 2: (x, y) is in an attractor of G. Since attractors are pairwise disjoint, (x, y) is only in one attractor of G say att. Obviously, $x \in att$ and $y \in att$. Let F be the forward reachable set of x in G' (i.e., the set of all states reachable from x). Then $F \subseteq att$. There is an attractor att' of G' such that $att' \subseteq F$. So, $att' \subseteq att$. All attractors of $A \backslash \{att\}$ of G are still in G'. Choose the mapping m such that $m(s) = s$ for $s \in A \backslash \{att\}$ and $m(att) = att'$. Moreover, let F_y be the forward reachable set of y in G'. Then $F_y \subseteq att$. There is an attractor att'_y of G' such that $att'_y \subseteq F_y$. So, we can also choose the mapping m such that $m(s) = s$ for $s \in A \backslash \{att\}$ and $m(att) = att'_y$. Since att' may be different to att'_y, the mapping m may not be uniquely determined. For example, consider the ABN of the BN in Example 4.3.1. Its STG G is as in Figure 4.1b. We have $A = \{\{000\}, \{101, 111\}\}$, $G' = G - (111, 101)$. Then, $A' = \{\{000\}, \{111\}\}$. Obviously, there is a mapping m where $m(\{000\}) = \{000\}$ and $m(\{101, 111\}) = \{111\}$.

From Case 1 and Case 2, there exists a mapping $m : A \to A'$ with $m(att) \subseteq att$ for all $att \in A$. Since attractors are pairwise disjoint, $m(att_1) \neq m(att_2)$ for all $att_1, att_2 \in A, att_1 \neq att_2$. Hence, m satisfies (*).

Therefore, we can conclude the proof. □

Theorem 4.3.2. *Let $\mathcal{N}$ be a BN and its STG be G. G' is the graph obtained by removing arcs from G. Let A and A' be the sets of attractors of G and G', respectively. Then, the exists a mapping $m : A \to A'$ with $m(att) \subseteq att$ for all $att \in A$ and $m(att_1) \neq m(att_2)$ for all $att_1, att_2 \in A, att_1 \neq att_2$ (*).*

Proof. We prove this theorem by using induction on p (the number of the removed arcs) and using Lemma 4.3.3.

Without loss of generality, we order the removed arcs as $e_1, ..., e_p$, $p \geq 1$. The base case, $p = 1$, clearly holds. Indeed, there is a mapping m satisfying (*) by Lemma 4.3.3. Assume that there is a mapping m satisfying (*) for all $p \leq k$.

The inductive case, $p = k + 1$, also holds. $G' = G - \{e_1, ..., e_{k+1}\}$. Let $G'' = G - \{e_1, ..., e_k\}$ and A'' be the set of attractors of G''. By the induction hypothesis, we have a mapping $m_1 : A \to A''$ with $m_1(att) \subseteq att$ for all $att \in A$ and $m_1(att_1) \neq m_1(att_2)$ for all $att_1, att_2 \in A, att_1 \neq att_2$. $G' = G'' - \{e_{k+1}\}$. By Lemma 4.3.3, we have a mapping $m_2 : A'' \to A'$ with $m_2(att) \subseteq att$ for all $att \in A''$ and $m_2(att_1) \neq m_2(att_2)$ for all $att_1, att_2 \in A'', att_1 \neq att_2$. Choose $m = m_2 \circ m_1$. Clearly, m satisfies (*).

Therefore, we can conclude the proof. □

Corollary 4.3.1. *Let $\mathcal{N}$ be a BN and its STG be G. G' is the graph obtained by removing arcs from G. Let A and A' be the sets of attractors of G and G', respectively. Then, $|A'| \geq |A|$.*

Proof. By Theorem 4.3.2, there is a mapping $m : A \to A'$ with $m(att) \subseteq att$ for all $att \in A$ and $m(att_1) \neq m(att_2)$ for all $att_1, att_2 \in A, att_1 \neq att_2$. Obviously, m is an injection. Hence, $|A'| \geq |A|$. □

4.4 FVS-Based Method

From the relations presented in Section 4.3, we propose an FVS-based method (called **FVS-ABN**) for finding all attractors (fixed points and cyclic attractors) of an ABN. This method includes many constituent steps. We first show the general approach of **FVS-ABN**. Then, we show each step in detail. More specifically, we formalize and analyze the problems related to each step. Then, we propose methods to efficiently solve these problems.

4.4.1 General Approach

The intuitive idea of **FVS-ABN** is as follows. Given an ABN $\mathcal{A}$, let G be the STG of $\mathcal{A}$. **FVS-ABN** systematically removes arcs from G to obtain a new acyclic STG G' (i.e., G' contains only fixed points). Let F be the set of fixed points of G'. In G, **FVS-ABN** then filters out F by using the reachability analysis on $\mathcal{A}$. The obtained result is a set of states that one-to-one corresponds to the set of attractors of $\mathcal{A}$. This set is sufficient because starting from a state s in an attractor, we can enumerate all other states of this attractors by listing all states reachable from s. We can compactly represent $FR^{\mathcal{A}}(\{s\})$ as a BDD [98] or a finite complete prefix [103]. Note that F also contains the set of fixed

points of the original STG G (say F_{fix}). Hence, **FVS-ABN** can remove F_{fix} from F before filtering the set F.

Algorithm 6 shows the description of **FVS-ABN**. Note that the result of Algorithm 6 is a set of states where each state represents (i.e., belongs to) an attractor of the ABN. Let see an example as follows. Consider the ABN of the BN shown in Example 4.3.1. Suppose that $U = \{x_1, x_2\}, b_1 = 0, b_2 = 1$. The reduced STG G' is given as in Figure 4.2b. Then, $F = \{000, 010, 111\}$ and $F_{fix} = \{000\}$. After finishing Line 8 of Algorithm 6, we obtain $F = \{010, 111\}$ and $A = \{000\}$. Suppose that $s = 010$ in the first iteration of the *while* loop (Lines 9-14 of Algorithm 6). Since 010 reaches 000 in G, 010 is not added to A. In the next iteration, $s = 111$, and it is added to A because 111 does not reach in G any state in $A \cup F = \{000\}$. Finally, $A = \{000, 111\}$, where 000 represents the attractor $\{000\}$ and 111 represents the attractor $\{111, 101\}$.

Algorithm 6 FVS-ABN

Input: An ABN $\mathcal{A}$.
Output: A set A of states of $\mathcal{A}$.

1: Find an FVS $U = \{x_{i_1}, ..., x_{i_k}\}$ of $\mathcal{A}$
2: Choose a set $B = \{b_{i_1}, ..., b_{i_k}\}$ of retained values corresponding to the nodes of U
3: Let G be the STG of $\mathcal{A}$
4: Let G' be the reduced STG with respect to U and B
5: $F \leftarrow$ the set of fixed points of G'
6: $F_{fix} \leftarrow$ the set of fixed points of G
7: $F \leftarrow F \backslash F_{fix}$
8: $A \leftarrow F_{fix}$
9: **while** $F \neq \emptyset$ **do**
10: Remove a state s from F
11: **if** s does not reach in G any state in $A \cup F$ **then**
12: $A \leftarrow A \cup \{s\}$
13: **end if**
14: **end while**
15: **return** A

Theorem 4.4.1. *Algorithm 6 finds exactly all attractors of an ABN.*

Proof. By Theorem 4.3.1, G' has no cycles. This means F is the set of all attractors of G'.

After finishing Line 8 of Algorithm 6, each attractor of G contains at least one state in $A \cup F$ by Theorem 4.3.2 (*). Through the *while* loop (Lines 9-14 of Algorithm 6), the property (*) is preserved. Indeed, in each iteration, if s reaches a state s' in $A \cup F$ and s is in an attractor *att* of G, then s' is also in *att* by the definition of an attractor.

After finishing the *while* loop, each attractor of G contains at least one state in A (a) because $F = \emptyset$. We show that A has no two states s and s' such that s and s' are in the same attractor of G (b). Indeed, if s and s' is in the same attractor of G, then s (resp. s') reaches s' (resp. s). If s is traversed before s', then s cannot be added to A. If s' is traversed before s, then s' cannot be added to A. Moreover, there is no state s in A such that s is not in any attractors (c). Indeed, if such a state s exists, it must reach at least

an attractor, implying that it reaches $A \cup F$. Hence, s cannot be added to A. This is a contradiction.

From (a), (b), and (c), we have that A one-to-one corresponds to the set of attractors of G. Therefore, Algorithm 6 finds exactly all attractors of the ABN. □

4.4.2 Computing Feedback Vertex Sets

FVS is an important concept in graph theory. The problem of finding a minimum FVS is known to be NP-hard [100]. Some approximation algorithms have been developed [137] to solve this problem. However, such algorithms are usually complicated or only applicable for undirected graphs. Hence, we here use a simple greedy algorithm for finding an (not necessarily minimum) FVS. This greedy algorithm relies on Strongly Connected Components (SCCs).

We first recall some definitions on graphs and an observation on FVSs. Let $IG(\mathcal{N}) = (V, E)$ be the interaction graph of a BN $\mathcal{N}$. Note that $IG(\mathcal{N})$ is a signed directed graph. We only consider the unsigned version of $IG(\mathcal{N})$ that is obtained by: (1) removing the sign in each edge of $IG(\mathcal{N})$; (2) merging edges that have the same starting and ending vertices into one edge. An SCC c is trivial if c is made of a single vertex v and $(v, v) \notin E$, and is non-trivial otherwise. A vertex v is a self vertex if $(v, v) \in E$. Note that, if v is a self vertex, all FVSs of $IG(\mathcal{N})$ must contain v.

Algorithm 7 shows the description of our greedy algorithm. We here use Tarjan's algorithm [138] for finding the set of SCCs. Then, the set of non-trivial SCCs is easily obtained. $IG(\mathcal{N})[c]$ denotes the subgraph of $IG(\mathcal{N})$ induced by the set of vertices c. $IG(\mathcal{N}) - V_{self}$ is equivalent to $IG(\mathcal{N})[V \backslash V_{self}]$. The outdegree of a vertex is defined as the number of edges starting from this vertex.

Algorithm 7 The **FindFVS** algorithm for finding an FVS

Input: A directed graph $IG(\mathcal{N}) = (V, E)$.
Output: An FVS U of $IG(\mathcal{N})$.

1: $U \leftarrow \emptyset$
2: $V_{self} \leftarrow$ the set of self vertices of $IG(\mathcal{N})$
3: $U \leftarrow U \cup V_{self}$
4: $IG(\mathcal{N}) \leftarrow IG(\mathcal{N}) - V_{self}$
5: $C \leftarrow$ the set of non-trivial SCCs of $IG(\mathcal{N})$
6: **while** $C \neq \emptyset$ **do**
7: Randomly pick an SCC c from C
8: Pick a vertex v with the maximum outdegree from c
9: $U \leftarrow U \cup \{v\}$
10: $C_c \leftarrow$ the set of non-trivial SCCs of $IG(\mathcal{N})[c \backslash \{v\}]$
11: $C \leftarrow C \cup C_c$
12: **end while**
13: **return** U

4.4.3 Computing Fixed Points

First, we consider the problem of computing the set of fixed points of the original STG G of the ABN (see Line 6 of Algorithm 6). F_{fix} can be easily computed by using BDDs.

Specifically, F_{fix} can be represented as a BDD characterized by the propositional formula

$$\bigwedge_{i=1}^{n}(x_i \leftrightarrow f_i(x)).$$

Consequently, we can use SAT (All-SAT) to compute F_{fix}. In addition, we can also use several other efficient methods [128, 129, 130] for solving this problem. For simplicity, we here use BDDs and SAT. These techniques are enough for our experiments.

Second, we consider the problem of computing the set of fixed points of the reduced STG G' of the ABN (see Line 5 of Algorithm 6). We state this problem as in Problem 4.4.1.

Problem 4.4.1. *Given an ABN $\mathcal{A} = (V^{\mathcal{A}}, F^{\mathcal{A}})$, an FVS U of $\mathcal{A}$, a retained set B. How can we efficiently compute the set of fixed points of the reduced STG G' with respect to U and B?*

In the ABN, a state s in G will become a fixed point in G' if and only if the updating of the node x_i does not change s_i for all $x_i \in V^{\mathcal{A}} \backslash U$ and the updating of the node x_i does not change s_i or changes s_i from b_i to $1 - b_i$ for all $x_i \in U$. Obviously, the set F of fixed points of G' can be characterized by a propositional formula. For example, consider the ABN of the BN shown in Example 4.3.1. Suppose that $U = \{x_1, x_2\}$. Then, F can be characterized by the formula

$$\begin{aligned} F^{char}_{U,B} &= \{(x'_1 \leftrightarrow x_1) \vee (x_1 \leftrightarrow b_1 \wedge x'_1 \leftrightarrow 1 - b_1)\} \\ &\wedge \{(x'_2 \leftrightarrow x_2) \vee (x_2 \leftrightarrow b_2 \wedge x'_2 \leftrightarrow 1 - b_2)\} \\ &\wedge (x'_3 \leftrightarrow x_3) \\ &\wedge (x'_1 \leftrightarrow x_2 \vee x_3) \wedge (x'_2 \leftrightarrow x_1 \wedge \neg x_2) \wedge (x'_3 \leftrightarrow x_1), \end{aligned}$$

where x and x' denote the current state and the next state, respectively. Clearly, x'_1, x'_2, x'_3 can be eliminated from $F^{char}_{U,B}$. We obtain a new formula of $F^{char}_{U,B}$ whose variables are only x_1, x_2, x_3. Finally, we have

$$\begin{aligned} F^{char}_{U,B} &= \{(x_2 \vee x_3 \leftrightarrow x_1) \vee (x_1 \leftrightarrow b_1 \wedge (x_2 \vee x_3 \leftrightarrow 1 - b_1))\} \\ &\wedge \{(x_1 \wedge \neg x_2 \leftrightarrow x_2) \vee (x_2 \leftrightarrow b_2 \wedge (x_1 \wedge \neg x_2 \leftrightarrow 1 - b_2))\} \\ &\wedge (x_1 \leftrightarrow x_3). \end{aligned}$$

We can generalize the formula $F^{char}_{U,B}$ as

$$F^{char}_{U,B} = \left\{ \bigwedge_{x_i \in U} [(f_i(x) \leftrightarrow x_i) \vee (x_i \leftrightarrow b_i \wedge f_i(x) \leftrightarrow 1 - b_i)] \right\} \wedge \left\{ \bigwedge_{x_i \notin U} [f_i(x) \leftrightarrow x_i] \right\}.$$

Since U is an FVS, the value of $x_i \notin U$ is uniquely determined from the values of the nodes of U. This means that we can obtain a formula that is equivalent to $F^{char}_{U,B}$ and contains only the variables of the nodes in U. Thus, we can claim that $|F|$ (the number of fixed points of G') is at most $2^{|U|}$. In real biological networks, the size of the minimum FVS is often much smaller than n. Hence, $|F|$ may be much smaller than the number of possible states of the ABN (i.e., 2^n) if we use a minimum FVS as U. Note that $2^{|U|}$ is only an upper bound of $|F|$, and $|F|$ depends on both U and B. Hence, using a minimum FVS may not ensure obtaining the smallest $|F|$ (see Example 4.4.1).

Example 4.4.1. *Consider the ABN counterpart of the BN shown in Example 4.3.1. The interaction graph of this ABN has three FVSs including* $\{x_1, x_2\}$, $\{x_2, x_3\}$, *and* $\{x_1, x_2, x_3\}$. *Herein,* $\{x_1, x_2\}$ *and* $\{x_2, x_3\}$ *two minimum FVSs. If we choose* $U = \{x_1, x_2\}, b_1 = 0, b_2 = 1$, *then* $|F| = |\{000, 010, 111\}| = 3$. *If we choose* $U = \{x_1, x_2, x_3\}, b_1 = 0, b_2 = 0, b_3 = 0$, *then* $|F| = |\{000, 101\}| = 2$.

The authors of [45] used an enumeration-based approach that exhausts all $2^{|U|}$ possible values of the nodes of U. However, since $F^{char}_{U,B}$ is a propositional formula, we can use some techniques, such as, BDDs or SAT (All-SAT). BDDs are often inefficient for large networks. In our method, we therefore use BDDs and All-SAT to calculate F for networks with $n \leq 60$ and $n > 60$, respectively.

Obviously, $|F|$ depends on the value of the set B. Consider the ABN of the BN in Example 4.3.1. If we choose $U = \{x_1, x_2\}, b_1 = 0, b_2 = 0$, then $|F| = |\{000, 101\}| = 2$ (see Figure 4.2a). If we choose $U = \{x_1, x_2\}, b_1 = 0, b_2 = 1$, then $|F| = |\{000, 010, 111\}| = 3$ (see Figure 4.2b). We expect that $|F|$ is as small as possible because the number of iterations of Algorithm 6 is clearly equal to $|F| - |F_{fix}|$ and $|F_{fix}|$ is fixed with respect to a given ABN. Hereafter, we consider the problem of efficiently setting the set B to obtain this expectation (see Problem 4.4.2).

Problem 4.4.2. *Given an ABN* $\mathcal{A} = (V^{\mathcal{A}}, F^{\mathcal{A}})$, *an FVS* U *of* $\mathcal{A}$, *a retained set* B. *Find a value for the retained set* B *such that the set of fixed points of the reduced STG* G' *is minimum.*

In Algorithm 6, F is the set of fixed points of the reduced STG G'. By the aforementioned analysis, F can be represented as a propositional formula $F^{char}_{U,B}$ of n Boolean variables $x_1, ..., x_n$ and m Boolean parameters $b_{i_1}, ..., b_{i_m}$, where $0 \leq m \leq n$. Problem 4.4.2 aims at finding an assignment $b^*_{i_1}, ..., b^*_{i_m}$ to $b_{i_1}, ..., b_{i_m}$ such that the number satisfying assignments of $F^{char}_{U,B}(b_{i_1}/b^*_{i_1}, ..., b_{i_m}/b^*_{i_m})$ to $x_1, ..., x_n$ (the number of fixed points) is minimum. For example, consider the ABN of the BN in Example 4.3.1. Suppose that $U = \{x_1, x_2\}$. Then, F can be characterized by the formula

$$\begin{aligned} F^{char}_{U,B} = &\{((x_2 \vee x_3) \leftrightarrow x_1) \vee (x_1 \leftrightarrow b_1 \wedge (x_2 \vee x_3) \leftrightarrow 1 - b_1)\} \\ &\wedge \{((x_1 \wedge \neg x_2) \leftrightarrow x_2) \vee (x_2 \leftrightarrow b_2 \wedge (x_1 \wedge \neg x_2) \leftrightarrow 1 - b_2)\} \\ &\wedge (x_1 \leftrightarrow x_3). \end{aligned}$$

If $(b^*_1, b^*_2) = (0, 0)$, then the number of satisfying assignments of $F^{char}_{U,B}(b_1/b^*_1, b_2/b^*_2)$ is 2, i.e., $|F| = 2$ (see Figure 4.2a). If $(b^*_1, b^*_2) = (0, 1)$, then the number of satisfying assignments of $F^{char}_{U,B}(b_1/b^*_1, b_2/b^*_2)$ is 3, i.e., $|F| = 3$ (see Figure 4.2b).

Problem 4.4.2 is related to the problem of counting the number of satisfying assignments of a propositional formula that is known as a #P-complete problem. Here, we consider the constructive version of Problem 4.4.2 (i.e., the output of Problem 4.4.2 includes the optimal assignment to $b_{i_1}, ..., b_{i_m}$ and the minimum number of satisfying assignments to $x_1, ..., x_n$). Problem 4.4.2 seems to be an optimization problem whose measure function is a #P function, thus it seems to be in Opt#P [139]. The proof would be very technical and long. However, we can see that it is too hard to solve Problem 4.4.2. Hence, we here use a heuristic method to solve it.

Let us see the characterized formula $F^{char}_{U,B}$ of F. Intuitively, we need to assign the most evaluation of f_{i_k} $(k \in \{1, ..., m\})$ to b_{i_k}. $a \in \mathbb{B}$ is the most evaluation of a function f if the

number of satisfying assignments on f's inputs of $f \leftrightarrow a$ is greater than or equal to the number of satisfying assignments on f's inputs of $f \leftrightarrow 1-a$. Assigning the most evaluation of f_{i_k} to b_{i_k} makes the number of assignments to $x_1, ..., x_n$ of $(x_{i_k} \leftrightarrow b_{i_k} \wedge f_{i_k}(x) \leftrightarrow 1 - b_{i_k})$ decreased. Since the number of assignments to $x_1, ..., x_n$ of $x_{i_k} \leftrightarrow f_{i_k}(x)$ does not depend on b_{i_k}, this rule can reduce the number of assignments to $x_1, ..., x_n$ of $\{x_{i_k} \leftrightarrow f_{i_k}(x) \wedge (x_{i_k} \leftrightarrow b_{i_k} \wedge f_{i_k}(x) \leftrightarrow 1 - b_{i_k})\}$. For example, the most evaluation of $f_1 = x_2 \vee x_3$ is 1 (from the truth table, 1 (75%) and 0 (25%)), thus we assign 1 to b_1. Similarly, b_2 is assigned to 0. Now, $|F|$ is 2. The number of attractors of the ABN is 2, we thus obtain the optimal value of $|F|$ because the number of attractor of the ABN is a lower bound of $|F|$ by Theorem 4.3.2. Note that this idea follows a greedy manner, thus the optimality of the solution is not guaranteed (see Example 4.4.2).

Example 4.4.2. *Consider an ABN of three nodes (x_1, x_2, and x_3). Its Boolean functions are given by $f_1 = (\neg x_1 \vee \neg x_3) \wedge x_2$, $f_2 = \neg x_1 \vee \neg x_2$, $f_3 = \neg x_2$. Figures 4.3a and 4.3b denote the interaction graph and the STG of the ABN, respectively. Suppose that $U = \{x_1, x_2\}$. By the heuristic presented in the previous paragraph, we have $b_1 = 0, b_2 = 1$ and $|F| = |\{010, 110\}| = 2$. However, if we choose $b_1 = 1, b_2 = 1$, we will obtain $|F| = |\{110\}| = 1$.*

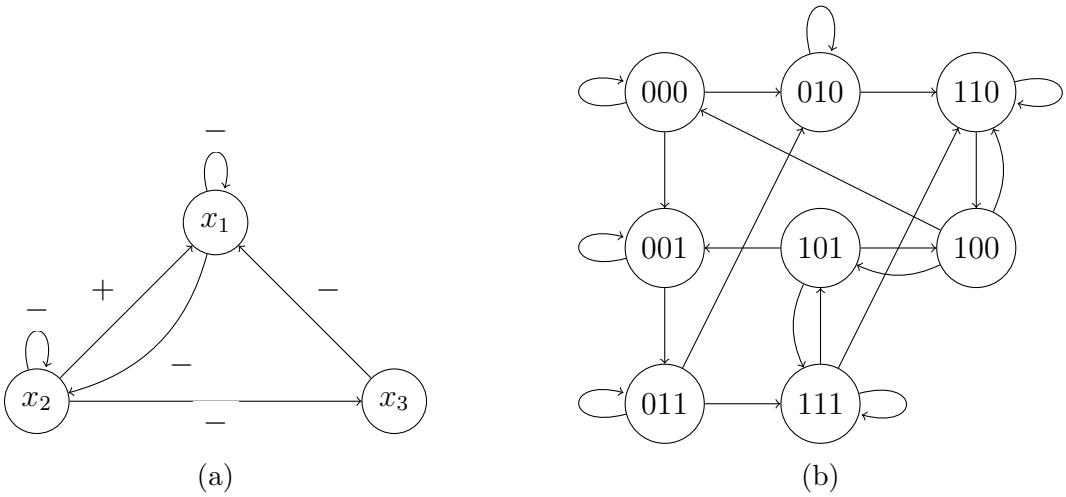


Figure 4.3: An example ABN with its interaction graph (a) and its STG (b).

There is a small problem: How to efficiently calculate the most evaluation of f_{i_k}? Constructing a truth table of f_{i_k} is a direct and simple way. However, when f_{i_k} has many input nodes, this way is inefficient because the number of rows of the truth table of f_{i_k} is $2^{|IN(f_{i_k})|}$. Since f_{i_k} can be encoded as a BDD, we can use the built-in **SATCount** function in BDDs. The time complexity of **SATCount** is $\mathcal{O}(|IN(f_{i_k})|)$ [109]. In summary, we here propose an algorithm for setting an assignment to B (see Algorithm 8). In this algorithm, **SATCount**(f_i) returns the number of satisfying assignments of f_i. c_{true} and c_{false} denote the numbers of assignments in which the values of f_i are 1 and 0, respectively. In the *for* loop, when $c_{true} = c_{false}$, we randomly assign either 1 or 0 to b_i.

4.4.4 Preprocessing

As aforementioned in the previous subsection, the number of iterations of Algorithm 6 is equal to $|F|$. Note that all fixed points of the ABN have been excluded from F (see

Algorithm 8 Algorithm for setting an assignment to B

Input: $B = \{b_{i_1}, ..., b_{i_m}\}$.
Output: An assignment $(b^*_{i_1}, ..., b^*_{i_m})$ to $b_{i_1}, ..., b_{i_m}$.

```
for all i ∈ {i_1, ..., i_m} do
    c_true ← SATCount(f_i)
    c_false ← 2^|IN(f_i)| − c_true
    if c_true > c_false then
        b*_i ← 1
    else if c_true < c_false then
        b*_i ← 0
    else
        b*_i ← randomly either 1 or 0
    end if
end for
return (b*_{i_1}, ..., b*_{i_m})
```

Line 7 of Algorithm 6). It is better if we can reduce the size of F but still retain the correctness of **FVS-ABN**. With this motivation, we here propose a preprocessing that aims at shrinking the set F. We name this preprocessing as **Preprocessing-SSF** (Shrinking the Set F). In each iteration of **Preprocessing-SSF**, we choose randomly a node x_i to be updated. Then, $F \leftarrow FI^{\mathcal{A}}_{x_i}(F)$, where $FI^{\mathcal{A}}_{x_i}(F)$ is the forward image set of F of the ABN $\mathcal{A}$ by updating node x_i (see Section 2.1). Note that $FI^{\mathcal{A}}_{x_i}(F)$ can be easily computed by using BDDs (see [43]). The number of iterations can be empirically obtained. We shall discuss this issue in more detail at the end of this subsection. Hereafter, we show that **Preprocessing-SSF** preserves the correctness of Algorithm 6 by three following properties of ABNs (see Propositions 4.4.1, 4.4.2, and 4.4.3). We omit the proofs of these properties because they are obvious.

Proposition 4.4.1. *Let F' be the forward image set of F by updating node x_i. Then, $|F'| \leq |F|$. In addition, two states of F may have the same next state by updating node x_i. In this case, we have $|F'| < |F|$.*

Proposition 4.4.2. *If state s belongs to an attractor of the ABN, then s' also belongs to this attractor, where s' is an arbitrary state reachable from s in the STG of the ABN.*

Proposition 4.4.3. *Let s be a state of the ABN $\mathcal{A}$. Then, $FR^{\mathcal{A}}(\{s'\}) \subseteq FR^{\mathcal{A}}(\{s\})$, where s' is a next state of s in the STG of $\mathcal{A}$.*

Proposition 4.4.1 guarantees that the cardinilaty of F does not increase through these iterations even may decrease. Proposition 4.4.2 guarantees that all cyclic attractors of the ABN still appear in the set F, thus all attractors of the ABN still appear in $A \cup F$. Proposition 4.4.3 justifies the usefulness of **Preprocessing-SSF** for the next steps of **FVS-ABN** because the forward reachable set of each state in F may be shrunk. Note that after finishing iterative procedure, F may contain some fixed points of the ABN. Hence, we need to again exclude the fixed points of the ABN from F, i.e., to again perform Line 7 of Algorithm 6. Now, all attractors of the ABN still appear in $A \cup F$. Consequently, the correctness of **FVS-ABN** is retained.

To illustrate **Preprocessing-SSF**, we continue with the ABN $\mathcal{A}$ of the BN shown in Example 4.3.1. Assume that $U = \{x_1, x_2\}, b_1 = 0, b_2 = 1$. Then, $F = \{000, 010, 111\}$ (see Figure 4.2b). After finishing Lines 7 and 8 of Algorithm 6, we have $F = \{010, 111\}$ and $A = \{000\}$. Assume that the number of iterations of **Preprocessing-SSF** is 2. Let $I = \langle x_{i_1}, x_{i_2} \rangle$ denote the sequence of the updated nodes. If $I = \langle x_1, x_2 \rangle$, then $F = \{010, 111\} \rightarrow F = \{110, 111\} \rightarrow F = \{110, 101\}$ (see Figure 4.1b). Obviously, the cardinilaty of F does not increase and all the attractors ($\{000\}$ and $\{101, 111\}$) still appear in $A \cup F$ through these iterations. We have $FR^{\mathcal{A}}(\{110\}) = \{110, 100, 000, 101, 111\}$ and $FR(\{010\}) = \{010, 110, 100, 000, 101, 111\}$. Proposition 4.4.3 holds because $FR^{\mathcal{A}}(\{110\}) \subseteq FR^{\mathcal{A}}(\{010\})$. If we choose $I = \langle x_1, x_3 \rangle$, then $F = \{010, 111\} \rightarrow F = \{110, 111\} \rightarrow F = \{111\}$ (see Figure 4.1b). In this case, the cardinilaty of F even decreases. In addition, if we choose $I = \langle x_2, x_3 \rangle$, then $F = \{010, 111\} \rightarrow F = \{000, 111\} \rightarrow F = \{000, 111\}$ (see Figure 4.1b). The cardinilaty of F is unchanged. However, after excluding the fixed points from F again, we have $F = \{111\}$, i.e., the cardinilaty of F decreases.

Finally, we discuss how to set the number of iterations of **Preprocessing-SSF** (say I_{max}). Obviously, we expect that I_{max} is good enough, i.e., I_{max} is not too large and the cardinilaty of F after **Preprocessing-SSF** is as small as possible. **Preprocessing-SSF** is optimal when the cardinilaty of F after **Preprocessing-SSF** is equal to the number of cyclic attractors of the ABN. We set I_{max} based on the following intuitions. First, if n increases (resp. decreases) then I_{max} should increase (resp. decrease). Second, if $|F|$ increases (resp. decreases) then I_{max} should increase (resp. decrease). Third, if F_{fix} increases (resp. decreases) then I_{max} should decreases (resp. increases). Last, I_{max} however should not exceed a threshold. Combining with running some sample networks, we empirically set I_{max} as

$$I_{max} = min(2 \times n^{1.5} \times |F|/(1 + |F_{fix}|), 5000),$$

where we use $1 + |F_{fix}|$ to deal with the case $|F_{fix}| = 0$.

4.4.5 Reachability Analysis

Reachability is a central problem in systems science. It is also the key task in our method. In theory, the reachability in ABNs has been proved PSPACE-complete [105]. Thus, it is difficult to solve this problem. However, in practice, there are various methods for checking the reachability in ABNs. Since the set of states reachable from the starting state may be very large, the explicitly depth-first search and breadth-first search manners are inefficient. We need a more efficient approach. The use of BDDs on a breadth-first search-based method is a better solution. However, it still meets the inherent problems of BDDs [110] (e.g., extremely long computational time, OOM). SAT-based BMC [110] is an efficient approach. However, it is incomplete unless we use a completeness threshold that is usually very hard to compute even for the case of SBNs [44]. Note that an ABN has high concurrency [105]. Hence, we can use an unfolding-based approach that is known to be an efficient approach that exploits the concurrency of asynchronous systems [106]. Recently, some efficient approximation methods (e.g., Pint [140], ASPReach [141]) have been proposed for checking the reachability in ABNs. The authors reported that these methods can handle large-scale networks. However, they are of course incomplete. Moreover, in the experiments reported in [140, 141], the start and target states only cover a small set of nodes. We here aim at proposing an exact and efficient method for finding

all attractors of an ABN. Therefore, we focus on unfolding-based methods for checking the reachability in ABNs.

The method for encoding an ABN as a 1-safe PN has been proposed [103]. Thus, we can apply the algorithms [106] for checking the reachability in 1-safe PNs to those in ABNs. Hereafter, we show how Line 11 of Algorithm 6 is performed. Obviously, we can perform it by calling **UnfReach**$(\mathcal{A}, s, A \cup F)$. The description of the function **UnfReach** is shown in Algorithm 9. However, there are two problems needed to be considered. First, in the case that the reachability holds, we do not need to build the whole finite complete prefix $\mathcal{P}_\mathcal{U}$. Second, M_F is a set of markings. It is too waste when calling **Mole** $|M_F|$ times due to the standard function of **Mole** is to check only whether a given 1-safe PN reaches a desirable marking.

To deal with these problems, we here propose a new method (called **OnTheFlyMole**) based on an on-the-fly manner. Note that the authors of [103] have also used Petri net unfoldings for checking the reachability in ABNs. However, they have not shown in detail their method. Our method may be similar to the method by [103]. We first add new transitions to $\mathcal{P}$. A new transition corresponds to a marking in M_F. For each transition $t_m (m \in M_F)$, we add new arcs from all places in m to t_m. This means that when reaching the marking m, t_m is enabled. Then, in the process of building the finite complete prefix $\mathcal{P}_\mathcal{U}$, whenever at least one new transition is enabled (i.e., this transition will appear in $\mathcal{P}_\mathcal{U}$) we terminate the process and return true (i.e., reachable). When finishing the process, we return false (i.e., unreachable). **Mole** also supports an on-the-fly manner. However, it only deals with the case of one transition. Therefore, we have adjusted a little the source code of **Mole** to implement our method.

Back to the **UnfReach** function, its special case is when $F = \emptyset$. For this case, we simply return false (i.e., unreachable). When $F \neq \emptyset$, we set the corresponding marking of s in $\mathcal{P}$ (denoted by $[[s]]_\mathcal{P}$) as the initial marking of $\mathcal{P}$ and then simply call **OnTheFlyMole**$(\mathcal{P}, M_F)$, where M_F is the set of markings of $\mathcal{P}$ corresponding to the states in F of $\mathcal{A}$.

Algorithm 9 UnfReach

Input: An ABN $\mathcal{A}$, a state s, a set F of states.
Output: Whether s reaches F in the STG of $\mathcal{A}$?

1: $F \leftarrow$ **Preprocessing-BCN**(s, F)
2: **if** $F = \emptyset$ **then**
3: **return** false
4: **else**
5: $\mathcal{P} \leftarrow$ the encoded 1-safe PN of $\mathcal{A}$
6: $M_0 \leftarrow [[s]]_\mathcal{P}$
7: Set M_0 as the initial marking of $\mathcal{P}$
8: $M_F \leftarrow \{[[x]]_\mathcal{P} | x \in F\}$
9: **return** **OnTheFlyMole**$(\mathcal{P}, M_F)$
10: **end if**

The size of the built prefix of an encoded 1-safe PN may be very large. In this case, the computation of finite prefixes may be intractable. To mitigate this issue, we first use a preprocessing step in the **UnfReach** function. This preprocessing aims at checking the reachability in an ABN without building its prefixes or at least reducing the number

of target states (i.e., the states of F). The optimal case is when $F = \emptyset$, implying that s does not reach F in the STG of $\mathcal{A}$. Hereafter, we only show the description of this preprocessing, its usefulness shall be discussed in Subsection 4.5.1.

We first define a special type of nodes in a BN as follows. A node $x_i, i \in \{1, ..., n\}$ is called a *zero-constant* or *one-constant* node if $f_i \wedge \neg x_i = 0$ or $f_i \vee \neg x_i = 1$, respectively. In the biological context, a constant node will retain its value once it is set to a specific value (0 or 1). For example, if $f_i = 0$ or $f_i = x_i \wedge ...$, then x_i is called a zero-constant node. On the other hand, if $f_i = 1$ or $f_i = x_i \vee ...$, then x_i is called a one-constant node. The proposed preprocessing is based on the properties of zero-constant and one-constant nodes. We name this preprocessing as **Preprocessing-BCN** (Based on Constant Node). Let V^0 and V^1 be the sets of zero-constant nodes and one-constant nodes of $\mathcal{A}$, respectively. For each node $x_i \in V^0$, if $s_i = 0$ we can remove from F the state s' such that $s'_i = 1$. For each node $x_i \in V^1$, if $s_i = 1$ we can remove from F the state s' such that $s'_i = 0$. Now, we obtain a new set F' satisfying s reaches F in the STG of $\mathcal{A}$ if and only if s reaches F' in the STG of $\mathcal{A}$.

4.5 Experiments

We have implemented the proposed method for finding ABN attractors in a JAVA tool with the same name **FVS-ABN**. **FVS-ABN** uses the JDD library [112] for BDD manipulation and Z3 [118] as the SAT solver. An executable file of **FVS-ABN** and some examples of real biological networks are available at https://sites.google.com/site/trinhgiangjaist/fvs-abn. To evaluate the efficiency of **Preprocessing-SSF**, we conducted experiments to compare the performance of two variants of **FVS-ABN**. The first variant (say $\mathcal{M}_1$) does not use **Preprocessing-SSF**, whereas the second variant (say $\mathcal{M}_2$) uses **Preprocessing-SSF**. We also compare the performance among $\mathcal{M}_1$, $\mathcal{M}_2$, **genYsis** [43], and **CABEAN** [49]. We choose **genYsis** and **CABEAN** because they are exact and famous tools for finding attractors of an ABN.

All the experiments were run on a virtual machine whose environment is CPU: Intel(R) Xeon(R) Silver 4116 4x2.10GHz, Memory: 24 GB, CentOS 7 64 bit. We used two sets of Boolean networks. The first set includes BN models of real biological networks obtained from the literature. The second set includes BNs random generated with **Bool Net R** package [117]. Note that **FVS-ABN** uses BDDs in **Preprocessing-SSF** and both **genYsis** and **CABEAN** are BDD-based methods. Since high memory consumption is an inherent problem of BDDs, memory needs to be considered. In our experiments, we set the heap size to 16 GB. With this heap size, all these methods never met OOM before exceeding the time limit in all networks.

4.5.1 Experimental Results on Real Biological Networks

We applied the four algorithms to BNs of 32 real biological networks whose sizes range from 19 to 101. A BN can have some input nodes x_i that do not change their values through the evolution of the BN (i.e., $f_i = x_i$). We here consider the networks where the value of an input node is not fixed to either 0 or 1. **CABEAN** requires to use a network reduction technique that removes all the *leaf nodes* [47] of the BN. This reduction technique fully conserves the attractors of an ABN [122]. To ensure the fairness of the experiments, we also used this reduction technique for both **FVS-ABN** and **genYsis**.

Last, the time limit for each network was set to 10 hours because the running time may be very long for large-scale networks.

Table 4.1 shows the experimental results. Columns 1, 2, 3, and 4 denote the name of the network, the number of nodes (n), the size of the used FVS ($|U|$), and the number of attractors ($|A|$), respectively. Column "$|F|$" denotes the size of the filtering set F before using **Preprocessing-SSF**, whereas Column "$|F_1|$" denotes that after using **Preprocessing-SSF**. Column "time (s)" denotes the running time in seconds. "-" stands for the case of not obtaining the result within the time limit. We observed that in some BNs, **CABEAN** terminated before exceeding the time limit and the Segmentation fault error was printed. We guessed that in the computation of attractors, **CABEAN** will terminate when meeting a criterion (e.g., the size of the computed attractor exceeds a threshold). Anyway, **CABEAN** did not finish the computation of attractors in this case. For these BNs, we reported the time when **CABEAN** terminated and used "*" to indicate the case. From these results, we obtain five remarks as follows.

First, the size of the FVS obtained by the **FindFVS** algorithm (see Algorithm 7) is much smaller than the network size in all the networks (especially in, e.g., the HGF_Signaling_in_Keratinocytes network, the PC12CellDifferentiation network, the T_Cell_Receptor_Signaling network). This observation confirms that **FindFVS** is good enough for finding a minimum or near minimum FVS.

Second, $|F_1|$ is smaller than $|F|$ in 29/32 networks. In addition, $\mathcal{M}_2$ outperforms $\mathcal{M}_1$ in most networks, especially in, e.g., the FA_BRCA_pathway network, the Bcell network, and the MAPK network. Note that in some networks (e.g., the Differentiation_of_T_lymphocyte network, the YeastApoptosis network, the IL_6_Signalling network), $\mathcal{M}_2$ is much slower than $\mathcal{M}_1$. This is apparent because we must take time for **Preprocessing-SSF**. However, the difference between running time of $\mathcal{M}_2$ and $\mathcal{M}_1$ is insignificant or the running time of $\mathcal{M}_2$ is reasonable (< 11 mins) in these networks. These observations are evidence for the usefulness of **Preprocessing-SSF**.

Third, $\mathcal{M}_2$ outperforms **genYsis** in most networks. In 8/32 networks, **genYsis** failed to obtain the result within the time limit, whereas $\mathcal{M}_2$ succeeded. In 1/32 network (the YeastApoptosis network), $\mathcal{M}_2$ is slower than **genYsis**. However, the difference between running time of $\mathcal{M}_2$ and **genYsis** is insignificant. In most of the 23/32 remaining networks, $\mathcal{M}_2$ is much faster than **genYsis**. Especially, the difference between the running time of $\mathcal{M}_2$ and **genYsis** is very large in some of these networks, such as, the Bcell network (22.59 and 8702.80), the HumanMyelomaCells network (47.00 and 12983.39), the TcellLGL network (55.56 and 21198.63). Furthermore, $\mathcal{M}_1$ even completely outperforms **genYsis** in some networks (e.g., the Colitis_associated_colon_cancer network, the Differentiation_of_T_lymphocytes network, the IL_6_Signalling network). These observations show the effectiveness of the FVS-based method as compared to **genYsis**.

Fourth, $\mathcal{M}_2$ also outperforms **CABEAN** in most networks. In 10/32 networks, **CABEAN** failed to finish the computation of attractors, whereas $\mathcal{M}_2$ succeeded within the time limit. Even the running time of **CABEAN** before terminating is greater (e.g., the T_Cell_Receptor_Signaling network) or much greater (e.g., the InflammatoryBowelDisease network, the TLGLSurvival network, the Colitis_associated_colon_cancer network) than the running time of $\mathcal{M}_2$. In 9/32 networks, $\mathcal{M}_2$ is slower than **CABEAN**. However, the difference between the running time of $\mathcal{M}_2$ and **CABEAN** is insignificant (e.g., the AuroraKinaseA network, the GuardCellAbscisicAcidSignaling network) or the running time of $\mathcal{M}_2$ is reasonable (e.g., the Differentiation_of_T_lymphocytes network, the Yeast-

Apoptosis network). In most of the 13/32 remaining networks, $\mathcal{M}_2$ is much faster than **CABEAN**. Especially, the difference between the running time of $\mathcal{M}_2$ and **CABEAN** is very large in some of these networks, such as, the TcellLGL network (55.56 and 916.23) and the Drosophila network (4.88 and 1984.40). Furthermore, $\mathcal{M}_1$ even completely outperforms **CABEAN** in some networks (e.g., the Colitis_associated_colon_cancer network, the Differentiation_of_T_lymphocytes network, the IL_6_Signalling network, the T_Cell_Receptor_Signaling network). These observations show the effectiveness of the FVS-based method as compared to **CABEAN**.

Last, $\mathcal{M}_1$ and $\mathcal{M}_2$ even can handle large networks in terms of attractor detection in ABNs without using any network reduction technique. We here report the running time of $\mathcal{M}_1$ and $\mathcal{M}_2$ for some large networks without using any network reduction technique. For the IL_6_Signalling network ($n = 86$), the running time of $\mathcal{M}_1$ and $\mathcal{M}_2$ is 1323.38 seconds and 368.56 seconds, respectively. For the T_Cell_Receptor_Signaling network ($n = 101$), the running time of $\mathcal{M}_1$ and $\mathcal{M}_2$ is 583.71 seconds and 1463.30 seconds, respectively. Note that **genYsis** failed to obtain the result within the time limit in all the two networks. The analysis for these networks was usually performed by using their reduced versions (e.g., removing leaf nodes or fixing input nodes) because of the performance limitations of the existing tools. Therefore, the advanced computation capability of our method can enable biologists to conduct more accurate analysis on large networks.

In this end of this subsection, we shall discuss the usefulness of **Preprocessing-BCN** presented in Subsection 4.4.5 as well as the impact of the picked FVS to the performance of our method (i.e., $\mathcal{M}_2$).

By applying $\mathcal{M}_2$ without using **Preprocessing-BCN** on some real networks of the benchmark, we observed that **Preprocessing-BCN** can accelerate the running time of our method. In the remy_tumorigenesis network, the speedup by using **Preprocessing-BCN** is $9.13/3.35 = 2.73$. Especially, in the IL_6_Signalling and T_Cell_Receptor_Signaling networks, $\mathcal{M}_2$ without using **Preprocessing-BCN** did not find all the attractors within 10 hours, whereas $\mathcal{M}_2$ with using **Preprocessing-BCN** found all the attractors in 297.51 seconds and 5.27 seconds, respectively (see Table 4.1).

By applying $\mathcal{M}_2$ with randomly choosing an FVS on some real networks of the benchmark, we observed that the picked FVS may largely impact the performance of our method. In the IL_6_Signalling network, we obtained the result as $|U| = 25$ and the running time is 3337.85 seconds. The result by using Algorithm 7 for choosing an FVS is $|U| = 21$ and the running time is 297.51 seconds (see Table 4.1). Especially, in the ButanolProduction network, we obtained the result as $|U| = 30$ and $\mathcal{M}_2$ did not find all attractors within 10 hours, whereas the result by using Algorithm 7 for choosing an FVS is $|U| = 18$ and the running time is 324.22 seconds (see Table 4.1). We have shown in Subsection 4.4.3 that having the minimum FVS may not ensure having the best performance. However, using a minimum or nearly minimum FVS is still a good strategy to obtain good performance at least in our benchmarks.

4.5.2 Experimental Results on Randomly Generated Networks

We randomly generated a set of N-K BNs [22] with network size n in the set {50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105, 110} and $K = 2$ (i.e., each node has exactly $K = 2$ input nodes). For each network size, 20 instances were generated. In total, we have 260 random BNs.

Table 4.1: Experimental results of $\mathcal{M}_1$, $\mathcal{M}_2$, **genYsis**, and **CABEAN** on real biological networks.

				$\mathcal{M}_1$		$\mathcal{M}_2$			**genYsis**	**CABEAN**
network name	n	$\|U\|$	$\|A\|$	$\|F\|$	time (s)	$\|F\|$	$\|F_1\|$	time (s)	time (s)	time (s)
OxidativeStressPathway [119]	19	4	2	4	0.71	4	1	0.20	1.4	0.73
AuroraKinaseA [119]	23	8	32	32	0.61	32	16	0.73	2.75	0.49
dahlhaus_neuroplastoma [124]	23	8	32	32	0.64	32	16	0.65	2.73	0.54
FA_BRCA_pathway [119]	28	11	1	13	-	13	2	0.73	10.02	4.19
Septation_Initiation_Network [119]	31	10	640	320	1.43	320	0	1.12	51.92	1.22
TumourCell [119]	32	13	9	117	1.75	117	0	0.60	2.33	0.51
Bordetella_bronchiseptica [119]	33	9	3	18	61.85	18	0	0.55	2.03	1.73
Lymphoid_myeloid_cell_specification [119]	33	8	21	35	1.75	35	0	0.59	21.69	3.51
CholesterolRegulatoryPathway [119]	34	3	4	2	0.44	2	0	0.25	4.85	0.49
remy_tumorigenesis [126]	35	16	25	892	72.08	892	5	3.35	15.91	2.49*
TCellSignaling [119]	40	6	8	5	0.46	5	1	0.16	0.22	0.04
ApoptosisNetwork [119]	41	7	8	8	6.08	12	8	7.27	581.07	6.62*
Treatment_of_Castration_Resistant [119]	42	14	16384	0	0.49	0	0	0.13	18.18	0.73
GuardCellAbscisicAcidSignaling [119]	44	8	28	20	0.61	32	15	1.33	7.90	0.83
InflammatoryBowelDisease [119]	47	22	1	960	-	960	1	2.47	-	12.77*
Stomatal_Opening_Model [119]	49	13	48	390	9.48	243	14	10.99	31.22	2.38
Differentiation_of_T_lymphocytes [119]	50	18	2050	5581	80.37	5581	0	627.76	-	89.75
Senescence [119]	51	12	17	84	13.69	84	2	9.93	18.05	3.00
Drosophila [125]	52	14	128	84	1.46	84	0	4.88	-	1984.40
MAPK [119]	53	10	18	102	-	226	6	8.15	-	5.54*
B_bronchiseptica_T_retortaeformis [119]	53	15	30	298	7794.31	298	0	15.61	3556.85	440.16
TcellLGL [119]	60	23	142	11156	-	11156	108	55.56	21198.63	916.23
TLGLSurvival [119]	61	25	318	18276	-	18276	260	174.66	-	1479.23*
PC12CellDifferentiation [119]	62	3	3	0	0.39	0	0	0.20	5.01	0.59
ButanolProduction [119]	66	18	8192	12416	-	12416	6144	324.22	-	24.80*
HumanMyelomaCells [119]	67	14	83	558	1305.75	558	0	47.00	12983.39	0.02*
HGF_Signaling_in_Keratinocytes [119]	68	10	72	256	5.3	256	0	3.79	1200.04	8.75
Colitis_associated_colon_cancer [119]	70	13	10	84	516.06	100	14	391.05	-	12614.96*
Bcell [123]	72	19	72	934	12912.62	934	69	22.59	8702.80	29.84
YeastApoptosis [119]	73	17	8448	4352	3.08	4352	4352	75.32	45.85	1.16
IL_6_Signalling [119]	86	21	32768	20480	104.14	20480	4096	297.51	-	11.71*
T_Cell_Receptor_Signaling [119]	101	10	128	72	2.89	72	24	5.27	3596.65	6.35*

We then applied $\mathcal{M}_2$, **genYsis**, and **CABEAN** to the 260 random BNs and recorded the number of failures (i.e., failed to obtain the result within 30 minutes). Since the usefulness of **Preprocessing-SSF** has been justified in the previous subsection, we did not apply $\mathcal{M}_1$ to these BNs. Note that we also used the network reduction technique as in Subsection 4.5.1. The results are shown in Figure 4.4. As we can see, the number of failures of **genYsis** or **CABEAN** rapidly approaches 20. On the other hand, $\mathcal{M}_2$ can even handle 30 or 55 percent of networks for $n = 105$ or $n = 110$, respectively. In addition, in each network size, the number of failures of **genYsis** or **CABEAN** is always larger than that of $\mathcal{M}_2$. These observations show that $\mathcal{M}_2$ is more scalable than both **genYsis** and **CABEAN** in terms of N-K BNs.

For $n = 110$, $\mathcal{M}_2$ failed to obtain the result within 30 minutes in 45 percent of networks. Note that these BNs have been reduced by using the network reduction technique based on leaf nodes. In the biological context, 110-node networks are not very large because comprehensive analysis of gene regulatory networks often requires formal models including hundreds or even thousands of elements [49]. Thus, $\mathcal{M}_2$ (as also other previous methods) is generally incapable of handling very large BNs (e.g., 500-node or 1000-node BNs). Improving $\mathcal{M}_2$ to handle such BNs is one of our future work.

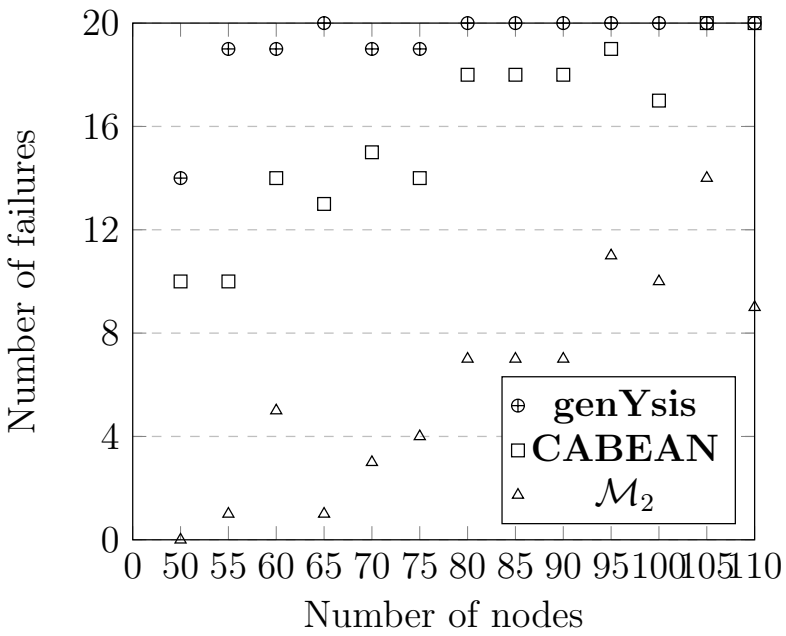


Figure 4.4: Experimental results of $\mathcal{M}_2$, **genYsis**, and **CABEAN** on randomly generated networks.

4.6 Improvements

As presented in Section 4.5, **FVS-ABN** outperforms the two state-of-the-art methods, **genYsis** [43] and **CABEAN** [49]. In particular, **FVS-ABN** can handle real biological networks with up to 101 nodes without using any network reduction technique, whereas the other methods cannot. In the biological context, 101-node networks are not very large because the comprehensive analysis of biological networks often requires formal models

that possess hundreds or even thousands of elements [49]. Moreover, **FVS-ABN** tends to not work well in the case that the used FVS is large (e.g., having more than 25 nodes). Hence, **FVS-ABN** needs to be improved to handle networks with larger n or FVSs.

In this section, we propose an improved method (called **iFVS-ABN**) that includes two improvements to **FVS-ABN**. The first improvement is a new method (called **ABNReach**) for checking the reachability in ABNs. We present the details of **ABNReach** in Subsection 4.6.1. The second improvement is to use an NFVS instead of an FVS to get the candidate set of states. This use relies on a relation between an NFVS of the interaction graph of an ABN and the dynamics of the ABN. We present the relation and the use of NFVSs in Subsection 4.6.2. Specifically, **iFVS-ABN** uses **ABNReach** instead of **UnfReach** at Line 11 of Algorithm 6 and finds an NFVS U^- instead of an FVS U at Line 1 of Algorithm 6, respectively. Other parts of **iFVS-ABN** are the same as those in **FVS-ABN**. Finally, we show the correctness of **iFVS-ABN** in Subsection 4.6.3.

4.6.1 Improvement in Reachability Analysis

Checking the reachability in ABNs is the key task in **FVS-ABN**. In the previous experiments (see Section 4.5), most of the running time was spent for checking the reachability in ABNs. Even in some networks in which **FVS-ABN** failed to obtain the result within the time limit, the main reason for the failure is that the time for checking the reachability in ABNs is extremely long. Hence, it is needed to improve the **UnfReach** algorithm.

From the previous discussions (see Subsection 4.4.5), we can see that each previous technique for checking the reachability in ABNs has its advantages and disadvantages. Hence, combining multiple techniques may be potential. We here propose a new method (named **ABNReach**) to check the reachability in ABNs. In general, **ABNReach** is a reasonable combination of multiple previous techniques for checking the reachability in ABNs. Algorithm 10 shows the description of **ABNReach**. In Algorithm 10, true indicates reachable, whereas false indicates unreachable. **ABNReach** includes three dependent phases as follows.

ABNReach starts its first phase by using **PintReach** (see Lines 1-8 of Algorithm 10). **PintReach** is the method by [140] for checking the reachability in Asynchronous Automata Networks (AANs). We use **PintReach** in the first phase because this method was reported able to check the reachability in very large networks with reasonable time. The inputs of **PintReach** include an AAN, an initial state, and a set of target states. Note that **PintReach** is only an approximation method; its output is one of three cases: true (i.e., reachable), false (i.e., unreachable), inconclusive (i.e., cannot determine to be either reachable or unreachable). If the output of **PintReach**$(\mathcal{Q}, s^{\mathcal{Q}}, F^{\mathcal{Q}})$ is either true or false, **ABNReach** simply returns this output (see Lines 4-7 of Algorithm 10). Otherwise, the output of **PintReach** is inconclusive, and **ABNReach** must start its second phase.

When the output of **PintReach** is inconclusive, the result of the reachability analysis tends to be reachable in most cases [141]. Hence, using SAT-based BMC with a low bound is a good preprocessing step, since SAT-based BMC is efficient for large systems in the case of using a low bound [110]. In the second phase, **ABNReach** therefore uses SAT-based BMC to check the reachability in ABNs (see Lines 9-14 of Algorithm 10). d is the bound for the SAT-based BMC, and is set to D_{max}. In practice, D_{max} should be low (e.g., about 30-50 [110]). Herein, we empirically set D_{max} as 30. $\mathcal{T}$ is the state transition formula of the ABN, and has been well defined in [43]. ϕ_{path} is a d-length unfolding of $\mathcal{T}$

and represents a d-length path in the STG of the ABN. Each arc of this path corresponds to a state transition of the ABN. $\chi(A, s^i)$ is the characteristic formula representing the set of states A in terms of variables of s^i. The characteristic formula of a set of states is defined based on the characteristic of a state: $\chi(A, s^i) = \bigvee_{s \in A} \chi(s, s^i)$. The characteristic formula of a state is defined as $\chi(s, s^i) = \bigwedge_{j=1}^{n} (s_j^i \leftrightarrow s_j)$. Then ϕ represents a d-length path satisfying its start state (i.e., s^0) is s and its end state (i.e., s^d) is in F. If **SAT**(ϕ), **ABNReach** can return true. Otherwise, **ABNReach** must start its final phase because SAT-based BMC is incomplete [110].

In the final phase, **ABNReach** simply uses the **UnfReach** algorithm presented in Subsection 4.4.5 (see Line 15 of Algorithm 10). We put **UnfReach** at the end of **ABNReach** because of the three following reasons. First, **UnfReach** still calculates the whole or a part of the reachable set, whereas **PintReach** and SAT-based BMC do not. Second, **UnfReach** is an on-the-fly method; it is also more efficient in the case of reachable. Third, **UnfReach** is an exact algorithm for checking the reachability in ABNs. Thus, we ensure that the result of **ABNReach** is correct.

Algorithm 10 ABNReach

Input: An ABN $\mathcal{A}$, a state s, a set F of states.
Output: Whether s reaches any state in F in $G(\mathcal{A})$?

1: $\mathcal{Q} \leftarrow$ the encoded asynchronous automata network of $\mathcal{A}$
2: $s^{\mathcal{Q}} \leftarrow$ the state of $\mathcal{Q}$ corresponding to s of $\mathcal{A}$
3: $F^{\mathcal{Q}} \leftarrow$ the set of states of $\mathcal{Q}$ corresponding to the states in F of $\mathcal{A}$
4: **if PintReach**$(\mathcal{Q}, s^{\mathcal{Q}}, F^{\mathcal{Q}}) = true$ **then**
5: **return** true
6: **else if PintReach**$(\mathcal{Q}, s^{\mathcal{Q}}, F^{\mathcal{Q}}) = false$ **then**
7: **return** false
8: **else**
9: $d \leftarrow D_{max}$
10: $\phi_{path} \leftarrow \bigwedge_{i=0}^{d-1} \mathcal{T}(s^i, s^{i+1})$
11: $\phi \leftarrow \chi(\{s\}, s^0) \wedge \phi_{path} \wedge \chi(F, s^d)$
12: **if SAT**(ϕ) **then**
13: **return** true
14: **else**
15: **return UnfReach**$(\mathcal{A}, s, F)$
16: **end if**
17: **end if**

4.6.2 Use of Negative Feedback Vertex Sets

The size of the set U (see Line 1 of Algorithm 6) may largely impact the performance of **FVS-ABN** (see Subsection 4.5.1). Using a smaller set of nodes may open a chance to get a smaller candidate set of states; thus to accelerate the running time of **FVS-ABN**. In a signed directed graph, the size of its minimum NFVS is less than or equal to the size of its minimum FVS, since an FVS is also an NFVS [99]. In addition, it is known that the presence of negative cycles is a necessary condition for the presence of cyclic attractors

of an ABN (see Theorem 4.6.1 [142]). These observations suggest us to explore relations between NFVSs and dynamics of ABNs.

For convenience, we first introduce some new notations as well as summarize some previous notations. $IG(\mathcal{N})$ denotes the interaction graph of a BN $\mathcal{N}$. $G(\mathcal{N})$ denotes the STG of an BN $\mathcal{N}$. $F(G)$ denotes the set of fixed points of an STG G. $R_{U,B}$ denotes the operation of systematically removing arcs from an STG with respect to a set U of nodes and a set B of retained values corresponding to the nodes of U. $R_{U,B}(G)$ denotes the reduced STG obtained by applying $R_{U,B}$ to an STG G.

Theorem 4.6.1. *[142] Let $\mathcal{A}$ be an ABN. If the interaction graph of $\mathcal{A}$ (i.e., $IG(\mathcal{A})$) has no negative cycle, then $\mathcal{A}$ has no cyclic attractor.*

Theorem 4.6.2. *Let $\mathcal{A}$ be an ABN. Let U^- be an NFVS of $IG(\mathcal{A})$ and B^- be a set of retained values corresponding to the nodes of U^-. Let att be an attractor of $\mathcal{A}$. Then there exists a state s such that $s \in att$ and s is a fixed point of the reduced STG (i.e., $R_{U^-,B^-}(G(\mathcal{A}))$).*

Proof. We first define $\Sigma(u,i)$ as the set of all states reachable from a state u by retaining the values of all nodes $x_j, j \in \{1,...,i\}$ and only updating the values of any node $x_j, j \in \{i+1,...,n\}$. We can obtain a property (*) satisfying $\Sigma(v,j) \subseteq \Sigma(u,i)$ with $j \geq i$ and $v \in \Sigma(u,i)$.

Without loss of generality, we can reorder the nodes of $\mathcal{A}$ such that $U^- = \{x_1,...,x_k\}, k \geq 0$. If $k = 0$, then $IG(\mathcal{A})$ has no negative cycle. In this case, $\mathcal{A}$ has only fixed points by Theorem 4.6.1, and then the theorem immediately holds. Thus, we consider the case $k \geq 1$.

Let $B^- = \{b_1,...,b_k\}$. We proceed the following procedure. First, we set i as 1 and randomly choose a state u^0 in att. Second, if $\Sigma(u^{i-1}, i-1)$ contains a state u' such that $u'_i = b_i$, then $u^i \leftarrow u'$. Otherwise, $u^i \leftarrow u^{i-1}$. Third, we increase i by 1. If $i \leq k$, we go back to the second step. Otherwise, we stop this procedure. We now obtain a state u^k. Then we have (a) $\forall i \in \{1,...,k\}$, either $u^i_i = b_i$ or all states $s \in \Sigma(u^{i-1}, i-1)$ satisfying $s_i = 1 - b_i$.

We retain the values of all nodes $x_i, i \in \{1,...,k\}$ in u^k and only update node $x_i, i \in \{k+1,...,n\}$. Eventually, we will reach a state u^* such that (b) $\forall i \in \{k+1,...,n\}, f_i(u^*) = u^*_i$, and (c) $\forall i \in \{1,...,k\}$, either $u^*_i = b_i$ or $f_i(u^*) = u^*_i = 1 - b_i$.

Indeed, let $\mathcal{A}^k$ be the ABN obtained by removing all nodes $x_i, i \in \{1,...,k\}$ from $\mathcal{A}$ and substituting the values of all nodes $x_i, i \in \{1,...,k\}$ in u^k to all Boolean functions $f_i, i \in \{k+1,...,n\}$. Since $U^- = \{x_1,...,x_k\}$ is an NFVS, $IG(\mathcal{A}^k)$ has no negative cycle. Then $\mathcal{A}^k$ has only fixed points by Theorem 4.6.1. Hence, every state in $\mathcal{A}^k$ will eventually reach a fixed point. We now can imply that (b) holds, since $\forall i \in \{1,...,k\}, u^k_i = u^*_i$.

Assume that $u^*_i = 1 - b_i, i \in \{1,...k\}$. We have $u^i_i = 1 - b_i$. By (a), all states $s \in \Sigma(u^{i-1}, i-1)$ must satisfy $s_i = 1 - b_i$. Let s' be the next state of u^* by updating only node x_i. By the definition of Σ, $s' \in \Sigma(u^*, i-1)$. By (*), we have $\Sigma(u^*, i-1) \subseteq \Sigma(u^{i-1}, i-1)$, since $u^* \in \Sigma(u^k, k) \subseteq \Sigma(u^{i-1}, i-1)$. Hence, $s' \in \Sigma(u^{i-1}, i-1)$ leading to $s'_i = 1 - b_i$. In other words, $f_i(u^*) = 1 - b_i$. We now can imply that (c) holds.

Since u^* satisfies (b) and (c), u^* is a fixed point in $R_{U^-,B^-}(G(\mathcal{A}))$. u^* is also in att, since $u^0 \in att$ and att is an attractor. Therefore, we can conclude the proof. □

Theorem 4.6.2 shows our new theoretical finding. Hereafter, we shall use an example to illustrate Theorem 4.6.2, as well as to show that this theorem only holds for the

case of NFVSs and does not hold for the case of PFVSs. Consider the ABN $\mathcal{A}$ given in Example 4.6.1. $IG(\mathcal{A})$ has a minimum PFVS ($U^+_{min} = \{x_1\}$), a minimum NFVS ($U^-_{min} = \{x_3\}$), and a minimum FVS ($U_{min} = \{x_1, x_3\}$). $G(\mathcal{A})$ has one fixed point ($\{111\}$) and one cyclic attractor ($\{000, 010, 011, 001\}$). $R_{U,B}(G(\mathcal{A}))$ denotes the reduced STG of $G(\mathcal{A})$ corresponding to the set of nodes U and the set of retained values B (see Section 4.3). $R_{U^-_{min},B^-}(G(\mathcal{A}))$ with $B^- = \{b_3\} = \{0\}$ is given in Figure 4.6a. As we can see, $F(R_{U^-_{min},B^-}(G(\mathcal{A}))) = \{010, 111\}$. This set covers all the attractors of $\mathcal{A}$, i.e., each attractor of $\mathcal{A}$ contains a fixed point of the reduced STG of $\mathcal{A}$. $R_{U^+_{min},B^+}(G(\mathcal{A}))$ with $B^+ = \{b_1\} = \{0\}$ is given in Figure 4.6b. As we can see, $F(R_{U^+_{min},B^+}(G(\mathcal{A}))) = \{111\}$. This set does not cover all the attractors of $\mathcal{A}$, since it does not contain any state of the cyclic attractor $\{000, 010, 011, 001\}$. This example shows that Theorem 4.6.2 does not hold for the case of PFVSs.

Example 4.6.1. *We give an ABN* $\mathcal{A} = \{V, F\}$, *where* $V = \{x_1, x_2, x_3\}$ *and* $F = \{f_1, f_2, f_3\}$ *with*

$$f_1 = x_1 \wedge x_2 \wedge x_3,$$
$$f_2 = x_1 \vee \neg x_3,$$
$$f_3 = (x_2 \wedge \neg x_3) \vee (x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (x_1 \wedge x_2 \wedge x_3).$$

Figures 4.5a. and 4.5b show the STG and the interaction graph of $\mathcal{A}$, *respectively.*

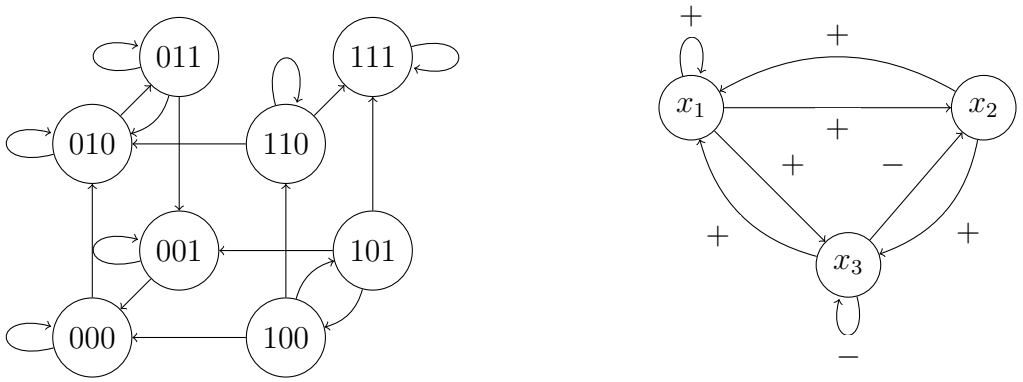


Figure 4.5: (a) STG and (b) interaction graph of the ABN given in Example 4.6.1.

By Theorem 4.6.2, the set of fixed points of the reduced STG (i.e., $F(R_{U^-,B^-}(G(\mathcal{A})))$) covers all the attractors of $\mathcal{A}$. Hence, $F(R_{U^-,B^-}(G(\mathcal{A})))$ is a sufficient candidate set of states. As a consequence, **iFVS-ABN** can use an NFVS U^- instead of an FVS U as in **FVS-ABN** to get the candidate set. Note that the size of $F(R_{U^-,B^-}(G(\mathcal{A})))$ may be larger than that of $F(R_{U,B}(G(\mathcal{A})))$. Hence, we propose an algorithm for calculating an NFVS, which ensures that the candidate set obtained by **iFVS-ABN** is always a subset of that obtained by **FVS-ABN**. The detail of this algorithm is as follows.

The problem of finding a minimum NFVS is NP-complete [99]. Since the problem of finding a minimum FVS has been proved NP-complete [100], a simple greedy algorithm (called **FindFVS**) has been proposed for finding an (not necessarily minimum) FVS (see Subsection 4.4.2). From these observations, we here propose a simple greedy algorithm

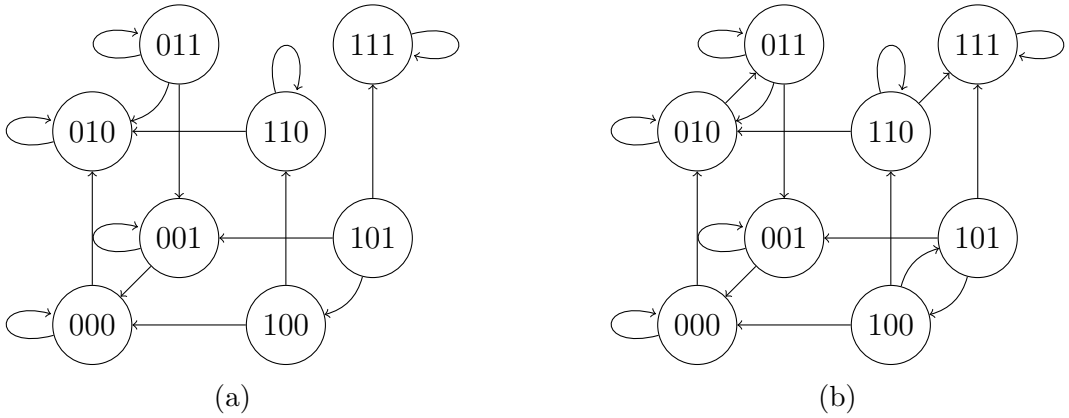


Figure 4.6: Reduced STGs of the ABN given in Example 4.6.1 with respect to (a) U^-_{min} and B^- and (b) U^+_{min} and B^+, respectively.

(named **FindNFVS**) for finding an (not necessarily minimum) NFVS of a signed directed graph. Algorithm 11 shows the description of the proposed algorithm. Algorithm 11 follows the method by [143] for checking whether a signed directed graph IG has a negative cycle. Herein, $IG - U^-$ is equivalent to $IG[V \backslash U^-]$, where V is the set of vertices of IG. We define the negative degree of a vertex v as the number of negative arcs starting or ending at v. Remark 4.6.1 convinces the usefulness of Algorithm 11.

Algorithm 11 FindNFVS

Input: A signed directed graph IG.
Output: An NFVS U^- of IG.

$U^- \leftarrow$ the set of vertices of IG having negative self loops
$U^-_{cand} \leftarrow$ the FVS obtained by applying the **FindFVS** algorithm to IG
$U^-_{cand} \leftarrow U^-_{cand} \backslash U^-$
$IG \leftarrow IG - U^-$
while IG has a negative cycle **do**
    Remove a vertex v from U^-_{cand} with a minimum of negative degree in IG
    $U^- \leftarrow U^- \cup \{v\}$
    $IG \leftarrow IG - \{v\}$
end while
return U^-

Remark 4.6.1. *The NFVS (say U^-) obtained by **FindNFVS** is always a subset of the FVS (say U) obtained by **FindFVS**. In other words, the NFVS used by **iFVS-ABN** is always a subset of the FVS used by **FVS-ABN**. Let B be a set of retained values corresponding to the nodes of U and let B^- be a set of retained values corresponding to the nodes of U^-. Equation 4.1 (presented in Subsection 4.4.3) shows the propositional formula that characterizes the set of fixed points of the reduced STG of $\mathcal{A}$ with respect to a set U of nodes and a set B of retained values. If $b^-_i = b_i, \forall x_i \in U^-$, then $F(R_{U^-,B^-}(G(\mathcal{A}))) \subseteq F(R_{U,B}(G(\mathcal{A})))$. Indeed, following Equation 4.1, a fixed point of $R_{U^-,B^-}(G(\mathcal{A}))$ is also a fixed point of $R_{U,B}(G(\mathcal{A}))$, since $U^- \subseteq U$. As a consequence, the candidate set of **iFVS-ABN** is always a subset of the candidate set of **FVS-ABN** if **iFVS-ABN** uses the retained set that is a subset of the retained set used by **FVS-ABN**. Both **iFVS-ABN***

*and **FVS-ABN** use the same algorithm (i.e., Algorithm 8 presented in Subsection 4.4.3) for setting the retained set. Hence, the candidate set of **iFVS-ABN** is always a subset of the candidate set of **FVS-ABN**. In addition, $F^{char}_{U,B}$ is clearly more complex than $F^{char}_{U^-,B^-}$, since $U^- \subseteq U$. Hence, the time for computing the candidate set of **iFVS-ABN** may be shorter than the time for computing that of **FVS-ABN**.*

$$F^{char}_{U,B} = \left\{ \bigwedge_{x_i \in U} [(f_i(x) \leftrightarrow x_i) \vee (x_i \leftrightarrow b_i \wedge f_i(x) \leftrightarrow 1 - b_i)] \right\} \wedge \left\{ \bigwedge_{x_i \notin U} [f_i(x) \leftrightarrow x_i] \right\} \quad (4.1)$$

4.6.3 Correctness

FVS-ABN is an exact method for finding attractors of an ABN (see Theorem 4.4.1). **iFVS-ABN** has two changes from **FVS-ABN**. We shall formally prove the correctness of **iFVS-ABN** as in Theorem 4.6.3. Note that this proof is similar to the proof for the correctness of **FVS-ABN**. For convenience, we repeat some details of Theorem 4.4.1.

Theorem 4.6.3. ***iFVS-ABN*** *finds exactly all attractors of an ABN.*

Proof. Let $\mathcal{A}$ be an ABN. Let U^- be the NFVS obtained by applying Algorithm 11 to $IG(\mathcal{A})$. Then, let B^- be the chosen retained set obtained by using Algorithm 8.

By Theorem 4.6.2, every attractor of the STG of $\mathcal{A}$ (say G) always contains at least one state that is a fixed point of the reduced STG of $\mathcal{A}$ (say G'). Hence, every attractor of G contains at least one state in F after finishing Line 5 of Algorithm 6. As demonstrated in Subsection 4.4.4, **Preprocessing-SSF** does not disappear cyclic attractors of G in F. Therefore, every attractor of G contains at least one state in $A \cup F$ (*) when starting the *while* loop of Algorithm 6. Through the *while* loop, the property (*) is preserved. Indeed, in each iteration if s reaches in G a state s' in $A \cup F$ and s belongs to an attractor att of G, then s' also belongs to att by the definition of an attractor. In this case, s is not added to $A \cup F$. However, att still contains s' that is a state in $A \cup F$. Note that we have this preservation because the result of **ABNReach** is correct (see Subsection 4.4.5).

After finishing the *while* loop, each attractor of G contains at least one state in A (a), since $F = \emptyset$. We show that A has no two states s and s' such that s and s' are in the same attractor of G (b). Indeed, if s and s' is in the same attractor of G, then s and s' reach each other in G. If s is traversed before s', then s cannot be added to A. If s' is traversed before s, then s' cannot be added to A. Moreover, there is no state s in A such that s is not in any attractor (c). Indeed, if such a state s exists, it must reach at least an attractor, implying that it reaches $A \cup F$ in G. Hence, s cannot be added to A. We have a contradiction.

From (a), (b), and (c), we have a one-to-one correspondence between A and the set of attractors of G. Hence, the result of **iFVS-ABN** is correct, i.e., this method finds exactly all attractors of $\mathcal{A}$. □

4.6.4 Evaluation

We have implemented **iFVS-ABN** in a JAVA tool. This tool uses the JDD library [112] for BDD manipulation (e.g., storing Boolean functions or sets of states by BDDs) and

Z3 [118] as the SAT solver for SAT-based BMC. An executable file of the implemented tool and some examples of real biological networks are available at https://github.com/giang-trinh/iFVS-ABN. We then conducted two experiments to evaluate the efficiency of the two improvements. The first experiment aims at evaluating the efficiency of **ABNReach** (i.e., the new method for checking the reachability in ABNs). We compared the performance of a variant of **FVS-ABN** (say **FVS-ABN***) and **FVS-ABN**. **FVS-ABN*** uses an FVS as **FVS-ABN** but uses **ABNReach** for checking the reachability in ABNs. The second experiment aims at evaluating the efficiency of the use of NFVSs. We compared the performance of **FVS-ABN*** and **iFVS-ABN**. All the experiments were run on a virtual machine whose environment is CPU: Intel(R) Xeon(R) Silver 4116 4x2.10GHz, Memory: 24 GB, CentOS 7 64 bit.

In the first experiment, we randomly generated two sets of BNs by using **Bool Net R** package [117]. The first set includes N-K BNs [1] with network size n in the set {120, 130, 140, 150, 160, 170, 180, 190, 200} and $K = 2$ (i.e., each node has exactly $K = 2$ input nodes). We used N-K BNs because they are commonly used for benchmarking in existing literature (see, e.g., [10, 14]). For each network size, 20 instances were generated using the **generateRandomNKNetwork** function. The second set includes canalyzing and scale-free BNs with network size n in the set {100, 150, 200, 250, 300, 350, 400, 500, 1000} and $K = 10$ (i.e., the maximum number of input nodes for each node is $K = 10$). **iFVS-ABN** is applicable for any type of Boolean functions or network topology. However, canalyzing functions are known as biologically relevant functions [90]. In addition, many systems (including gene regulatory networks) possess the scale-free topology in which the number of input nodes for each node follows the scale-free Zeta distribution [120]. For each network size, 20 instances were also generated using the **generateRandomNKNetwork** function with the parameters: $K = 10$, $topology = "scale\_free"$, $functionGeneration = "generateCanalyzing"$. We then applied **FVS-ABN*** and **FVS-ABN** to these randomly generated networks and recorded the number of failures (i.e., failed to obtain the result within three hours). Note that there is a network reduction technique that preserves all the attractors of an ABN [122]. Hence, we used this reduction technique for all the compared methods.

Table 4.2: Numbers of failures of **FVS-ABN*** and **FVS-ABN** on N-K networks.

n	120	130	140	150	160	170	180	190	200
FVS-ABN*	6	9	9	10	10	12	15	18	19
FVS-ABN	10	13	13	15	16	16	18	19	19

Table 4.2 shows the experimental results of **FVS-ABN*** and **FVS-ABN** on N-K networks. As we can see, **FVS-ABN** and **FVS-ABN*** can handle networks of up to 200 nodes. In each network size (except $n = 200$), the number of failures of **FVS-ABN** is always greater than that of **FVS-ABN***. This observation shows the efficiency of **ABNReach** in terms of N-K networks.

Table 4.3 shows the results on canalyzing and scale-free networks. In each network size, the number of failures of **FVS-ABN*** is always much less than that of **FVS-ABN**. Especially, for $n = 1000$, **FVS-ABN*** can even handle 8/20 instances, whereas **FVS-ABN** failed in all instances. This observation shows the efficiency of **ABNReach** in terms of canalyzing and scale-free BNs.

Table 4.3: Numbers of failures of **FVS-ABN*** and **FVS-ABN** on canalyzing and scale-free networks.

n	100	150	200	250	300	350	400	500	1000
FVS-ABN*	0	0	0	3	1	2	0	3	12
FVS-ABN	2	3	5	10	10	10	10	14	20

As we can see, the performance of **FVS-ABN*** for canalyzing and scale-free networks is better than that for N-K networks. This observation is reasonable, since N-K networks are generally more complex than canalyzing and scale-free networks. As many previous methods (e.g., [43, 49]), **FVS-ABN*** seems to be less efficient for N-K networks (even with $K = 2$). In addition, in most randomly generated networks, $U = U^-$ (i.e., the used FVS is equal to the used NFVS); therefore, the performance of **FVS-ABN*** is generally not different from the performance of **iFVS-ABN**. Hence, in the second experiment, we used real biological networks to highlight the performance difference between **FVS-ABN*** and **iFVS-ABN**.

In the second experiment, we used real biological networks obtained from The Cell Collective [119]. To evaluate the efficiency of the use of NFVSs, we only chose the networks where $U \neq U^-$ (i.e., the used FVS is not equal to the used NFVS). In total, we obtained a set of 13 real biological networks whose sizes range from 26 to 321. We then applied **FVS-ABN*** and **iFVS-ABN** to BNs of these networks. We set the time limit for each network to 10 hours, since the running time may be very long for large-scale BNs. Note that we also used the network reduction technique [122] as in the first experiment, since this reduction technique preserves all the attractors of an ABN.

Table 4.4: Experimental results of **FVS-ABN*** and **iFVS-ABN** on real biological networks.

			FVS-ABN*		**iFVS-ABN**	
name	n	$\|A\|$	$\|U\|$	time (secs)	$\|U^-\|$	time (secs)
Differentiation_of_T_lymphocytes	50	2050	18	952.73	6	578.09
HumanMyelomaCells	67	83	13	173.25	6	105.42
HGF_Signaling_in_Keratinocytes	68	72	10	2.34	0	0.54
Influenza_A_Virus_Replication_Cycle	131	524	29	-	10	42.33
Signaling_in_Macrophage_Activation	321	4096	16	21216.07	1	6712.42
Wg_Pathway_of_Drosophila	26	16384	15	3.67	1	3.67
TumourCell	32	9	10	0.81	5	0.54
TCellSignaling	40	8	5	0.46	2	0.45
Treatment_of_Castration_Resistant	42	16384	14	0.40	0	0.36
Senescence	51	17	10	22.91	5	18.94
PC12CellDifferentiation	62	3	2	0.45	0	0.46
YeastApoptosis	73	8448	16	76.20	3	76.74
IL_6_Signalling	86	32768	21	2127.45	10	2056.36

Table 4.4 shows the experimental results of **FVS-ABN*** and **iFVS-ABN** on real biological networks. Columns "name", n, and $|A|$ denotes the network name, the number of nodes, and the number of ABN attractors, respectively. Columns $|U|$ and $|U^-|$ denote the sizes of the used FVS and the used NFVS, respectively. Column "time

(secs)" denotes the running time in seconds. "-" stands for the case of timeout. In 5/13 BNs (e.g., the Differentiation_of_T_lymphocytes network), **iFVS-ABN** is much faster than **FVS-ABN***. Especially in the Influenza_A_Virus_Replication_Cycle and Signaling_in_Macrophage_Activation networks, **iFVS-ABN** outperforms **FVS-ABN***. In these networks, $|U^-|$ is much less than $|U|$. In the 8/13 remaining BNs, the running time of **iFVS-ABN** is comparable to the running time of **FVS-ABN***. In some networks (e.g., the YeastApoptosis network, the IL_6_Signalling network), **iFVS-ABN** is slower than **FVS-ABN*** although $|U^-|$ is much less than $|U|$. We realized that $|F|$ (before applying **Preprocessing-SSF**) of **FVS-ABN*** and **iFVS-ABN** are the same in these networks. Hence, this observation may be due to the randomness of **Preprocessing-SSF**. However, the difference is insignificant. In particular, **iFVS-ABN** can handle the Signaling_in_Macrophage_Activation network (a large network with $n = 321$) in reasonable time (less than two hours). Furthermore, the time for computing the used NFVS is insignificant (less than two seconds in most cases). These observations show the efficiency of the use of NFVSs.

4.7 Discussion

In this chapter, we have formally stated and proved the relations between an FVS of a BN and the dynamics of this BN. These relations do not depend on the updating scheme of the BN; thus, they can be used for analyzing different types of BNs, such as, ABNs and GABNs. From these relations, we have proposed an FVS-based method (called **FVS-ABN**) for finding all attractors of an ABN. Our approach relies on the principle of removing arcs in the STG of the ABN to get a candidate set of states and the reachability analysis on the ABN to filter out this candidate set. The obtained set one-to-one corresponds to the set of attractors. We have also formally proved the correctness of **FVS-ABN**. In addition, we have proposed a preprocessing procedure (called **Preprocessing-SSF**) to reduce the computational burden, whereas the correctness of **FVS-ABN** is preserved. Finally, we have developed an unfolding-based and on-the-fly method for checking the reachability in ABNs. In principle, **FVS-ABN** works well on large networks having relatively small FVSs and not too large attractors. Fortunately, these characteristics are often found in real biological networks [14, 45].

We have implemented **FVS-ABN** as a JAVA tool, and then used this tool to conduct experiments on real biological networks and randomly generated N-K networks. The experimental results confirm the usefulness of **Preprocessing-SSF** and are very promising because **FVS-ABN** can handle large networks of up to 101 nodes without using any network reduction technique. These results also show the effectiveness of **FVS-ABN** as compared to the two state-of-the-art methods, **genYsis** and **CABEAN**. Note that the main advantage of our approach is to reduce the attractor detection in ABNs to the reachability problem in ABNs. This advantage opens a chance to efficiently solve the attractor detection in ABNs by applying the advents in reachability research that can handle very large asynchronous networks [140, 141].

Since **FVS-ABN** encompasses many constituent steps, it is potentially possible to improve it. There are some potential ways to improve **FVS-ABN**. The first way is to reduce the number of fixed points of the reduced STG. We can use an exact method for finding a minimum FVS (instead of using the greedy algorithm **FindFVS**) or a more

efficient heuristic for setting the retained set B. The second way is to propose an efficient heuristic for variable ordering, since **Preprocessing-SSF** uses BDDs. A good variable ordering can reduce significantly the computational time of **Preprocessing-SSF**. The third way is to use a more efficient method or to efficiently combine multiple techniques for checking the reachability in ABNs. We can consider some other techniques in terms of Petri net unfoldings, such as, contextual Petri nets [144], merged processes [145], unravelings [146]. Furthermore, we can also use some static analyzers (e.g., **PintReach** [140], **ASPReach** [141]) in a preprocessing step.

Following the possible ways aforementioned, we have proposed a new method (called **iFVS-ABN**) that includes the two substantial improvements to the previous method, **FVS-ABN**. First, we have proposed the **ABNReach** algorithm that reasonably combines multiple previous methods to efficiently check the reachability in ABNs. Second, we have formally stated and proved the relation between an NFVS of the interaction graph of an ABN and the dynamics of the ABN. This relation is a new theoretical finding and can be extended to that for other types of BNs, such as, GABNs and a non-standard type of asynchronous BNs, random order asynchronous Boolean networks [9, 10]. Based on this relation, **iFVS-ABN** uses an NFVS instead of an FVS as **FVS-ABN** to obtain the candidate set of states. In addition, we have also developed a simple but useful greedy algorithm for calculating an NFVS that is used in **iFVS-ABN**. Then, we have formally proved the correctness of **iFVS-ABN**. The efficiency of **iFVS-ABN** was evaluated by the experiments on various types of networks. The experimental results show that the two improvements are effective and **iFVS-ABN** outperforms the original one, **FVS-ABN**. In particular, **iFVS-ABN** can handle large networks with up to 1000 nodes in terms of randomly generated networks and 321 nodes in terms of real biological networks. This advanced computation capability can enable biologists to conduct more accurate analysis on large networks, then to discover more biological insights.

Although **iFVS-ABN** outperforms the previous method in terms of N-K networks [5], it does not handle well large networks whose sizes start from 200 nodes. N-K networks are one helpful tool, which allows investigating a variety of different dynamic properties of BNs and regulatory systems [6]. In addition, N-K networks are usually used in statistical studies [54, 107, 147, 148, 149] for discovering insights into the dynamics of BNs. Hence, improving **iFVS-ABN** to handle well larger N-K networks is necessary. Moreover, the applicable range of **iFVS-ABN** is so far from genome-scale networks that can possess thousands of genes. Making **iFVS-ABN** capable to handle such networks is needed, and is one of future work.

Finally, note that both **FVS-ABN** and **iFVS-ABN** use **Mole** and some other tools as subroutines. Since the computational complexity of such tools is unclear, it is difficult to analyze the computational complexity of the whole algorithm. We leave the analysis of theoretical or practical computational complexity as future work. In addition, Random Order Asynchronous Boolean Networks (ROABNs) [9, 10] are also a popular type of asynchronous BNs and still get much attention from research communities [3, 9, 150, 151]. Hence, it is worth to study the dynamics of an ROABN. In the future, we intend to extend our methods (**FVS-ABN** and **iFVS-ABN**) for ABNs to those for ROABNs. It is interesting but challenging because ROABNs are generally more complex in dynamics than ABNs [9].

Chapter 5

Attractor Detection in Deterministic Generalized Asynchronous Boolean Networks (DGABNs)

5.1 Introduction

In the biological context, the synchronous updating class (e.g., SBNs [1]) was criticized because of the assumption that the dynamics of Gene Regulatory Networks (GRNs) is deterministic and synchronous, i.e., all genes change their expression levels simultaneously [43]. Thus, the asynchronous updating class, in which all genes take different time to change their expression levels, is closer to biological phenomena [10, 43, 96]. For example, a very recent work [97] has explicitly backed up the necessity of asynchronous models for modeling GRNs over a realistic proof-of-concept case study. The proposed alternative is the non-deterministic asynchronous updating class (e.g., ABNs [10]) with the assumption that only one randomly selected gene can be updated at a single step. However, it did not give encouraging results, since the networks change drastically their properties due to the non-determinism [13]. Moreover, this updating class has also some disadvantages including the high complexity of the state transition graph and the inclusion of many incompatible or unrealistic pathways [114]. With this motivation, a new class of updating schemes, asynchronous but deterministic, was proposed [14, 152]. The deterministic asynchronous updating class can help us to model and analyze biological networks more reasonably [9, 13]. For example, we can model asynchronous phenomena that are not random, a thing that is quite difficult with the non-deterministic asynchronous updating class [13]. Furthermore, models with deterministic asynchronous updating class are particularly useful when information about the kinetics of biological processes is known [19].

To date, there are various types of BNs with the deterministic asynchronous updating class: Deterministic Generalized Asynchronous Boolean Networks (DGABNs) [14], Deterministic Asynchronous Boolean Networks (DABNs) [14], Mixed-Context Boolean Networks (MxBNs) [13], Deterministic Asynchronous (DA) models [79], and Block-Sequential Boolean Networks (BSBNs) [152]. They have been widely used in modeling and analysis of GRNs [9, 79, 82, 153]. In this chapter, we focus on DGABNs that are a typical type of deterministic asynchronous BNs. There are three reasons for this choice.

First, DGABNs offer an interesting compromise between SBNs and ABNs, thus could provide a suitable modeling formalism of various types of systems [75]. For example, the

authors of [154] applied the updating schemes of DGABNs and ABNs to the model of the Spatial Prisoner's Dilemma game, which is the most used game in the area of evolutionary game theory. Based on simulations, they obtained that these two updating schemes lead basically the same outcome of the model. Recently, Martin Schneiter et al. [155] used different types of BNs to formulate a simplified pluricellular epithelium model, which intends to present plausibly the self-organization of ciliary beating patterns as well as of the associated fluid transport across the airway epithelium. The simulation results show that DGABNs lead to more realistic dynamics (flexibility and robustness) and may therefore be favored by evolution. Many other applications of DGABNs in various fields can be found (see, e.g., [3, 156]).

Second, DGABNs are general and interesting mathematical objects because there is no restriction on their Boolean functions and contexts. For example, SBNs are a special case of DGABNs [14, 75]. In addition, studying DGABNs can be a good starting point for further studies on more complex models such as DABNs or MxBNs [13]. This is reasonable because both DABNs and MxBNs are constructed based on DGABNs and seem to be more computationally complex than DGABNs [5, 13].

Third, to our best knowledge, all the previous studies on DGABNs are theoretical or simulation-based. For example, Carlos Gershenson first proposed DGABNs [14] and analyzed their dynamics by simulations [13, 14]. Li et al. [157] proposed a semi-tensor product-based framework to study many variants of asynchronous BNs including DGABNs. This approach is deeply theoretical but not practical, since the sizes of the matrices that need to be considered are exponential in the number of nodes. Hence, analytical and practical studies for DGABNs are needed. In general, the deterministic behavior of DGABNs makes them relatively easy to analyze as compared to non-deterministic asynchronous BNs (e.g., ABNs) [14]. Since many analytical and practical studies have been done for ABNs [19, 43], such studies for DGABNs are potentially possible.

In this dissertation, two central issues of GRNs (see, e.g., [21]), attractor detection and optimal control, are well studied. We first formulate the Extended State Transition Graph (ESTG) of a DGABN (see Section 5.2). The state transition is encoded as a Satisfiability Modulo Theory (SMT) formula. This formulation is a starting point for further analysis of DGABNs. Next, we formally state and prove several relations in dynamics between DGABNs and other popular models (see Section 5.3) including deterministic asynchronous models [79], block-sequential Boolean networks [152], generalized asynchronous Boolean networks [14], and mixed-context Boolean networks [13]. We then propose one SMT-based method for attractor detection (see Section 5.4) and two SMT-based methods for optimal control of DGABNs, respectively. These methods are implemented in a JAVA tool called **DABoolNet**. In order to highlight the scalability of the proposed methods, two experiments on randomly generated networks (see Section 5.5) and one artificial network are conducted. To our best knowledge, **DABoolNet** is the first analytical and practical tool for attractor detection and optimal control of DGABNs. In addition, several case studies are presented to show the applications of our methods. For attractor detection in DGABNs, we apply **DABoolNet** to two real biological networks and compare the obtained results to the previous insights into these networks found in the literature (see Subsection 5.4.2). We also use **DABoolNet** to verify several insights into the dynamics of random Boolean networks (i.e., N-K models) presented in [13, 14] (see Subsection 5.4.3). For optimal control of DGABNs, we apply **DABoolNet** to one real biological network.

Since the dissertation is divided into two parts, the Attractor Detection and Optimal

Control parts, this chapter only presents the obtained results on attractor detection in DGABNs. The obtained results on optimal control of DGABNs shall be presented in Chapter 6.

5.2 Extended State Transition Graph

A DGABN has a set of n nodes ($V = \{x_1, ..., x_n\}$). Each node x_i is identified as a Boolean variable, and is associated with a Boolean function f_i ($f_i : \mathbb{B}^{|IN(f_i)|} \rightarrow \mathbb{B}$). Each node x_i is also associated with two parameters: $p_i \in \mathbb{N}^+$ and $q_i \in \mathbb{N}$ ($q_i < p_i$). p_i defines the period between two consecutive updates of node x_i and q_i determines the time to the first update of node x_i. We use Λ to denote a parameter indicating the maximum allowed period of a node (i.e., $p_i \leq \Lambda, \forall i \in \{1, ..., n\}$). We also use γ to denote the least common multiple of all p's. The set of all $p's$ and $q's$ is called the *context* of a DGABN. Example 5.2.1 is an example of DGABNs.

Example 5.2.1. *Consider a DGABN $\mathcal{D}$ of two nodes (x_1, x_2). Its Boolean functions and context are given as*

$$f_1 = (x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2), f_2 = x_1;$$
$$p_1 = 1, p_2 = 2, q_1 = 0, q_2 = 0.$$

$x_i(t) \in \mathbb{B}$ denotes the value of node x_i at time t. A state of a DGABN at time t is denoted by $x(t) = (x_1(t), ..., x_n(t))^\top$. At time t, node x_i will be updated by $x_i(t+1) = f_i(x(t))$ when the modulus of time t over p_i is equal to q_i (i.e., $t\%p_i = q_i$). For example, at time $t = 1$, only node x_1 of the DGABN shown in Example 5.2.1 will be updated. If two or more nodes will be updated, they will be updated simultaneously (e.g., at time $t = 0$, node x_1 and node x_2 of $\mathcal{D}$ will be updated simultaneously). Then, the current state $x(t)$ will transit to the next state $x(t+1)$. This is a state transition. The dynamics of a DGABN is deterministic because $x(t+1)$ is uniquely determined for given $x(t)$ and t. In particular, if all $p's$ are 1, then a DGABN becomes a SBN in which all the nodes will be updated simultaneously at any time t [14].

The dynamics of a DGABN depends on the initial state (the state of the DGABN at time $t = 0$). There are 2^n possible initial states. Since the evolution of DGABNs is based on modulus arithmetic, we only need to consider the scaled time t_{scaled} of t to γ (i.e., $t_{scaled} = t\%\gamma$). Indeed, the pattern of updating nodes is repeated after each γ time steps. Note that the visited states are not necessarily repeated. Figure 5.1 shows the dynamics of the DGABN shown in Example 5.2.1. Herein, circles denote states and dashed circles denote initial states of the DGABN. An arc and its above text denote a state transition and the scaled time of t when the state transition occurs, respectively. The DGABN is updated in a pattern with period of two: x_1 and x_2 together, x_1 alone. If the DGABN starts with state 10, then x_1 and x_2 will be updated simultaneously, leading to state 01. Next, x_1 will be updated, leading to state 01. In the next time step, the pattern is repeated (i.e., x_1 and x_2 will be updated simultaneously). However, the next state (i.e., 00) has not been visited.

Start from an initial state, the DGABN will eventually lead to an *attractor* because the number of states of the DGABN is finite. Definition 2.3.1 is a general definition of

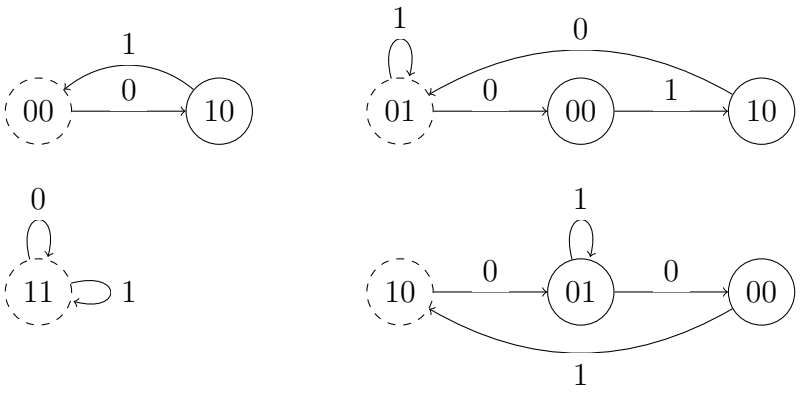


Figure 5.1: Dynamics of the DGABN shown in Example 5.2.1.

an attractor for all types of BNs. An attractor is said to be a fixed point or a cyclic attractor if it consists of only one state or at least two states, respectively. The fixed points of a BN are the same regardless of its updating scheme. The cyclic attractors of deterministic (e.g., DGABNs) and non-deterministic (e.g., ABNs) asynchronous BNs can be different. In non-deterministic asynchronous BNs, the system oscillates irregularly among the states of a cyclic attractor due to the randomness involved in the updating scheme. In this case, the cyclic attractor is referred to as a loose attractor [10]. In deterministic asynchronous BNs, the system oscillates regularly among the states in a cyclic attractor due to the deterministic dynamics. In this case, the cyclic attractor is referred to as a limit cycle [158]. However, there is no systematic definition for limit cycles of a DGABN. Thus, we still use the term of cyclic attractors for DGABNs. Reconsider the DGABN $\mathcal{D}$ shown in Example 5.2.1. If it starts from 11, it will reach a fixed point ($\{11\}$). If it starts from 00, it will reach a cyclic attractor ($\{00, 10\}$). If it starts from 01 or 10, it will reach the same cyclic attractor ($\{01, 00, 10\}$).

Although the definition of a DGABN enables us to study the behavior of a GRN in the long run, it does not provide a systematic mean for its analysis. We here propose a synchronization method for DGABNs, which provides a synchronous representation for the dynamics of a DGABN. This method can pave potential ways for analysis and control of DGABNs.

We define an *extended state* of a DGABN, which includes a state of this DGABN and an embedded value that represents the scaled time t_{scaled} of time t when reaching this state. Formally, $es \in \mathbb{B}^n \times \{0, ..., \gamma - 1\}$ is an extended state, where es_i $(i = 1, ..., n)$ denotes the value of internal node x_i and es_{n+1} denotes the value of the embedded time. We define the *image* of an extended state es as a state $[[es]]^I \in \mathbb{B}^n$ satisfying $[[es]]^I_i = es_i, i = \{1, ..., n\}$. Furthermore, we have $[[ES]]^I = \bigcup_{es \in ES}[[es]]^I$ where ES is a set of extended states. For clarification, we denote an extended stated es by $([[es]]^I, es_{n+1})$. For example, $(00, 1)$ means that $x_1 = 0$, $x_2 = 0$, and $x_3 = t_{scaled} = 1$. Then, the transition formula between two extended states is given as in Equation 5.1, where es^j denotes the current extended state and es^{j+1} denotes the next extended state; "%" is the modulus operator; "=" is the logical predicate EQUALITY. Herein, the scaled time of the current extended state will increase by one; if it equals to γ, then it is set to 0. This characteristic is presented by $es^{j+1}_{n+1} = (es^j_{n+1} + 1)\%\gamma$. The value of the ith node will be updated if the modulus of time t over p_i is equal to q_i and be not changed otherwise. The former case is presented by $[es^j_{n+1}\%p_i = q_i \wedge (es^{j+1}_i = f_i(es^j)]$, whereas the latter case is presented by

$[es^j_{n+1}\%p_i \neq q_i \wedge (es^{j+1}_i = es^j_i)]$. In addition, this transition formula is in form of an SMT formula in infix notation [118]. Note that each es^j_i $(i = 1, ..., n)$ corresponds to a Boolean SMT variable and each es^j_{n+1} corresponds to an integer SMT variable.

$$T(es^j, es^{j+1}) := \left\{es^{j+1}_{n+1} = (es^j_{n+1} + 1)\%\gamma\right\} \wedge \\ \bigwedge_{i=1}^{n} \left\{ \left[es^j_{n+1}\%p_i = q_i \wedge (es^{j+1}_i = f_i(es^j)\right] \vee \left[es^j_{n+1}\%p_i \neq q_i \wedge (es^{j+1}_i = es^j_i)\right] \right\} \tag{5.1}$$

From the definition of an extended states and the transition formula between two extended states, the whole dynamics of a DGABN can be captured by an Extended State Transition Graph (ESTG). An ESTG is a directed graph such that a node represents an extended states and an arc represents the transition between two extended states, respectively. Hereafter, we discuss the number of extended states of the ESTG of a DGABN. Since $es_{n+1} \in \{0, ..., \gamma - 1\}$, we can have $\gamma \times 2^n$ possible extended states. However, the number of possible initial extended states is only 2^n instead of $\gamma \times 2^n$ because in an initial extended state, es_{n+1} is 0, i.e., the start time is always 0. Hence, the ESTG may have less than $\gamma \times 2^n$ extended states. The extended states, which are not in the ESTG, are called *spurious* extended states. For example, consider a DGABN shown in Example 5.2.2. Its ESTG is given in Figure 5.2b. As we can see, the extended state $(00, 1)$ is not in this ESTG, thus it is a spurious extended state.

Example 5.2.2. *Consider a DGABN $\mathcal{D}$ of two nodes (x_1, x_2). Its Boolean functions and context are given as*

$$f_1 = x_1 \vee (\neg x_1 \wedge x_2), f_2 = (x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2); \\ p_1 = 2, p_2 = 1, q_1 = 0, q_2 = 0.$$

By the definition of an ESTG, each extended state in the ESTG of a DGABN has exactly one successor extended state. In the case that $\gamma = 1$, two consecutive extended states of the ESTG may be the same because es_{n+1} is always 0. Thus, the ESTG can have fixed points or limit cycles. We define a limit cycle of length $p > 1$ as the sequence $es^0, ..., es^{p-1}$ of extended states such that es^j are pairwise distinct, es^{j+1} is the next extended state of es^j in the ESTG for all $j \in \{0, p - 2\}$, and es^0 is the next extended state of es^{p-1} in the ESTG. Note that a fixed point can be seen as a limit cycle of length one. In the case that $\gamma > 1$, two consecutive extended states are always different because they at least differ in the embedded time. Hence, the ESTG has only limit cycles. In addition, based on Equation 5.1, we can easily imply that the length of a limit cycle of the ESTG is a multiple of γ.

Since an ESTG can capture the whole dynamics of a DGABN, the attractors of the DGABN are represented by fixed points or limit cycles of its ESTG. Since a fixed point may only appear when $\gamma = 1$, the length of an attractor (i.e., the length of a fixed point or a limit cycle) is a multiple of γ. Especially, a fixed point $\{s\}$ of the DGABN is represented by a limit cycle c of its ESTG where the length of c is γ and any extended state es in c satisfies $[[es]]^I = s$ (see Example 5.2.3). Note that since the ESTG of a DGABN may have $\gamma \times 2^n$ extended states, naive approaches for finding attractors (e.g., constructing the ESTG and then applying graph algorithms) are intractable when n is large. In Section 5.4, we shall consider how to efficiently compute these attractors.

Example 5.2.3. *Reconsider the DGABN shown in Example 5.2.1. Its ESTG is shown in Figure 5.2a. Since* $\gamma = 2$*, this ESTG has no fixed points. This ESTG has three limit cycles including* $\{(11,0),(11,1)\}$*,* $\{(00,0),(10,1)\}$*, and* $\{(01,0),(00,1),(10,0),(01,1)\}$*.* $\{(11,0),(11,1)\}$ *corresponds to the fixed point* $\{11\}$ *of the DGABN, whereas* $\{(00,0),(10,1)\}$ *and* $\{(01,0),(00,1),(10,0),(01,1)\}$ *correspond to the two cyclic attractors* $\{00,10\}$ *and* $\{01,00,10\}$ *of the DGABN, respectively.*

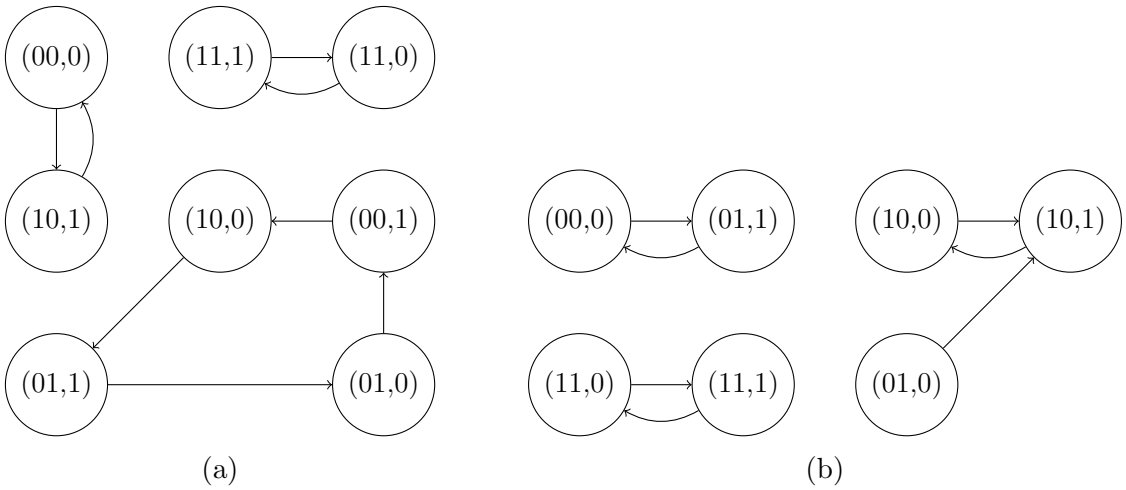


Figure 5.2: (a) ESTG of the DGABN shown in Example 5.2.1. (b) ESTG of the DGABN shown in Example 5.2.2.

5.3 Relations in Dynamics between DGABNs and Other Models

In this section, we shall analyze relations in dynamics between DGABNs and other models, such as, Deterministic Asynchronous (DA) models [9, 79, 80], block-sequential Boolean networks [81, 82], generalized asynchronous Boolean networks [14], and mixed-context Boolean networks [13]. The obtained relations are theoretical findings contributing to understanding the dynamics of Boolean networks. These findings also pave the potential ways to analyze these other models based on DGABNs.

5.3.1 Relations to Deterministic Asynchronous Models

In a DA model, each node x_i is associated with a pre-selected time unit $\gamma_i \geq 1$ [9]. The update of a node depends on the current time step by

$$x_i(t+1) = \begin{cases} f_i(x(t)) & \text{if } t+1 = k\gamma_i, k \in \{1,2,...\}, \\ x_i(t) & \text{otherwise.} \end{cases}$$

If multiple nodes can update at time t, then they will update synchronously.

Definition 5.3.1. *Given a DA model. Let* $\mathcal{D}$ *be its corresponding DGABN.* $\mathcal{D}$ *and the DA model share the same sets of nodes and Boolean functions. The context of* $\mathcal{D}$ *is as follows:* $p_i = \gamma_i, q_i = p_i - 1, i \in \{1,...,n\}$.

Obviously, a DA model and its corresponding DGABN specified by Definition 5.3.1 have the same dynamical behavior. Indeed, assume that the node x_i of the DA model will be updated at time t, i.e., $t+1 = k\gamma_i, k \in \{1,2,...\}$. In the DGABN, $t\%p_i = (k\gamma_i - 1)\%\gamma_i = \gamma_i - 1 = q_i$, thus x_i will also be updated at time t. This finding allows us to directly apply the methods developed for DGABNs to DA models.

5.3.2 Relations to Block-Sequential Boolean Networks

A Block-Sequential Boolean Network (BSBN) [81, 82] is a tuple $\langle V, F, d\rangle$ where $V = \{x_1, ..., x_n\}$ is the set of nodes, $F = \{f_1, ..., f_n\}$ is the set of Boolean functions associated with the nodes, and d is a deterministic updating scheme (see Definition 5.3.2). At each time step, nodes in a block are updated in parallel, but blocks follow each other sequentially along with d, leading to a new state. Since each state has only one outgoing transition, a BSBN may have two types of attractors including fixed points and limit cycles [81]. Note that the time unit of a BSBN is $nb(d)$ times of the time unit of SBNs or DGABNs. Example 5.3.1 shows an example of BSBNs.

Definition 5.3.2 (Adapted from [81])**.** *A deterministic updating scheme on the set V of n nodes is a function $d : \{1, ..., n\} \to \{1, ..., m\}, m \leq n$. A block of d is the set $B_i = \{v \in V | d(v) = i\}, 1 \leq i \leq m$. The number of blocks of d is denoted by $nb(d) \equiv m$. Frequently, d will be denoted as a sequence of blocks, i.e., $d = (j \in B_1)(j \in B_2)...(j \in B_{nb(d)})$.*

Example 5.3.1. *Let $\mathcal{B} = \langle V, F, d\rangle$ be a BSBN, where V and F are given as in DGABN shown in Example 5.2.1 and $d = (x_1)(x_2)$. The STG of $\mathcal{B}$ is given in Figure 5.3a. For example, in state 00, x_1 is updated, leading to a new state 10, then x_2 is updated, leading to a new state 11. Now, (00, 11) is a state transition of $\mathcal{B}$. As we can see, $\mathcal{B}$ has only one fixed point ($\{11\}$).*

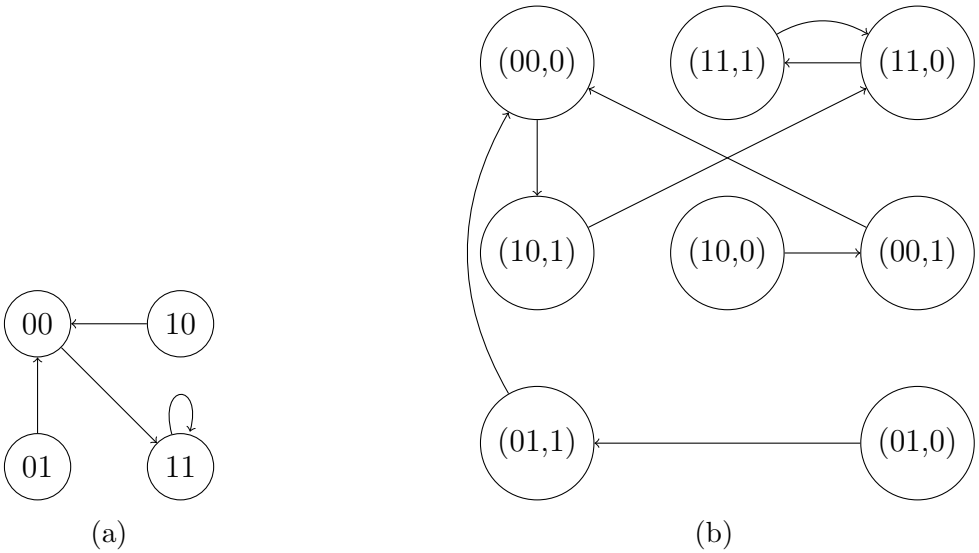


Figure 5.3: (a) STG of the BSBN shown in Example 5.3.1 and (b) ESTG of the encoded DGABN of this BSBN.

Definition 5.3.3. *Let $\mathcal{B} = \langle V, F, d\rangle$ be a BSBN. Then, $\mathcal{D}$ is a DGABN such that its set of nodes is V and its set of Boolean functions is F and its context is given as: $p_i = nb(d), q_i = j-1, x_i \in B_j, i \in \{1, ..., n\}$. Given a state s of $\mathcal{B}$, the corresponding extended state of s is $[[s]]^{\mathcal{D}}$, where $[[s]]^{\mathcal{D}}_i = s_i, i \in \{1, ..., n\}$ and $[[s]]^{\mathcal{D}}_{n+1} = 0$.*

Definition 5.3.3 gives the encoding of a BSBN as a DGABN. Figure 5.3b shows the ESTG of the encoded DGABN of the BSBN in Example 5.3.1. This ESTG has one limit cycle of length two ($\{(11, 0), (11, 1)\}$). As we can see, a state s reaches a state s' in the BSBN if and only if the corresponding extended state of s reaches the corresponding extended state of s' in the encoded DGABN. For example, 00 reaches 11 if and only if $(00, 0)$ reaches $(11, 0)$. We can also see that the set of attractors of the BSBN one-to-one corresponds to the set of attractors of the encoded DGABN. Hereafter, we formalize the relations in dynamics between a BSBN and its encoded DGABN (see Theorems 5.3.1 and 5.3.2). These relations allow us to apply our methods proposed for DGABNs to BSBNs.

Theorem 5.3.1. *Let $\mathcal{B} = \langle V, F, d\rangle$ be a BSBN and $\mathcal{D}$ be its encoded DGABN by Definition 5.3.3. For any pair of states s and s', we have s' is reachable from s in $\mathcal{B}$ if and only if $[[s']]^{\mathcal{D}}$ is reachable from $[[s]]^{\mathcal{D}}$ in $\mathcal{D}$.*

Proof. Assume that s' is the next state of s in $\mathcal{B}$, i.e., $s \xrightarrow{\mathcal{B}} s'$, where $\xrightarrow{\mathcal{B}}$ denotes a state transition in $\mathcal{B}$. Then, $s \xrightarrow{B_1} s^1...s^{m-1} \xrightarrow{B_m} s'$ where $m = nb(d)$ and B_i is the ith block of d and $s \xrightarrow{B_1} s^1$ denotes that s^1 is the state obtained by updating all nodes of B_1 in parallel with the current state is s. In $[[s]]^{\mathcal{D}}$, the nodes of B_1 will be updated because $[[s]]^{\mathcal{D}}_{n+1} = 0, [[s]]^{\mathcal{D}}_{n+1} \% p_i = 0 = q_i, x_i \in B_1$. Then, $[[s]]^{\mathcal{D}} \xrightarrow{\mathcal{D}} (s^1, 1)$ where $(s^1, 1)$ is an extended state of $\mathcal{D}$ with $t_{scaled} = 1$. Similarly, we have $[[s]]^{\mathcal{D}} \xrightarrow{\mathcal{D}} (s^1, 1)...(s^{m-1}, m-1) \xrightarrow{\mathcal{D}} [[s']]^{\mathcal{D}}$. This means that $[[s']]^{\mathcal{D}}$ is reachable from $[[s]]^{\mathcal{D}}$ in $\mathcal{D}$.

Assume that $[[s]]^{\mathcal{D}} \xrightarrow{\mathcal{D}} (s^1, 1)...(s^{m-1}, m-1) \xrightarrow{\mathcal{D}} [[s']]^{\mathcal{D}}$. Let V_1 be the set of nodes whose updates make $\mathcal{D}$ transits from $[[s]]^{\mathcal{D}}$ to $(s^1, 1)$. Then, $[[s]]^{\mathcal{D}}_{n+1} \% p_i = 0 = q_i, i \in V_1$. Thus, $V_1 = B_1$, implying that $s \xrightarrow{B_1} s^1$ in $\mathcal{B}$. Similarly, we have $s \xrightarrow{B_1} s^1...s^{m-1} \xrightarrow{B_m} s'$. This means that s' is reachable from s in $\mathcal{B}$.

Now, we can conclude the proof. □

Theorem 5.3.2. *Let $\mathcal{B} = \langle V, F, d\rangle$ be a BSBN and $\mathcal{D}$ be its encoded DGABN by Definition 5.3.3. Let $A^{\mathcal{B}}$ and $A^{\mathcal{D}}$ be the sets of attractors of $\mathcal{B}$ and $\mathcal{D}$, respectively. Then, $A^{\mathcal{B}}$ one-to-one corresponds to $A^{\mathcal{D}}$. In addition, given an attractor att of $\mathcal{B}$, its corresponding attractor of $\mathcal{D}$ is att' satisfying $att = [[att']]^{\mathcal{B}}$, where $[[ES]]^{\mathcal{B}} = \{s | (s, 0) \in ES\}$ with ES is a set of extended states.*

Proof. Assume that att is an attractor of $\mathcal{B}$. Let $FR^{\mathcal{B}}(S)$ denote the set of states reachable from the set S of states in $\mathcal{B}$. Let s be a state in att. Then, $FR^{\mathcal{B}}(\{s\}) = att$. Let $FR^{\mathcal{D}}(ES)$ denote the set of extended states reachable from the set ES of extended states in $\mathcal{D}$. Start from $[[s]]^{\mathcal{D}}$, following the evolution of $\mathcal{D}$, we will come back to $[[s]]^{\mathcal{D}}$ by Theorem 5.3.1. Since $\mathcal{D}$ has only fixed points or limit cycles, $FR^{\mathcal{D}}(\{[[s]]^{\mathcal{D}}\})$ is an attractor of $\mathcal{D}$. Moreover, $[[FR^{\mathcal{D}}(\{[[s]]^{\mathcal{D}}\})]]^{\mathcal{B}} = FR^{\mathcal{B}}(\{s\})$ by Theorem 5.3.1. Thus, $[[FR^{\mathcal{D}}(\{[[s]]^{\mathcal{D}}\})]]^{\mathcal{B}} = att$. (i)

Given $att_1, att_2 \in A^{\mathcal{B}}, att_1 \cap att_2 = \emptyset$. Consider an arbitrary pair of states $s^1 \in att_1$ and $s^2 \in att_2$. If $FR^{\mathcal{D}}(\{[[s^1]]^{\mathcal{D}}\}) \cap FR^{\mathcal{D}}(\{[[s^2]]^{\mathcal{D}}\}) \neq \emptyset$, then they are the same attractor

since $\mathcal{D}$ has only fixed points or limit cycles. This implies that $att_1 = att_2$ contradicting to $att_1 \cap att_2 = \emptyset$. Thus, $FR^{\mathcal{D}}(\{[[s^1]]^{\mathcal{D}}\}) \cap FR^{\mathcal{D}}(\{[[s^2]]^{\mathcal{D}}\}) = \emptyset$. (ii)

Assume that att' is an attractor of $\mathcal{D}$. Let att be a set of states in $\mathcal{B}$ such that $att = [[att']]^{\mathcal{B}}$. By Theorem 5.3.1, each state of att is reachable from any other state of att. Suppose that there exists a state $s \notin att$ such that it is reachable from att. Then $[[s]]^{\mathcal{D}}$ is not in att' but is reachable from att' by Theorem 5.3.1. This is a contradiction. Thus, att is an attractor of $\mathcal{B}$. (iii)

Given $att'_1, att'_2 \in A^{\mathcal{D}}, att'_1 \cap att'_2 = \emptyset$. Oviously, $[[att'_1]]^{\mathcal{B}} \cap [[att'_2]]^{\mathcal{B}} = \emptyset$. (iv)

There is an injection from $A^{\mathcal{B}}$ to $A^{\mathcal{D}}$ by (i) and (ii). There is also an injection from $A^{\mathcal{D}}$ to $A^{\mathcal{B}}$ by (iii) and (iv). Thus, $A^{\mathcal{B}}$ one-to-one corresponds to $A^{\mathcal{D}}$. In addition, $att = [[att']]^{\mathcal{B}}$ with att is an attractor of $\mathcal{B}$ and att' is its corresponding attractor of $\mathcal{D}$. We also obtain that $|att'| = nb(d) \times |att|$ because $\mathcal{D}$ has only fixed points or limit cycles. □

5.3.3 Relations to Generalized Asynchronous Boolean Networks

Generalized Asynchronous Boolean Networks (GABNs) are interesting mathematical objects and have been widely studied [14, 150, 157] besides ABNs. GABNs have the asynchronous and non-deterministic updating scheme. At each time step, they randomly select any number of nodes to update synchronously. This means that a GABN can update synchronously no node, only one node, some nodes, or all the nodes. The STG of a GABN has 2^n nodes and up to 2^{2n} arcs [14]. This maybe makes the analysis of a GABN computationally hard.

Example 5.3.2. *Let $\mathcal{G}$ be the GABN counterpart of the DGABN $\mathcal{D}$ shown in Example 5.2.1 (i.e., the GABN shares the sets of nodes and Boolean functions with $\mathcal{D}$). The STG of $\mathcal{G}$ is shown in Figure 5.4. As we can see, each arc of the ESTG of $\mathcal{D}$ (see Figure 5.2a) corresponds to an arc of the STG of $\mathcal{G}$. For example, $((00,0),(10,1))$ corresponds to $(00,10)$ and $((10,0),(01,1))$ corresponds to $(10,01)$. $\mathcal{G}$ has only one attractor $(\{11\})$, whereas $\mathcal{D}$ has three attractors. We can see that $\{11\}$ corresponds to the attractor $\{(11,1),(11,0)\}$ of $\mathcal{D}$.*

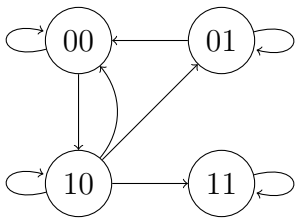


Figure 5.4: STG of the GABN shown in Example 5.3.2.

Obviously, all state transitions of a DGABN will be covered in the STG of its GABN counterpart by the updating scheme of GABNs. See Example 5.3.2 as an illustration. Hereafter, we formally state and prove several relations in dynamics between a DGABN and its GABN counterpart (see Lemma 5.3.1 and Theorem 5.3.3). Theorem 5.3.3 shows that given a DGABN and its GABN counterpart, each attractor of the GABN contains at least one attractor of the DGABN.

Lemma 5.3.1. *Let $\mathcal{D}$ be a DGABN and $\mathcal{G}$ be its GABN counterpart. Let es be an extended state of $\mathcal{D}$ and $FR^{\mathcal{D}}(\{es\})$ be the set of extended states reachable from es in $\mathcal{D}$. Then, $[[FR^{\mathcal{D}}(\{es\})]]^I \subseteq FR^{\mathcal{G}}(\{[[es]]^I\})$.*

Proof. Let $G(\mathcal{D})$ and $G(\mathcal{G})$ be the ESTG and STG of $\mathcal{D}$ and $\mathcal{G}$, respectively. If (es, es') is an arc in $G(\mathcal{D})$, then $([[es]]^I, [[es']]^I)$ is also an arc in $G(\mathcal{G})$ because in GABNs, any number of nodes can be synchronously updated. Thus, it is easy to imply that $[[FR^{\mathcal{D}}(\{es\})]]^I \subseteq FR^{\mathcal{G}}(\{[[es]]^I\})$. □

Theorem 5.3.3. *Let $\mathcal{D}$ be a DGABN and $\mathcal{G}$ be its GABN counterpart. Let $A^{\mathcal{D}}$ and $A^{\mathcal{G}}$ be the sets of attractors of $\mathcal{D}$ and $\mathcal{G}$, respectively. Then, there exists a mapping $m : A^{\mathcal{G}} \to A^{\mathcal{D}}$ with $[[m(att)]]^I \subseteq att$ for all $att \in A^{\mathcal{G}}$. In addition, $m(att_1) \neq m(att_2)$ for all $att_1, att_2 \in A^{\mathcal{G}}, att_1 \neq att_2$. This means that m is an injection.*

Proof. Let $att \in A^{\mathcal{G}}$ and $s \in att$ be a state of $\mathcal{G}$. Obviously, $FR^{\mathcal{G}}(\{s\}) = att$.

Let es' be an extended state of $\mathcal{D}$ such that $es'_i = s_i, i \in \{1, ..., n\}$ and $es'_{n+1} = 0$. By Lemma 5.3.1, we have $[[FR^{\mathcal{D}}(\{es'\})]]^I \subseteq FR^{\mathcal{G}}(\{[[es']]^I\}) = FR^{\mathcal{G}}(\{s\}) = att$. Clearly, there is an attractor att' of $\mathcal{D}$ such that $att' \subseteq FR^{\mathcal{D}}(\{es'\})$. Then, $[[att']]^I \subseteq [[FR^{\mathcal{D}}(\{es'\})]]^I$ be the definition of images. Thus, $[[att']]^I \subseteq att$. We now can choose the mapping m as $m(att) = att'$. Note that since att' may not be unique, m may not be uniquely determined.

For all $att_1, att_2 \in A^{\mathcal{G}}, att_1 \neq att_2$, we have $att_1 \cap att_2 = \emptyset$ since attractors of $\mathcal{G}$ are pairwise disjoint. $[[m(att_1)]]^I \subseteq att_1$ and $[[m(att_2)]]^I \subseteq att_2$ imply that $[[m(att_1)]]^I \cap [[m(att_2)]]^I = \emptyset$. Then, $m(att_1) \cap m(att_2) = \emptyset$ by the definition of images. Thus, $m(att_1) \neq m(att_2)$. Therefore, m is an injection. □

In Section 3.3, we have formally stated and proved several relations in dynamics between a GABN and its SBN counterpart. Theorem 3.3.1 shows that any attractor of a GABN always contains an attractor of its SBN counterpart. Since an SBN is a special DGABN, this theorem is a special case of Theorem 5.3.3.

5.3.4 Relations to Mixed-Context Boolean Networks

To deal with the lack of knowledge on real contexts of DGABNs, Carlos Gershenson proposed a new type of Boolean models called Mixed-Context Boolean Networks (MxBNs) [5, 13]. He introduced the idea of *mixed context*, where a mixed context is a statistical mixture of a set of pure contexts. The updating scheme of MxBNs is basically like that of DGABNs. However, MxBNs have to make a random choice between pure contexts at each time step, making their dynamics non-deterministic and generating a probability structure that is non-Kolmogorovian (quantum-like) [13].

An MxBN is a DGABN with $M \geq 1$ *pure contexts*. Each pure context consists of p's and q's as in DGABNs. At each time step, we randomly select one pure context among M pure contexts and use it in the DGABN. Hence, the dynamics of an MxBN can be captured by overlapping the ESTGs of its M constituent DGABNs. See Example 5.3.3 for an example of MxBNs.

Example 5.3.3. *Given an MxBN $\mathcal{M}$ of two nodes (x_1, x_2). Its Boolean functions and $M = 2$ pure contexts are given by*

$$f_1 = (x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2), f_2 = x_1;$$
$$M_1 : p_1 = 1, p_2 = 2, q_1 = 0, q_2 = 0;$$
$$M_2 : p_1 = 1, p_2 = 1, q_1 = 0, q_2 = 0.$$

Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be the DGABNs corresponding to M_1 and M_2, respectively. Figure 5.5 shows the ESTG of $\mathcal{M}$. This ESTG is formed by overlapping the ESTG of $\mathcal{D}_1$ (normal lines) and the ESTG of $\mathcal{D}_2$ (dashed lines).

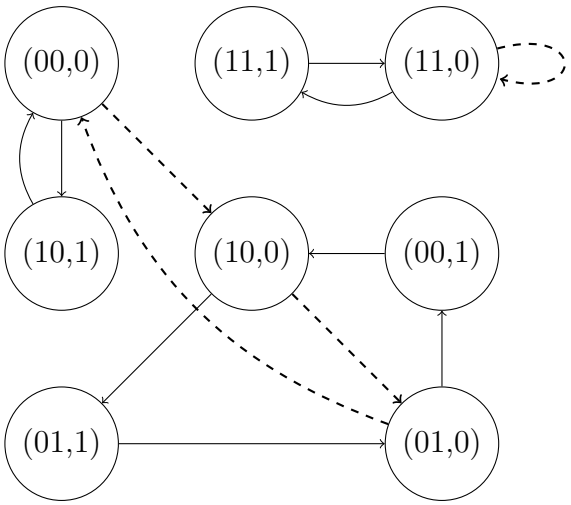


Figure 5.5: ESTG of the MxBN shown in Example 5.3.3.

In Example 5.3.3, $\mathcal{M}$ has two attractors including {(11,0), (11,1)} and {(00,0), (10,1), (10,0), (01,1), (01,0), (00,1)}. Note that an extended state of an MxBN may have more than one outgoing transition by the introduction of non-determinism to MxBNs, leading to the existence of complex attractors as in ABNs [43]. In addition, $\mathcal{D}_1$ has three attractors including {(00,0), (10,1)}, {(11,0), (11,1)}, and {(10,0), (01,1), (01,0), (00,1)}; $\mathcal{D}_2$ has two attractors including $\{(11,0)\}$ and {(00,0), (10,0), (01,0)}. We can see that each attractor of $\mathcal{M}$ always contains at least one attractor of $\mathcal{D}_1$. This observation is also valid for the case of $\mathcal{D}_2$. We generalize this observation in Theorem 5.3.4.

Theorem 5.3.4. *Let $\mathcal{M}$ be an MxBN of M pure contexts. Let $\mathcal{D}$ be an arbitrary DGABN among M constituent DGABNs of $\mathcal{M}$. Let $\mathcal{A}^{\mathcal{M}}$ and $\mathcal{A}^{\mathcal{D}}$ be the sets of attractors of $\mathcal{M}$ and $\mathcal{D}$, respectively. Then, there exists a mapping $m : \mathcal{A}^{\mathcal{M}} \to \mathcal{A}^{\mathcal{D}}$ with $m(att) \subseteq att$ for all $att \in \mathcal{A}^{\mathcal{M}}$. In addition, $m(att_1) \neq m(att_2)$ for all $att_1, att_2 \in \mathcal{A}^{\mathcal{M}}, att_1 \neq att_2$. This means that m is an injection.*

Proof. Let $att \in \mathcal{A}^{\mathcal{M}}$. Obviously, an attractor of $\mathcal{M}$ always contains an extended state es such that $es_{n+1} = 0$. Thus, there is an extended state es such that $es \in att$ and $es_{n+1} = 0$. Then, $FR^{\mathcal{M}}(\{es\}) = att$. In addition, es is also an extended state of $\mathcal{D}$.

Since the ESTG of $\mathcal{M}$ contains the ESTG of $\mathcal{D}$, $FR^{\mathcal{D}}(\{es\}) \subseteq FR^{\mathcal{M}}(\{es\})$. Obviously, there is an attractor att' of $\mathcal{D}$ such that $att' \subseteq FR^{\mathcal{D}}(\{es\})$. Then, $att' \subseteq FR^{\mathcal{M}}(\{es\}) = att$. We now can choose the mapping m as $m(att) = att'$. Note that since att' may not be unique, m may not be uniquely determined.

For all $att_1, att_2 \in \mathcal{A}^{\mathcal{M}}, att_1 \neq att_2$, we have $att_1 \cap att_2 = \emptyset$ because attractors of $\mathcal{M}$ are pairwise disjoint. $m(att_1) \subseteq att_1$ and $m(att_1) \subseteq att_1$ imply that $m(att_1) \cap m(att_2) = \emptyset$. Thus, $m(att_1) \neq m(att_2)$. Therefore, m is an injection. □

Theorem 5.3.4 suggests us a promising way to find all attractors of an MxBN. First, we can find attractors of one of its constituent DGABNs by applying our method presented in

Section 5.4. Then, we can filter the set of DGABN attractors by checking the reachability property in this MxBN. This idea is similar to the filtering algorithm for finding GABN attractors based on SBN attractors, which is presented in Subsection 3.4.3. Proposing an efficient method for attractor detection in MxBNs is one of our future work.

5.4 Computing Attractors

Attractor detection in various types of BNs has attracted much attention. Many studies has been done but they mainly focus on SBNs (e.g., [43, 44, 50]) and ABNs (e.g., [43, 49]). There are very few studies (e.g., [75, 80]) specifically done for DGABNs. Moreover, these few studies are theoretical or simulation-based studies. These facts motivate algorithms for analytically and practically finding attractors of a DGABN.

Dubrova and Teslenko proposed an efficient SAT-based method for finding attractors of an SBN [44]. Since the ESTG of a DGABN is deterministic like the STG of a SBN, there is a potential to extend the SAT-based method for SBNs to that for DGABNs. However, it is difficult to directly use SAT for DGABNs because the transition formula of an ESTG (see Equation 5.1) contains integer variables (e.g., x^j_{n+1}) and the modulus operator. Since integers are bounded, we can encode x^j_{n+1} by a bit-vector [118] of size m (m is the smallest integer larger than or equal to the logarithm to the base two of γ) and then use the bit-blasting technique [159] to transform the transition formula to an equivalent SAT formula. Satisfiability Modulo Theory (SMT) may provide a more natural encoding. Modern SMT solvers (e.g., Z3 [118]) use the efficient algorithms of SAT solvers as their solving cores, thus can handle very large problem instances. In addition, the bit-blasting technique is also integrated into some SMT solvers as a solving tactic [118]. Therefore, we here propose a new SMT-based method for finding attractors of an DGABN. Note that proposing ESTG paves the way to extend the method by Dubrova and Teslenko for SBNs to that for DGABNs.

5.4.1 SMT-Based Method

The intuitive idea of the proposed SMT-based method (named **DA-SMT-Att**) is as follows. **DA-SMT-Att** searches for a p-length path (i.e., this path makes p transitions between extended states) in the ESTG of a DGABN. Since a fixed point of the DGABN is represented by a limit cycle of length γ in the ESTG of this DGABN, p can start with γ ($\gamma \geq 1$). If a path is found and it contains a limit cycle, **DA-SMT-Att** adds this limit cycle to the set of marked attractors. In an ESTG, an extended state has a unique next extended state, thus once a path reaches a limit cycle, it never leaves this cycle. This suggests a way to check whether a path contains a limit cycle by checking whether the last extended state of the path occurs at least twice in this path. In the next iterations, **DA-SMT-Att** only searches for paths such that their last extended states are not in the set of marked attractors. If a path is found and it contains no limit cycle, **DA-SMT-Att** increases p (e.g., doubles p) and continues the search for a path with the new length. If a path does not exist, **DA-SMT-Att** can terminate the search. Since the ESTG is deterministic like the STG of an SBN, **DA-SMT-Att** always terminates and correctly finds all attractors of the DGABN (see Theorem 5.4.1).

Theorem 5.4.1. ***DA-SMT-Att*** *terminates and correctly finds all attractors of a DGABN.*

Proof. Let $\mathcal{D}$ denote the DGABN.

Until at least one attractor remains unmarked, we can find a path of any length such that its last extended state is not in any marked attractors because we can cycle in an attractor forever. This means that **DA-SMT-Att** will not terminate if at least one attractor remains unmarked (a).

In the beginning of **DA-SMT-Att**, the number of marked attractors is 0. The length of a path such that it does not contain any limit cycles must be less than the number of extended states of the ESTG of $\mathcal{D}$ (b). A better upper bound is the *diameter* of the ESTG. The diameter of a graph is defined as the length of the longest shortest path between two nodes. Since p always increases when no limit cycle is found and the ESTG has at least one limit cycle, we eventually find a path containing a limit cycle by (a). Now this limit cycle is marked (i.e., it is added to the set of marked attractors). If all attractors of $\mathcal{D}$ are marked, **DA-SMT-Att** must eventually terminate by (b). Otherwise, $\mathcal{M}$ continues its search by (a). Since the number of attractors of $\mathcal{D}$ is finite, all attractors will eventually be marked. Now, **DA-SMT-Att** can terminate by (b) and all attractors of $\mathcal{D}$ are found. □

In each iteration of **DA-SMT-Att**, the finding of a p-length path can be performed by using an SMT solver (we here use Z3 [118]). Let A be the set of attractors that is updated through the iterations. Note that such a path must satisfy two conditions. First, the value of the scaled time of its starting extended state must be 0. As mentioned in Section 5.2, there may be some spurious extended states that are not in the ESTG of the DGABN. This condition guarantees that all extended states along with the path are always in the ESTG because the starting extended state is one of the possible initial extended states of the DGABN. Second, its ending extended state must be not in A as mentioned in the previous paragraph. The path can be encoded as an SMT formula P (see Equation 5.2). Then, we simply use Z3 to solve P.

$$P := (es^0_{n+1} = 0) \wedge \left\{ \bigwedge_{j=0}^{p-1} \mathcal{T}^{\mathcal{D}}(es^j, es^{j+1}) \right\} \wedge \neg\chi(A^F, es^p) \tag{5.2}$$

In Equation 5.2, es^j denotes the $(j+1)$th extended state of the path. Then, es^0 and es^p denote the starting and ending extended states of the p-length path, respectively. $\mathcal{T}^{\mathcal{D}}(es^j, es^{j+1})$ represents the transition from es^j to es^{j+1}, where $\mathcal{T}^{\mathcal{D}}$ is the transition formula of the DGABN $\mathcal{D}$ (see Equation 5.1). Thus, $\bigwedge_{j=0}^{p-1} \mathcal{T}^{\mathcal{D}}(es^j, es^{j+1})$ represents a p-length path of $\mathcal{D}$. $(es^0_{n+1} = 0)$ represents the first condition of the path. $\neg\chi(A^F, es^p)$ represents the second condition of the path, where A^F is the flattened set of A (i.e., A^F is the set of extended states); $\chi(A^F, es^p)$ is the characteristic formula representing all extended states of A^F in terms of variables of es^p. The characteristic formula of a set of extended states is defined based on the characteristic formula of an extended state: $\chi(A^F, es^p) = \bigvee_{es \in A^F} \chi(es, es^p)$. The characteristic formula of an extended state es in terms of variables of es^p is defined as $\chi(es, es^p) = \bigwedge_{i=1}^{n+1}(es_i = es^p_i)$.

Let us see a running example of **DA-SMT-Att** on the DGABN $\mathcal{D}$ shown in Exam-

ple 5.2.1. **DA-SMT-Att** starts with $p = \gamma = 2$ and $A = \emptyset$. We obtain

$$\begin{aligned}
&\mathcal{T}^{\mathcal{D}}(es^0, es^1) = \{es_3^1 = (es_3^0 + 1)\%2\} \wedge \{(es_3^0\%1 = 0 \wedge \\
&(es_1^1 = (es_1^0 \wedge es_2^0) \vee (\neg es_1^0 \wedge \neg es_2^0))) \vee (es_3^0\%1 \neq 0 \wedge es_1^1 = es_1^0)\} \\
&\wedge \{(es_3^0\%2 = 0 \wedge es_2^1 = es_1^0) \vee (es_3^0\%2 \neq 0 \wedge es_2^1 = es_2^0)\}; \\
&\mathcal{T}^{\mathcal{D}}(es^1, es^2) = \{es_3^2 = (es_3^1 + 1)\%2\} \wedge \{(es_3^1\%1 = 0 \wedge \\
&(es_1^2 = (es_1^1 \wedge x_2^1) \vee (\neg es_1^1 \wedge \neg es_2^1))) \vee (es_3^1\%1 \neq 0 \wedge es_1^2 = es_1^1)\} \\
&\wedge \{(es_3^1\%2 = 0 \wedge es_2^2 = x_1^1) \vee (es_3^1\%2 \neq 0 \wedge es_2^2 = es_2^1)\}; \\
&P = (es_3^0 = 0) \wedge \mathcal{T}^{\mathcal{D}}(es^0, es^1) \wedge \mathcal{T}^{\mathcal{D}}(es^1, es^2).
\end{aligned}$$

DA-SMT-Att then uses Z3 to solve P. Clearly, a path is found. Suppose that this path is $(00, 0) \to (10, 1) \to (00, 0)$ (see Figure 5.2a). Since the last extended state $((00, 0))$ occurs twice, we obtain a limit cycle of length 2. Now, we can add this limit cycle $(\{(00, 0), (10, 1)\})$ to A. The flatted set A^F is $\{(00, 0), (10, 1)\}$ and $\chi(A^F, es^2) = \chi((00, 0), es^2) \vee \chi((10, 1), es^2)$ where $\chi((00, 0), es^2) = (es_1^2 = 0) \wedge (es_2^2 = 0) \wedge (es_3^2 = 0)$, $\chi((10, 1), es^2) = (es_1^2 = 1) \wedge (es_2^2 = 0) \wedge (es_3^2 = 1)$. In the next iteration, **DA-SMT-Att** continues to search a path of length two with the new formula $P = (es_3^0 = 0) \wedge \mathcal{T}^{\mathcal{D}}(es^0, es^1) \wedge \mathcal{T}^{\mathcal{D}}(es^1, es^2) \wedge \neg\chi(A^F, es^2)$. Suppose that the next found path is $(00, 1) \to (10, 0) \to (01, 1)$. This path is cycle-free because the last extended state $((01, 1))$ occurs only one. Now, **DA-SMT-Att** increases p to four and starts the next iteration with $p = 4$ and $A = \{\{(00, 0), (10, 1)\}\}$. By running two more iterations, **DA-SMT-Att** terminates and all the attractors of $\mathcal{D}$ are detected with $A = \{\{(00,0), (10,1)\}, \{(11,0), (11,1)\}, \{(00,1), (10,0), (01,1), (01,0)\}\}$.

In **DA-SMT-Att**, the last value of p is called the unfolding depth of the DGABN. Obviously, in each iteration of **DA-SMT-Att**, the number variables and the number of clauses of P depend on both n and p. In the ESTG of the DGABN, starting from an extended state, we must go through at least $\gamma - 1$ different extended states to reach an extended state with the same t_{scaled}. It seems to make the diameter of the ESTG longer. Thus, if γ is large even when n is small, the diameter may be very large. The diameter of the ESTG is an upper bound of the unfolding depth, leading the unfolding depth of the DGABN may be very large. In this case, P will have too many variables and clauses, and the time for solving P may be extremely long.

5.4.2 Case Study

In this subsection, we apply our proposed method for attractor detection in DGABNs (i.e., **DA-SMT-Att**) to two real biological networks and compare the obtained results to the previous insights on these networks already explored in the literature.

The first network is the reduced network of the guard cell ABA signal transduction network analyzed in [9]. The BN of this reduced network includes three nodes: x_1 standing for CIS, x_2 standing for Ca_c^{2+}, and x_3 standing for Ca^{2+}ATPase. Its Boolean functions are given by

$$f_1 = x_2, f_2 = x_1 \wedge \neg x_3, f_3 = x_2.$$

Each node x_i is associated with a time unit γ_i, forming a DA model. Saadatpour et al. [9] applied their method to this network with many different choices of time units (i.e., many

different DA models). We here encode a DA model as a DGABN (see Subsection 5.3.1). Then, we apply **DA-SMT-Att** to find attractors of the encoded DGABN. The obtained results are given as follows.

For the case that $\gamma_1 = 2\gamma_2, \gamma_2 = 2k+1, \gamma_3 = 2, k = 2$, we have that the DA model has a limit cycle of length $\gamma_2 + 1 + 2k = 4k + 2 = 10$ by Proposition 1 of [9]. Note that in [9], the authors claimed that the cycle length is only $2k+2$ because of their way for counting the time staying each state. Following their proof for this proposition, the correct length should be $4k+2$. By applying **DA-SMT-Att**, the encoded DGABN has two limit cycles of length 10. One of these two limit cycles is {(101,0), (111,9), (111,8), (111,7), (111,6), (110,5), (100,4), (100,3), (100,2), (101,1)}. This result is consistent with Proposition 1 of [9].

For the case that $\gamma_1 = 2\gamma_2, \gamma_2 = 2k, \gamma_3 = 2, k = 2$, we have that the DA model has a limit cycle of length $\gamma_2 + 2k = 4k = 8$ by Proposition 2 of [9]. Note that in [9], the authors claimed that the cycle length is only $2k$ because of their way for counting the time staying each state. Following their proof for this proposition, the correct length should be $4k$. By applying **DA-SMT-Att**, the encoded DGABN has two limit cycles of length 8. One of these two limit cycles is {(110,5), (110,4), (100,3), (100,2), (101,1), (101,0), (111,7), (111,6)}. This result is consistent with Proposition 2 of [9].

The second network is the mammalian cell cycle network analyzed in [82]. The BN of this network includes 10 nodes (genes) and its detail is given in Table 5.1. The authors of [82] analyzed this network under different deterministic updating schemes. For each deterministic updating scheme, we encode the corresponding BSBN as a DGABN (see Subsection 5.3.2). Then, we apply **DA-SMT-Att** to find attractors of the encoded DGABN. The obtained results are given as follows.

Table 5.1: BN model of the cell cycle network.

Gene	Boolean function
$CycD$	$CycD$
Rb	$(\neg CycD \wedge \neg CycB) \wedge ((\neg CycE \wedge \neg CycA) \vee p27)$
$E2F$	$(\neg Rb \wedge \neg CycA \wedge \neg CycB) \vee (p27 \wedge \neg Rb \wedge \neg CycB)$
$CycE$	$(E2F \wedge \neg Rb)$
$CycA$	$(\neg Rb \wedge \neg Cdc20 \wedge \neg(Cdh1 \wedge UbcH10)) \wedge (E2F \vee CycA)$
$p27$	$(\neg CycD \wedge \neg CycB) \wedge ((\neg CycE \wedge \neg CycA) \vee (p27 \wedge \neg(CycE \wedge CycA)))$
$Cdc20$	$CycB$
$Cdh1$	$(\neg CycA \wedge \neg CycB) \vee Cdc20 \vee (p27 \wedge \neg CycB)$
$UbcH10$	$\neg Cdh1 \vee (Cdh1 \wedge UbcH10 \wedge (Cdc20 \vee CycA \vee CycB))$
$CycB$	$\neg Cdc20 \wedge \neg Cdh1$

For the case that the deterministic updating scheme is $(CycD, Rb, Cdc20, Cdh1, CycA)(p27, UbcH10, CycB)(E2F)(CycE)$, the BSBN has one fixed point with $CycD = 0$, a limit cycle of length 4 with $CycD = 1$, and a limit cycle of length 8 with $CycD = 0$ (see Figure 7 of [82]). By applying **DA-SMT-Att**, the encoded DGABN has a limit cycle of length 4, a limit cycle of length 16, and a limit cycle of length 32. Since γ of the encoded DGABN is 4, three attractors of the encoded DGABN correspond to three attractors of the BSBN. See Table 5.2 for details of these DGABN attractors. Herein, the order of the nodes in a state is $(CycD, p27, E2F, CycE, CycA, p27, Cdc20, Cdh1, UbcH10, CycB)$. Note that in

each DGABN attractor, we only show the images of extended states whose t_{scaled} are 0. This result is consistent with the relations presented in Subsection 5.3.2.

For the case that the deterministic updating scheme is $(CycD, p27, Cdc20, Cdh1, UbcH10, CycB)(E2F)(CycE)(Rb, CycA)$, the BSBN has one fixed point with $CycD = 0$, a limit cycle of length 2 with $CycD = 0$, a limit cycle of length 6 with $CycD = 0$, and a limit cycle of length 6 with $CycD = 1$ (see Figure 8 of [82]). By applying **DA-SMT-Att**, the encoded DGABN has a limit cycle of length 4, a limit cycle of length 8, and two limit cycles of length 24. Since γ of the encoded DGABN is 4, four attractors of the encoded DGABN correspond to four attractors of the BSBN. See Table 5.2 for details of these DGABN attractors. This result is consistent with the relations presented in Subsection 5.3.2.

Table 5.2: Details of DGABN attractors of the cell cycle network.

Figure 7	{0100010100}
	{1011000100, 1000101010, 1000100011, 1000100100}
	{0000100100, 0011000100, 0011001010, 0000010011, 0100100100, 0011010100, 0000101010, 0000100011}
Figure 8	{0100010100}
	{0111110100, 0000000100}
	{0000100011, 0000100000, 0011100100, 0011000110, 0011001110, 0000001011}
	{1011001110, 1000001011, 1000100011, 1000100000, 1011100100, 1011000110}

5.4.3 Verifying the Previous Insights

Many numerical experiments were conducted to discover several insights into the dynamics of DGABNs [13, 14]. However, their method is incomplete and is limited to small networks ($n \leq 10$). Since **DA-SMT-Att** can find exactly all the attractors of a DGABN and can be applicable to large networks (see Section 5.5), we here reproduced these experiments with larger networks to verify these insights. We hope that **DA-SMT-Att** will be helpful in further research on DGABNs and their applications.

Following the experimental method by [13, 14], we randomly generated 1000 N-K BNs with $K = 3$ (i.e., each node has exactly $K = 3$ input nodes) and different numbers of nodes ($n = 3, 4, ..., 15$) by using **Bool Net R** package [117]. We then randomly generated a context for each BN with $\Lambda = 3$. In total, we have 13000 randomly generated DGABNs. We applied **DA-SMT-Att** to find attractors of each DGABN and its SBN counterpart (i.e., $\Lambda = 1$). For each DGABN or its SBN counterpart, the number of attractors and the percentage of extended states in attractors were reported.

Figure 5.6 shows the average number of attractors of the randomly generated networks with different n. We can see that the average number of attractors of DGABNs for low Λ ($\Lambda = 1$ and $\Lambda = 3$) has a linear increment proportional to n. This observation is consistent with the insight on the average number of attractors presented in Section 3 of [13].

Figure 5.7 shows the percentage of attractor states (in a base-10 logarithmic scale) of the randomly generated networks with different n. By observing this figure, we can see that the percentage of attractor states seems to decrease exponentially as n increases for

both SBNs and DGABNs. However, the percentage of attractor states decreases slower for low Λ ($\Lambda = 1$) than for a higher one ($\Lambda = 3$). This observation is consistent with the insight on the percentage of attractor states presented in Section 3 of [13]. Another observation, which can be obtained from this figure, is that SBNs ($\Lambda = 1$) have more states in attractors than DGABNs ($\Lambda = 3$). This is consistent with the insight on the percentage of attractor states presented in Subsection 3.2 of [14].

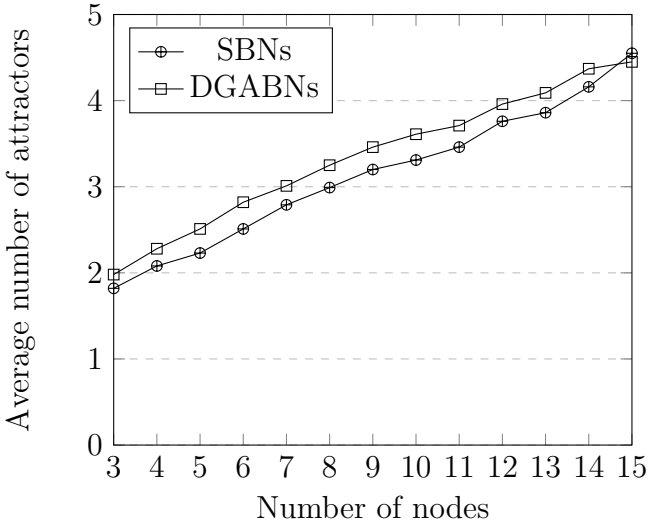


Figure 5.6: Average number of attractors varying the number of nodes.

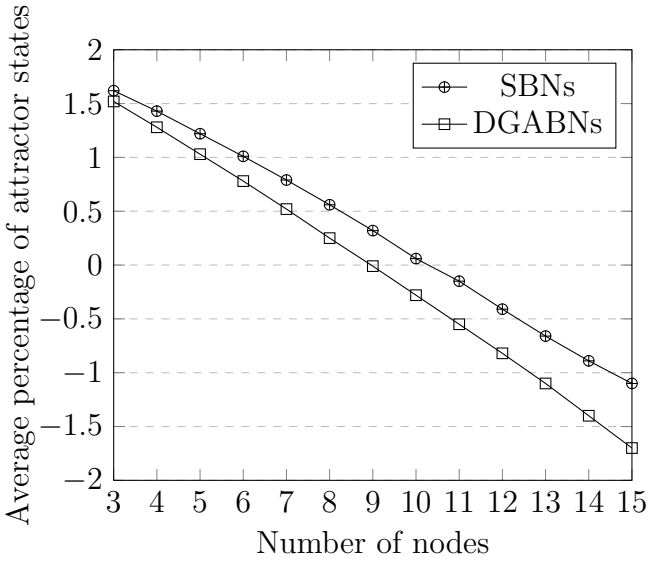


Figure 5.7: Average percentage (log scale) of attractor states varying the number of nodes.

5.5 Experimental Results

We have implemented the proposed method (i.e., **DA-SMT-Att**) in a JAVA tool integrating the Z3 solver called **DABoolNet**. An executable file of **DABoolNet** and several

example networks are available at: https://github.com/giang-trinh/daboolnet. To our best knowledge, **BooleanNet** [80] is the sole practical tool for analysis of DA models. However, a DA model is only a special DGABN (see Subsection 5.3.1) and **BooleanNet** is only a simulation tool. Thus, it is difficult to compare **DABoolNet** with other previous tools with respect to attractor detection in DGABNs. We here only focus on how well **DA-SMT-Att** scales up.

In order to evaluate the scalability of **DA-SMT-Att**, we applied this method to randomly generated DGABNs. We randomly generated 24 DGABNs with different numbers of nodes and $\Lambda = 3$. All the 24 DGABNs have the same $\gamma = 6$. We then ran **DABoolNet** to find attractors of these DGABNs. The time limit was set to four hours for each DGABN. The running environment is a PC with CPU: Intel Core i7 2.4 GHz, Memory: 16 GB, Windows 10 Home 64 bit.

Table 5.3 shows the obtained results. Column "n" stands for the number of nodes. Column "$a \times l$" stands for the number and length of attractors computed by **DA-SMT-Att**. The computational time (in seconds) is given in Column "time (s)". From these results, we see that **DA-SMT-Att** can find attractors of large DGABNs within practical computational time.

Table 5.3: Experimental results of **DA-SMT-Att** on randomly generated networks.

n	$a \times l$	time (s)	n	$a \times l$	time (s)
10	2 x 6	2.40	130	16 x 12	2312.64
20	8 x 6	14.49	140	14 x 12	342.61
30	10 x 6	49.37	150	8 x 6, 8 x 12, 4 x 36	711.71
40	16 x 6, 8 x 12	124.72	160	24 x 12	11996.69
50	10 x 12, 2 x 24, 4 x 36	529.95	170	6 x 12	1135.65
60	8 x 24	318.25	180	2 x 12	132.50
70	5 x 12	210.00	190	16 x 12	657.86
80	9 x 12, 2 x 24	324.91	200	4 x 6, 8 x 18, 2 x 36, 2 x 72	2902.37
90	8 x 6	304.19	250	10 x 6, 12 x 30	11338.33
100	4 x 6	136.90	300	5 x 6, 1 x 12, 12 x 30	8290.07
110	3 x 6, 1 x 12	88.64	350	timeout	timeout
120	24 x 12	820.93	400	timeout	timeout

5.6 Discussion

In this chapter, we have proposed the formulation of an ESTG. An ESTG captures the whole dynamics of a DGABN and paves potential ways to analyze this DGABN. Based on this formulation, we have proposed an SMT-based method (called **DA-SMT-Att**) for attractor detection in DGABNs, which is one of central issues in systems biology. The experimental results show that **DA-SMT-Att** can handle well large networks. We have also stated and proved several relations between DGABNs and other models including DA models, BSBNs, GABNs, and MxBNs. These relations not only contribute to understanding the dynamics of Boolean networks but also pave potential ways to analyze these models based on DGABNs.

To show the applications of **DA-SMT-Att**, we have applied this method to find attractors of two real biological networks. The results obtained by **DA-SMT-Att** are consistent with the previous insights on these real networks in the literature. We have also used **DA-SMT-Att** to verify several previously numerical insights into the dynamics of SBNs and DGABNs.

Note that, in our experiment, **DA-SMT-Att** suffers from an inherent problem of SMT. When γ of a DGABN is large (e.g, $\gamma \geq 30$), the unfolding depth in **DA-SMT-Att** may be too large even for DGABNs with small n. In this case, the SMT formula P will have too many variables and clauses, and the time for solving P may be extremely long. To mitigate this problem, further improvements for SMT (e.g., variable ordering) are needed.

Part II

Optimal Control

Chapter 6

Optimal Control of Deterministic Generalized Asynchronous Boolean Networks

6.1 Introduction

Control of biological systems is one of the central issues in systems biology [38]. In theory, biological systems are complex and contain highly non-linear components and thus existing methods in control theory cannot be directly applied to control of biological systems. In practice, control of cells may be useful for systems-based drug discovery and cancer treatment [19, 38, 39]. Thus, it is an important and interesting challenge to develop theories and methods for control of biological systems [21]. Since BNs are highly non-linear systems and have been widely used in modeling biological systems, it is reasonable to try to develop methods for control of BNs.

In recent years, several approaches have been developed for control of BNs. We can classify them into two main directions. The first one (e.g., [39, 58, 59, 60, 61, 62, 63]) uses node perturbations, whereas the second one (e.g., [55, 56, 57]) uses external inputs to control a BN. The second direction assumes that the set of control inputs is known and fixed for a finite sequence of steps. Somehow this is not very realistic, concerning the effort in searching for potential drug targets [21]. However, this direction is still useful because (a) extensive studies have been done on selecting and analyzing the minimum set of control nodes [160, 161] and (b) some methods of the first direction can be cast into the second direction [59]. To our best knowledge, there is no study specifically designed for DGABNs. Although optimality may rarely be the main constraint compared to correct behavior of systems in experiments, the optimal control problem is still useful and interesting because it is more general than the standard control problem [21, 56]. Therefore, we focus on optimal control of DGABNs.

We formulate the optimal control problem of DGABNs in Section 6.2. We then propose two SMT-based methods for solving the formulated problem under the two control modes, time-sensitive (see Section 6.3) and non-time-sensitive (see Section 6.4). To show the applications of the proposed methods, we apply the proposed methods to optimal control of a real biological network (see Section 6.5), the apoptosis network [162]. Finally, we evaluate the scalability of the proposed methods by conducting an experiment on an artificial network (see Section 6.6).

6.2 Problem Formulation

Before defining the problem of optimal control of DGABNs, we introduce DGABNs with control nodes. In the rest of this chapter, we refer a DGABN as a DGABN with control nodes.

Definition 6.2.1. *A BN with control nodes is defined as a triple (V, F, U), where $V = \{x_1, ..., x_n\}$ $(n \geq 1)$ is the set of internal nodes, $F = \{f_1, ..., f_n\}$ is the set of Boolean functions, and $U = \{u_1, ..., u_m\}$ $(m \geq 0)$ is the set of external (control) nodes. Each node x_i is identified as a Boolean variable, and is associated with a Boolean function $f_i : \mathbb{B}^n \times \mathbb{B}^m \to \mathbb{B}$. Also, each node u_i is identified as a Boolean variable. $x_i(t) \in \mathbb{B}$ and $u_i(t) \in \mathbb{B}$ denote the state of internal node x_i and the state of external node u_i at time t, respectively. $x(t) = (x_1(t), ..., x_n(t))^\top$ and $u(t) = (u_1(t), ..., u_m(t))^\top$ denote the state and the control input of the BN at time t, respectively.*

Definition 6.2.1 gives the definition of a BN with control nodes. Internal node x_i updates its state by $x_i(t+1) = f_i(x(t), u(t))$, whereas a control node can receive an arbitrary Boolean value at each time step. Then, a DGABN with control nodes is defined in Definition 6.2.2.

Definition 6.2.2. *A DGABN with control nodes is a BN with control nodes such that each internal node x_i is associated with two parameters, $p_i \in \mathbb{N}^+$ and $q_i \in \mathbb{N}, q_i < p_i$. Internal node x_i can be updated at time t if $t\%p_i = q_i$. If multiple internal nodes can be updated, then all of them are updated simultaneously.*

We here formally define optimal control of DGABNs as in Definition 6.2.3 adapted from [21]. In this definition, internal nodes stand for usual nodes (i.e., genes or proteins), external (control) nodes can stand for external interventions (e.g., drugs, radiation, or chemotherapy), the initial state can stand for a disease or cancerous state, and the desired state can stand for a healthy or normal state. Usually, $u_i(k) = 0$ implies that u_i is not applied at time k, whereas $u_i(k) = 1$ implies that u_i is applied at time k with the application cost g_i. Note that the cost vector g may be fixed or not fixed over time. In the biological context, g may depend on the current state of the system, i.e., g may be represented as a function $g_f : U \times \mathbb{B}^n \to \mathbb{N}$. In addition, we can consider various types of cost, such as, the cost of applying drugs, the total harmful effects of the applied drugs, the total concentration of some given genes [163]. Representing the function g_f or these types of cost as SMT formulas is easy thanks to the expressive power of SMT. For simplicity, we assume that g is fixed over time and the cost function is the cost of applying control nodes. Example 6.2.1 shows a DGABN with a setting for optimal control.

Definition 6.2.3. *Given a DGABN including a set of internal nodes $(X = \{x_1, ..., x_n\})$ and a set of external (control) nodes $(U = \{u_1, ..., u_m\})$, an initial state $x^{ini} \in \mathbb{B}^{1\times n}$, a desired state $x^{des} \in \mathbb{B}^{1\times n}$, a target time M, and a cost vector $g \in \mathbb{N}^{1\times m}$. Note that control nodes can appear in Boolean functions of the DGABN, i.e., $f_i : \mathbb{B}^{n+m} \to \mathbb{B}$ $(i = 1, ..., n)$. Let decide whether or not there exists a control sequence of 0-1 control vectors $\langle u(0), ..., u(M-1)\rangle$ such that $x(0) = x^{ini}$, $x(M) = x^{des}$, and the linear cost function $C = \sum_{j=0}^{M-1}(\sum_{i=1}^{m}(u_i(j) \times g(u_i)))$ is minimum. Then, output one if it exists.*

Example 6.2.1. *A DGABN $\mathcal{D}$ includes three internal nodes (x_1, x_2, x_3) and two control nodes (u_1, u_2). Its setting for optimal control is given by*

$$f_1 = \neg u_1, f_2 = x_1 \wedge u_2, f_3 = x_1 \vee x_2;$$
$$p_1 = 2, p_2 = 1, p_3 = 2; q_1 = 1, q_2 = 0, q_3 = 0;$$
$$g(u_1) = 1, g(u_2) = 2; x^{ini} = 000, x^{des} = 011.$$

We here consider two control modes (time-sensitive and non-time-sensitive) for optimal control of DGABNs as for optimal control of other systems [164]. In the time-sensitive mode, the condition $x(M) = x^{des}$ must be strictly satisfied, i.e., the DGABN must reach the desired state at exactly the target time M (as in Definition 6.2.3). The time-sensitive mode is often used in standard forms of optimal control problems of BNs [165]. It is useful when we want, for example, to find a drug treatment over a given time frame $[0, M]$ minimizing the cost of using the drugs. In many applications, we may want, for example, to find a drug treatment such that the cost of using the drugs is minimum and the treatment time should be as early as possible before a given time M. In this case, the non-time-sensitive mode is useful. In the non-time-sensitive mode, the condition $x(M) = x^{des}$ can be relaxed, i.e., the DGABN can reach the desired state before or at time step M. Specifically, the two last sentences of Definition 6.2.3 are replaced by: Let decide whether or not there exists a control sequence of 0-1 control vectors $\langle u(0), ..., u(M'-1)\rangle$ such that

$$x(0) = x^{ini}, x(M') = x^{des}, M' \leq M,$$

and the linear cost function

$$C = \sum_{j=0}^{M'-1} (\sum_{i=1}^{m} (u_i(j) \times g(u_i)))$$

is minimum. Then, output one if it exists.

6.3 SMT-Based Method for the Time-Sensitive Mode

Langmead and Jha proposed an efficient SAT-based method for standard control of SBNs [55]. Inspired by this SAT-based method, we propose two methods for optimal control of DGABNs under the two control modes called **DA-SMT-Con-TS** and **DA-SMT-Con-NTS**, respectively. However, there are some differences between our methods and the SAT-based method. First, the SAT-based method is only applicable to SBNs, whereas our methods are applicable to DGABNs, which are more general than SBNs. Second, the SAT-based method solves the standard control problem, whereas our methods solve the optimal control problem, which is more general than the standard control problem. Third, the SAT-based method only supports the time-sensitive mode, whereas our methods support both the time-sensitive and non-time-sensitive modes.

For the time-sensitive mode, the intuitive idea of **DA-SMT-Con-TS** is as follows. Let es^t be the corresponding extended state of state $x(t)$, where $es^t_i = x_i(t), i \in \{1, ..., n\}$

and $es^t_{n+1} = t\%\gamma$. Let u^t denote the control input at time t. We first encode an M-length path from es^0 to es^M in the ESTG of the DGABN $\mathcal{D}$ as an SMT formula P (see Equation 6.1), which is based on the transition formula between two extended states of the DGABN. We then solve P under minimizing the cost function C in Z3 (see [166] for optimization in Z3). If $\mathbf{SAT}(P)$, then a control sequence and an optimum cost, which can be easily obtained from the satisfying assignments of the corresponding SMT variables, are released. Otherwise, "there are no control policies" is released.

In Equation 6.1, T_{start} (see Equation 6.2) expresses that the path starts with x^{ini} at time $t = 0$ and T_{end} (see Equation 6.3) expresses that the path ends with x^{des} at time $t = M$. Clearly, we can easily adjust T_{start} and T_{end} to express the case of multiple initial states or multiple desired states. This is useful because in the biological context we often only consider the values of some dominant genes, other genes can receive arbitrary values. T^M stands for M transitions of this path. Note that $\mathcal{T}^{\mathcal{D}}(es^j, es^{j+1})$ in Equation 6.5 is the transition formula between two extended states of the DGABN $\mathcal{D}$, which is defined in Equation 6.4.

$$P := T_{start} \wedge T^M \wedge T_{end} \tag{6.1}$$

$$T_{start} := (es^0_{n+1} = 0) \wedge \bigwedge_{i=1}^{n} (es^0_i = x^{ini}_i) \tag{6.2}$$

$$T_{end} := (es^M_{n+1} = M\%\gamma) \wedge \bigwedge_{i=1}^{n} (es^M_i = x^{des}_i) \tag{6.3}$$

$$\begin{aligned} &\mathcal{T}^{\mathcal{D}}(es^j, es^{j+1}) := \left\{es^{j+1} = (es^j + 1)\%\gamma\right\} \wedge \\ &\bigwedge_{i=1}^{n} \left\{\left[(es^j\%p_i = q_i) \wedge (es^{j+1}_i = f_i(es^j, u^j))\right] \vee \left[(es^j\%p_i \neq q_i) \wedge (es^{j+1}_i = es^j_i)\right]\right\} \end{aligned} \tag{6.4}$$

$$T^M := \bigwedge_{j=0}^{M-1} \mathcal{T}^{\mathcal{D}}(es^j, es^{j+1}) \tag{6.5}$$

Let us consider Example 6.2.1. In the time-sensitive mode, we obtain a control sequence as $\langle (0,0), (0,0), (0,0), (1,1)\rangle$ and the minimum cost is $C = 3$ for $M = 4$. This result is shown in Table 6.1. In Table 6.1, Column "t" denotes the time of evolution (not scaled to γ) and Column "Updated nodes" denotes the nodes that will be updated at time t. When $M = 5$, there are no control sequences. However, in the non-time-sensitive mode, we will obtain the control sequence as that for the case $M = 4$.

6.4 SMT-Based Method for the Non-Time-Sensitive Mode

Obviously, optimal control of DGABNs in the non-time-sensitive mode is harder than that in the time-sensitive mode. For standard control of DGABNs, we can simply find the first target time M_{first} ($0 \leq M_{first} \leq M$) in which the control condition is satisfied (i.e., there exists a control sequence driving the DGABN from x^{ini} to x^{des} at time M_{first}).

Table 6.1: Result of the optimal control problem shown in Example 6.2.1 with $M = 4$ under the time-sensitive mode.

Updated nodes	t	x_1	x_2	x_3	u_1	u_2	cost
x_2, x_3	0	0	0	0	0	0	0
x_1, x_2	1	0	0	0	0	0	0
x_2, x_3	2	1	0	0	0	0	0
x_1, x_2	3	1	0	1	1	1	3
	4	0	1	1			

However, for optimal control of DGABNs, the minimum cost of the case that $M = M_{first}$ may not be the smallest minimum cost (the target time corresponding to this smallest minimum cost is called M_{min}, $0 \leq M_{min} \leq M$). Hence, exact methods are needed for optimal control of DGABNs in the non-time-sensitive mode. We here propose a method called **DA-SMT-Con-NTS** to solve this problem.

Based on the method for the time-sensitive mode, we modify the SMT formula P (defined in Equation 6.1) to represent an M-length path from es^0 to es^M in the ESTG of the DGABN such that along with this path, once we reach an extended state satisfying the following condition, all next extended states of the path will be equal to this extended state (i.e., there are no updates). The condition means that the values of internal nodes of the extended state are same as those of the desired state x^{des} and is represented by $\bigwedge_{i=1}^{n}(es_i^j = x_i^{des})$. The transition formula of such a path is shown in Equation 6.7. If the condition does not hold, then we update the DGABN as usual by the state transition formula $\mathcal{T}^{\mathcal{D}}$. Otherwise, we do not update the DGABN. Note that we add a new variable r^j to indicate either the updating case ($r^j = 1$) or the non-updating case ($r^j = 0$). This helps us to represent the new cost function and to easily obtain the real control sequence.

The modified formula P' is shown in Equation 6.6. The cost function is also adjusted as

$$C' = \sum_{j=0}^{M-1}(\sum_{i=1}^{m}(u_i(j) \times g(u_i) \times r(j))),$$

where $r(j)$ corresponds to the variable r^j. We then solve P' under minimizing the cost function C' in Z3. If $\mathbf{SAT}(P')$, then a control sequence and an optimum cost are released. Otherwise, "there are no control policies" is released. The optimum cost can be directly obtained from the satisfying assignment of the SMT variable C'. The control sequence can be obtained by first obtaining a sequence of 0-1 control inputs $\langle u(0), ..., u(M)\rangle$ from the satisfying assignments of the corresponding SMT variables, and then excluding the spurious control inputs. A control input $u(j)$ is said to be spurious if the satisfying assignment of r^j is 0. Note that M_{min} can be determined as the number of control inputs in the control sequence. Furthermore, if we want to minimize both the cost and the target time, we can simply adjust the cost function, e.g,

$$C' = \sum_{j=0}^{M-1}(\sum_{i=1}^{m}(u_i(j) \times g(u_i) \times r(j)) \times (M+1) + r(j)).$$

The adjusted cost function guarantees that the cost is always minimized first, then the

target time is minimized. However, since the cost function becomes more complex, the running time may be longer.

$$P' := T_{start} \wedge T'^{M} \wedge T_{end} \tag{6.6}$$

$$T'^{M} := \bigwedge_{j=0}^{M-1} \mathcal{T}'^{\mathcal{D}}(es^j, es^{j+1}) \tag{6.7}$$

$$\begin{aligned} \mathcal{T}'^{\mathcal{D}}(es^j, es^{j+1}) := & \left\{ \mathcal{T}^{\mathcal{D}}(es^j, es^{j+1}) \wedge \neg \left[\bigwedge_{i=1}^{n} (es_i^j = x_i^{des}) \right] \wedge (r^j = 1) \right\} \\ & \vee \left\{ \left[\bigwedge_{i=1}^{n} (es_i^{j+1} = es_i^j) \right] \wedge \left[\bigwedge_{i=1}^{n} (es_i^j = x_i^{des}) \right] \wedge (r^j = 0) \right\} \end{aligned} \tag{6.8}$$

Let us reconsider the running example in Section 6.3 with $M = 5$ and the non-time-sensitive mode. We here change the desired state x^{des} to 111. By applying **DA-SMT-Con-NTS**, we can obtain the minimum cost $C' = 2$ and a sequence of control inputs $\langle (0,0),(0,0),(0,1),(0,0),(0,0) \rangle$. We also have $r^0 = 1, r^1 = 1, r^2 = 1, r^3 = 0, r^4 = 0$, respectively. By excluding the spurious control inputs, we obtain a control sequence $\langle (0,0),(0,0),(0,1) \rangle$. Herein, $M_{min} = 3$ and is optimal.

6.5 Case Study

In this section, we applied our proposed methods for optimal control of DGABNs to the apoptosis network, which is very important for programmed cell death and has been widely studied [162]. The BN of this network includes 11 internal nodes and one control node. Its details are given in Table 6.2. In this BN, $x_7 = 0$ and $x_9 = 1$ implies cell death, whereas $x_7 = 1$ and $x_9 = 0$ implies cell survival.

Table 6.2: BN model of the apoptosis network.

Gene	Node	Boolean function
TNF	u	
T2	x_1	$\neg x_8 \wedge u$
IKKa	x_2	$\neg x_6 \wedge \neg x_6 \wedge u$
NF_KB	x_3	$\neg x_5$
NF_KB_{nuc}	x_4	$x_3 \wedge \neg x_5$
I_KB	x_5	$(\neg x_2 \wedge x_4 \wedge u) \vee (\neg x_2 \wedge x_4 \wedge \neg u)$
A20a	x_6	$x_4 \wedge u$
IAP	x_7	$(x_4 \wedge \neg x_9 \wedge u) \vee (x_4 \wedge \neg x_9 \wedge \neg u)$
FLIP	x_8	x_4
C3a	x_9	$\neg x_7 \wedge x_{10}$
C8a	x_{10}	$(x_1 \vee x_9) \wedge \neg x_{11}$
CARP	x_{11}	$(x_4 \wedge \neg x_9 \wedge u) \vee (x_4 \wedge \neg x_9 \wedge \neg u)$

Then, we randomly generated four different contexts with $\Lambda = 3$ for this BN because we do not know the knowledge on the real context of the apoptosis network. We now have

four different random DGABNs. The setting for optimal control of all the four DGABNs is given as follows. The initial state was set to $(0, ..., 0)^\top$. The values of x_7 and x_9 in the desired state were set to 0 and 1, respectively. The other nodes in the desired state can receive arbitrary Boolean values. This means that the objective of this control is to guide the network toward cell death states. The target time M was set to 50. Since there is only one control node, we simply set $g(u) = 1$. Herein, we considered the non-time-sensitive mode.

Next, we applied **DA-SMT-Con-NTS** to the four random DGABNs. The obtained results are given in Table 6.3. Column "result" denotes whether a control sequence exists (yes) or not (no). Column "c_{min}" denotes the optimal cost. "-" denotes the case there is no control sequence, in which c_{min} and M_{min} are undetermined. From these results, we can see that the activation of cell death can be controlled by manipulating the value of the control node (gene TNF).

Table 6.3: Results on optimal control of the apoptosis network.

Network	Result	c_{min}	M_{min}
random-1	yes	2	9
random-2	no	-	-
random-3	yes	1	7
random-4	yes	3	38

6.6 Evaluation

We have implemented the proposed methods (i.e., **DA-SMT-Con-TS** and **DA-SMT-Con-NTS**) in a JAVA tool integrating the Z3 solver called **DABoolNet**. An executable file of **DABoolNet** and some example networks are available at https://github.com/giang-trinh/daboolnet. To our best knowledge, **DABoolNet** is the sole tool for optimal control of DGABNs.

In order to evaluate the scalability of the proposed methods, we applied them to one artificial example varying many parameters. The artificial DGABN includes nX internal nodes and nU control nodes ($nX > nU$). Its nodes and Boolean functions are given in Table 6.4. Its context was randomly generated with $\Lambda = 3$ (i.e., $\gamma \leq 6$). Note that in our proposed methods for optimal control of DGABNs, the number of variables and the number of clauses of the path formula do not depend on γ but depend on the number of nodes n and the target time M. Hence, we simply set Λ to 3. The cost vector was set as $g(u_i) = 1 + i\%2$ ($i = 1, ..., nU$). The initial state was fixed to $(1, ..., 1)^\top$ and the desired state was obtained by randomly flipping some nodes in the initial state. Then, we varied nX, nU, and M. Since optimal control of DGABNs in the non-time-sensitive mode is harder than that in the time-sensitive mode, we only ran **DA-SMT-Con-NTS** for each combination of nX, nU, and M. Since the time for solving optimal control is usually less than the time for solving attractor detection, the time limit was set to one hour (instead of three hours as in Section 5.5) for each combination. In addition, we also applied two variants of **DA-SMT-Con-NTS** to each combination to compare their performance. Herein, the first variant (say $\mathcal{M}_1$) corresponds to the case of minimizing only the cost, whereas the second variant (say $\mathcal{M}_2$) corresponds to the case of minimizing

both the cost and the target time. Finally, the running environment is a PC with Intel(R) Core(TM) i7 2.40 GHz processor and 16 GB of memory.

Table 6.4: An artificial example for optimal control of DGABNs.

Node	Boolean function
x_1	$x_1 \wedge x_2 \wedge \neg u_1$
x_i $(i = 2, ..., nU)$	$x_{i-1} \wedge x_i \wedge x_{i+1} \wedge \neg u_i$
x_i $(i = nU + 1, ..., nX - 1)$	$x_{i-1} \wedge x_i \wedge x_{i+1}$
x_{nX}	$x_{nX-1} \wedge x_{nX}$

Table 6.5: Results of **DA-SMT-Con-TS** on the artificial example.

				$\mathcal{M}_1$		$\mathcal{M}_2$	
nX	nU	M	result	M_{min}	time (s)	M_{min}	time (s)
200	10	20	no	-	1.04	-	0.98
200	10	40	no	-	2.02	-	2.19
200	10	80	yes	7	37.23	7	37.91
200	20	20	no	-	1.14	-	1.53
200	20	40	yes	1	7.72	1	6.49
200	20	80	no	-	3.69	-	3.87
300	10	20	yes	3	2.66	3	2.70
300	10	40	no	-	2.70	-	3.43
300	10	80	no	-	5.61	-	5.78
300	20	20	yes	10	2.90	10	3.01
300	20	40	yes	23	10.19	5	18.35
300	20	80	yes	7	40.30	1	45.03
400	10	20	yes	19	3.01	4	3.01
400	10	40	yes	5	15.95	5	17.27
400	10	80	yes	4	119.86	4	130.80
400	20	20	no	-	2.08	-	2.11
400	20	40	yes	8	20.17	8	20.85
400	20	80	yes	4	62.58	4	66.23
500	10	20	no	-	2.58	-	2.41
500	10	40	no	-	14.03	-	10.47
500	10	80	no	-	9.52	-	9.60
500	20	20	yes	4	6.87	4	6.34
500	20	40	yes	6	21.93	2	21.83
500	20	80	yes	13	71.98	1	80.78

Table 6.5 shows the obtained results. Column "result" denotes whether a control sequence exists (yes) or not (no). Column "time " stands for the computational time (in seconds) for each combination of nX, nU, and M. "-" denotes the case that there are no control sequences, in which M_{min} is undetermined. In some combinations (e.g., $nX = 300, nU = 20, M = 40$), M_{min} obtained by $\mathcal{M}_2$ is less than M_{min} obtained by $\mathcal{M}_1$. In addition, $\mathcal{M}_2$ is slower than $\mathcal{M}_1$ in most combinations. These observations are consistent with the analysis on $\mathcal{M}_2$ presented in Section 6.4. Furthermore, the computational time

of $\mathcal{M}_1$ and $\mathcal{M}_2$ for each combination is reasonable even when nX and M are large (e.g., $nX = 500$ and $M = 80$). To sum up, we see that **DA-SMT-Con-NTS** can solve optimal control in the non-time-sensitive mode of large DGABNs within practical computational time. We also suggest to preferably use $\mathcal{M}_1$ when we only focus on minimizing the cost of applying control nodes and to preferably use $\mathcal{M}_2$ when we focus on minimizing both the cost of applying control nodes and the target time.

6.7 Discussion

Besides attractor detection, optimal control is also one of central issues in systems biology. In this chapter, based on the theoretical foundation for DGABNs proposed in Chapter 5, we have developed the two SMT-based methods for optimal control of DGABNs under the two control modes, time-sensitive (**DA-SMT-Con-TS**) and non-time-sensitive (**DA-SMT-Con-NTS**). Then, we have shown the application of the proposed methods to a real biological network. We have also applied the proposed methods to optimal control of one artificial example varying many parameters. The experimental results show that our methods can handle well large networks.

Since SMT supports a variety of data types and arithmetic operators, it is potentially possible to extend the SMT-based methods for attractor detection and optimal control of DGABNs (i.e., **DA-SMT-Att**, **DA-SMT-Con-TS**, and **DA-SMT-Con-NTS**) to those for multi-valued models of DGABNs where each node can receive multiple values, and more operators are introduced [167]. It is also interesting to extend **DA-SMT-Att** to incorporate gene perturbation experiments in DGABNs or to extend **DA-SMT-Con-TS** and **DA-SMT-Con-NTS** to incorporate node perturbations control of DGABNs (i.e., the first main control direction mentioned in Section 6.1).

DGABNs are particularly useful when the information about the kinetics of biological processes is known [19]. However, the prior kinetic information is usually not available. In this case, the context (i.e., $p's$ and $q's$) can be randomly sampled from a time interval that is within biological limitations [19, 150]. In addition, MxBNs are an non-deterministic extension of DGABNs to deal with the case of lacking knowledge on real contexts. Deterministic Asynchronous Probabilistic Boolean Networks (DA-PBNs) [165] are a stochastic extension of DGABNs to deal with the case of lacking information on real Boolean functions. Therefore, proposing efficient methods for attractor detection and optimal control of MxBNs or DA-PBNs is one of our future work. In the next chapter, we shall consider optimal control of DA-PBNs.

Chapter 7

Optimal Control of Deterministic Asynchronous Probabilistic Boolean Networks

7.1 Introduction

In recent years, control of deterministic or probabilistic BNs has received considerable attention. Regarding deterministic BNs, two efficient SMT-based methods have been proposed for optimal control of DGABNs under the two control modes (see Chapter 6). In addition, the time-variant state feedback stabilization problem of DGABNs was studied in [76]. The proposed reachable set approach for solving this problem needs to handle a matrix of size exponential in the number of nodes (i.e., $\mathcal{O}(2^n \times 2^n)$). Hence, the proposed approach is only applicable to networks with small n. Regarding probabilistic BNs, finite-horizon control of SPBNs and its variants (e.g., optimal control [64], control with hard constraints [71]) are usually considered. The controllability and stabilization of SPBNs were studied in [168, 169] by expressing an SPBN in an algebraic form based on the Semi-Tensor Product (STP) [170]. Like [76], this STP-based approach also needs to handle matrices of size exponential in the number of nodes. The dynamic programming-based approach for solving optimal control of SPBNs was proposed and gradually improved [64, 67, 70, 71, 73, 74]. In general, the methods of this approach rely on the theory of discrete-time Markov decision process [64]. They require to compute transition probability matrices of size exponential in the number of nodes; hence, they are impractical for large networks. To avoid computing transition probability matrices in finite-horizon optimal control of SPBNs, several efficient approaches were proposed, such as, the integer programming-based approach [65, 66, 71] (for some special variants), the optimization-based approach [68], the model checking-based approach [69, 72].

In this chapter, we focus on optimal control of DA-PBNs because of the following reasons. First, DA-PBNs are general models: A DGABN is a special DA-PBN and an SPBN is a special DA-PBN [16]. Consequently, optimal control of DA-PBNs is harder than that of DGABNs or SPBNs. In addition, developed methods for optimal control of DA-PBNs can be directly applied for those of DGABNs or SPBNs. Second, in the context of systems biology, DA-PBNs seem to be more suitable to model GRNs, since a DA-PBN comprises all the synchronous, asynchronous, and probabilistic natures [16, 77]. Finally, it lacks efficient methods for optimal control of DA-PBNs. To our best knowledge, [77]

is the sole method for optimal control of DA-PBNs. However, this method is inefficient because it requires to build transition probability matrices of size $(\gamma 2^n) \times (\gamma 2^n)$, where γ is a given constant.

Based on several typical aims of control, which are mainly inspired by realistic applications in systems biology [21, 69], we formulate three finite-horizon optimal control problems of DA-PBNs, called Problems OptC-1, OptC-2, and OptC-3 (see Section 7.3). Problem OptC-1 (resp. OptC-2) aims at finding a control policy that maximizes (resp. minimizes) the reachability probability from the initial state to the target state at a given time step of the considered DA-PBN. Problem OptC-3 aims at finding a control policy that minimizes a given cost function for applying interventions. For theoretical aspects, we present several analysis on computational complexity of the three problems in both the general and restricted cases (see Section 7.4). For practical aspects, we propose three approaches for solving the three problems (see Section 7.5), which are based on Probabilistic Model Checking (PMC) [171], Stochastic Satisfiability Modulo Theory (SSMT) [172], and Polynomial Optimization Problem (POP) [173], respectively. The PMC-based and POP-based approaches are not new but they are non-trivial extensions of those [68, 69] for optimal control of SPBNs. In addition, two reduction rules are developed to reduce the computational burden of the POP-based approach. The SSMT-based approach is new and gives more potentials that come from efficient solving techniques for SSMT [172].

In addition to a case study on a realistic network (see Section 7.5), computational experiments are performed to evaluate the performance of the three proposed approaches (see Section 7.6). Based on the experimental results, we present experimental analysis along with theoretical analysis on the effects of some factors (e.g., the number of nodes, the target time step) on the performance of the proposed approaches. We also present a comprehensive comparison among these approaches. These analysis and comparison are significant contributions of this research because of the following reasons. In all the previous work, the proposed approaches for optimal control of SPBNs or DA-PBNs were only compared to the dynamic programming-based approach. It lacks a comparison among different proposed approaches in both theoretical and experimental aspects. It also lacks analysis on how the running time of the proposed approaches depends on some factors in the control setting.

7.2 Preparations

Before defining the optimal problems of DA-PBNs, we briefly review PBNs with control nodes. In the rest of this chapter, we refer a PBN as a PBN with control nodes.

Definition 7.2.1. *A Probabilistic Boolean Network (PBN) with control nodes is defined as a quadruple (V, F, U, C), where $V = \{x_1, ..., x_n\}$ $(n \geq 1)$ is the set of internal nodes, $U = \{u_1, ..., u_m\}$ $(m \geq 0)$ is the set of external (control) nodes, $F = \{F_1, ..., F_n\}$, and $C = \{C_1, ..., C_n\}$. Each node x_i is identified as a Boolean variable, and is associated with a non-empty set of Boolean functions, $F_i = \{f_1^{(i)}, ..., f_{l_i}^{(i)}\}$. Each Boolean function $f_j^{(i)}$ has a probability of selection associated with it, $c_j^{(i)}$. Thus, $C_i = \left\{c_1^{(i)}, ..., c_{l_i}^{(i)}\right\}$ such that $\sum_{j=1}^{l_i} c_j^{(i)} = 1$. Also, each node u_i is identified as a Boolean variable. $x_i(t) \in \mathbb{B}$ and $u_i(t) \in \mathbb{B}$ denote the state of internal node x_i and the state of external node u_i at time t, respectively. $x(t) = (x_1(t), ..., x_n(t))^\top$ and $u(t) = (u_1(t), ..., u_m(t))^\top$ denote the state and*

the control input of the PBN at time t, respectively.

Internal node x_i updates its state by

$$x_i(t+1) = f_j^{(i)}(x(t), u(t)),$$

where $f_j^{(i)}$ is a Boolean function selected from F_i with the probability $c_j^{(i)}$. A control node can receive an arbitrary Boolean value at each time step. Similar to BNs, an updating scheme of a PBN specifies the way that the internal nodes of the PBN update their states through time evolution. Following the updating scheme, the PBN transits from a state to another state (possibly identical) with a probability. This transition is called the *probability transition*. From this, the dynamics of a PBN can be represented by all possible states of the PBN along with all possible probability transitions from each state.

There are two typical types of PBNs. The first one is Synchronous Probabilistic Boolean Networks (SPBNs) [4], where all the nodes are updated simultaneously. The second one is Deterministic Asynchronous Probabilistic Boolean Networks (DA-PBNs) [15, 16] whose formal definition is given in Definition 7.2.2. Roughly speaking, SPBNs and DA-PBNs are probabilistic extensions of SBNs and DGABNs, respectively.

Definition 7.2.2. *A DA-PBN [15, 16] is a PBN such that each internal node x_i is associated with two parameters, $p_i \in \mathbb{N}^+$ and $q_i \in \mathbb{N}, q_i < p_i$. The set of all $p's$ and $q's$ is called the context of a DA-PBN. Let Λ denote the maximum value of all $p's$ (i.e., $p_i \leq \Lambda, \forall i \in \{1, ..., n\}$). Internal node x_i can be updated at time t if $t\%p_i = q_i$. If multiple internal nodes can be updated, then all of them are updated simultaneously.*

In SBNs, ABNs, or SPBNs, the evolution of a state is time-invariant. However, in DGABNs or DA-PBNs, the evolution of a state depends on the time of entering this state. The concept of an *extended state* is defined in Section 5.2, allowing to express the dynamics of a DGABN by extended states and state transitions starting from these extended states. Analogously, we can use extended states and probability transitions starting from these extended states to represent the dynamics of a DA-PBN.

$ev \in \mathbb{B}^n \times \{0, ..., \gamma - 1\}$ is an extended state, where γ is the least common multiple of all p's. ev_i $(i = 1, ..., n)$ denotes the value of internal node x_i, whereas ev_{n+1} denotes the value of the embedded time. The number of possible extended states is $\gamma \times 2^n$; but, there are only 2^n possible initial extended states (i.e., the states satisfying $ev_{n+1} = 0$) of the DA-PBN. Note that the concept of an extend state is similar to the concept of an *augmented logical state* [15]. However, an augmented logical state uses $\lceil \log_2(\gamma) \rceil$ new Boolean variables to represent $t\%\gamma$, whereas an extended state uses an integer variable. Then, the probability of transiting from extended state a to extended state b under control input u is

$$P\left(ev(t+1) = b | ev(t) = a, u(t) = u\right) = P(b_{n+1} = (a_{n+1} + 1)\%\gamma)$$
$$\times \prod_{i \in \mathbb{N}^+_{\leq n} \setminus \Omega} P(b_i \leftrightarrow a_i) \times \sum_{j_1 \in \mathbb{N}^+_{\leq l_{i_1}}, ..., j_k \in \mathbb{N}^+_{\leq l_{i_k}}} \left\{ \prod_{h=1}^{k} \left[c_{j_h}^{(i_h)} \times P\left(b_{i_h} \leftrightarrow f_{j_h}^{i_h}(a, u)\right) \right] \right\},$$

where $\Omega := \{x_{i_1}, ..., x_{i_k}\}$ is the set of updated nodes (i.e., $a_{n+1}\%p_{i_j} = q_{i_j}, j \in \mathbb{N}^+_{\leq k}$). See Example 7.2.1 for a DA-PBN and its dynamics.

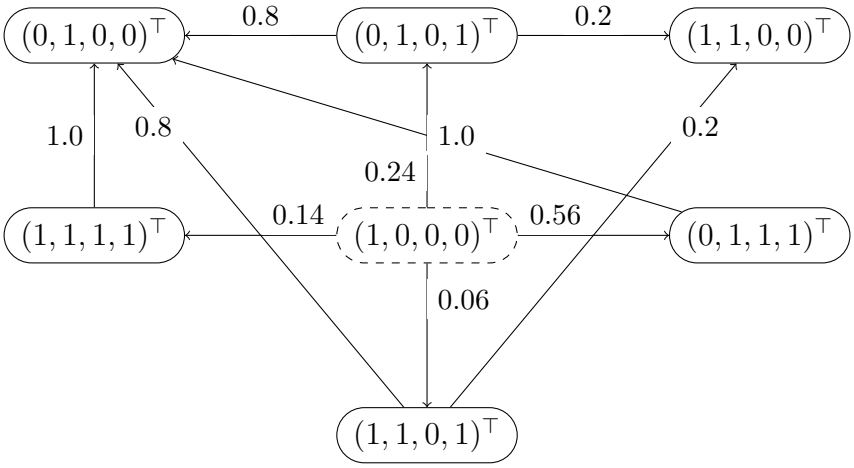


Figure 7.1: Dynamics of the DA-PBN shown in Example 7.2.1. Extended states are shown by rounded rectangles, whereas probability transitions are shown by arcs along with real numbers. The initial extended state is shown by the dashed rounded rectangle.

Example 7.2.1. *Consider the DA-PBN* $\mathcal{DP}$

$$f^{(1)} = \begin{cases} f_1^{(1)} = x_3 \wedge u_1, c_1^{(1)} = 0.8 \\ f_2^{(1)} = \neg x_3, c_2^{(1)} = 0.2 \end{cases}, p_1 = 1, q_1 = 0;$$

$$f^{(2)} = f_1^{(2)} = x_1 \wedge \neg x_3, c_1^{(2)} = 1.0 \quad , p_2 = 2, q_2 = 0;$$

$$f^{(3)} = \begin{cases} f_1^{(3)} = x_1 \wedge \neg x_2, c_1^{(3)} = 0.7 \\ f_2^{(3)} = x_2 \wedge u_1, c_2^{(3)} = 0.3 \end{cases}, p_3 = 1, q_3 = 0.$$

Then, Figure 7.1 shows all probability transitions starting from $(1,0,0)^\top$ *of* $\mathcal{DP}$ *under* $u_1(0) = u_1(1) = 0$.

7.3 Problem Formulation

We formulate three optimal control problems of DA-PBNs: OptC-1, OptC-2, and OptC-3. Problems OptC-1 and OptC-2 are generalized from the reachability problem and the safety problem of SPBNs [69], respectively. See Example 7.3.1 for an illustration of Problems OptC-1 and OptC-2. Problem OptC-3 is generalized from the expected cost problem of SPBNs [68]. See Example 7.3.2 for an illustration of Problems OptC-3. Each of these problems is suitable for a specific aim of control [68, 69].

Definition 7.3.1 (Problem OptC-1)**.** *Given a DA-PBN* $\mathcal{DP}$*. Suppose that an initial state of* $\mathcal{DP}$ *is* x^{ini} *and a desired state of* $\mathcal{DP}$ *is* x^{des}*. Find a control sequence* $\langle u(0), ..., u(M-1)\rangle$ *that maximizes the reachability probability from* x^{ini} *to* x^{des} *at time* M*.*

Definition 7.3.2 (Problem OptC-2)**.** *Given a DA-PBN* $\mathcal{DP}$*. Suppose that an initial state of* $\mathcal{DP}$ *is* x^{ini}*, the unsafe state of* $\mathcal{DP}$ *is* x^{des}*, and* $\varepsilon \in [0,1]$ *is given. Find a control sequence* $\langle u(0), ..., u(M-1)\rangle$ *that minimizes the reachability probability from* x^{ini} *to* x^{des} *at time* M*. If the minimum probability is at most* ε*, then* $\mathcal{DP}$ *is said to be safe.*

Definition 7.3.3 (Problem OptC-3)**.** *Given a DA-PBN $\mathcal{DP}$. Suppose that the initial state of $\mathcal{DP}$ is x^{ini} and the control time M is given. Find a control sequence $\langle u(0), ..., u(M-1)\rangle$ that minimizes the expected cost*

$$J = E\left[\sum_{k=0}^{M-1}\{\mathcal{Q}x(k) + \mathcal{R}u(k)\} + \mathcal{Q}_f x(M) \,\middle|\, x(0) = x^{ini}\right],$$

where $\mathcal{Q}, \mathcal{Q}_f \in \mathbb{R}^{1\times n}, \mathcal{R} \in \mathbb{R}^{1\times m}$ are weighting vectors.

Table 7.1: Reachability probability and expected cost with all possible control sequences.

$u_1(0)$	$u_1(1)$	Reachability probability	Expected cost
0	0	0.06	0
0	1	0.434	0.3
1	0	0.06	0
1	1	0.434	0.3

Example 7.3.1. *Consider the DA-PBN in Example 7.2.1. Suppose that $x^{ini} = (1,0,0)^\top$, $x^{des} = (1,1,0)^\top$, and $M = 2$. For $u_1(0) = u_1(1) = 0$, we have two 2-length paths from $(1,0,0,0)^\top$ to $(1,1,0,0)^\top$ (see Figure 7.1). Thus, the reachability probability is $0.24 \times 0.2 + 0.06 \times 0.2 = 0.06$. In a similar way, we can calculate the reachability probability for other control sequences (see Table 7.1). The solution for Problem OptC-1 is $\langle(0)^\top,(1)^\top\rangle$ or $\langle(1)^\top,(1)^\top\rangle$ with $P_{max} = 0.434$ (P_{max} is the maximum reachability probability). The solution for Problem OptC-2 is $\langle(0)^\top,(0)^\top\rangle$ or $\langle(1)^\top,(0)^\top\rangle$ with $P_{min} = 0.06$ (P_{max} is the minimum reachability probability). If $\varepsilon = 0.1$, then the DA-PBN is safe. If $\varepsilon = 0.01$, then the DA-PBN is not safe.*

Example 7.3.2. *Consider the DA-PBN in Example 7.2.1. Suppose that $x^{ini} = (1,0,0)^\top$, $M = 2$, and $J = E[x_3(2)]$. Relying on Figure 7.1, we have $E[x_3(1)] = 0.14 + 0.56 = 0.7$, $E[x_3(2)] = 0$ if $u_1(0) = u_1(1) = 0$. In a similar way, we can calculate the expected cost for other control sequences (see Table 7.1). The solution for Problem OptC-3 is $\langle(0)^\top,(1)^\top\rangle$ or $\langle(1)^\top,(1)^\top\rangle$ with $J_{min} = 0$ (J_{min} is the minimum expected cost).*

In Problem OptC-3, we consider that a linear cost function is appropriate because of the following reasons. For a binary variable $\delta \in \mathbb{B}$, the relation $\delta^2 = \delta$ holds. Thus, in the cost function, the quadratic term such as $x_i^2(k)$ is not necessary. In control of GRNs, the expression of a certain gene is frequently focused (see, e.g., [64]). Thus, in the cost function, the quadratic term such as $x_i(k)x_j(k), i \neq j$ is not necessary. By allowing negative numbers in elements of weighting vectors, we can express more control aims. For example, if we want to reduce the activity of a gene x_i in affecting biological regulation, we can set $\mathcal{Q}_f(x_i)$ as a large positive real number. On the other hand, if we want to upgrade the activity of a gene x_i in affecting biological regulation, we can set $\mathcal{Q}_f(x_i)$ as a small negative real number.

7.4 Complexity Analysis

In this section, we give several analysis on the computational complexity of the three optimal control problems of DA-PBNs. To analyze the time complexity with respect to

n and m, we assume that the target time (i.e., M) is polynomially bounded by n. Since it is not realistic to consider an exponential number of time steps, this assumption is reasonable. In addition, we assume that the time for evaluating a Boolean function is polynomial. To analyze the space complexity with respect to n and m, we assume that $l_i, i = 1, ..., n$ are polynomially bounded by n and the amount of space for storing and evaluating a Boolean function is polynomial.

7.4.1 Complexity of Problem OptC-1

Theorem 7.4.1. *Problem OptC-1 is NP^{PP}-hard.*

Proof. We use a polynomial-time reduction from the E-MAJ-SAT problem to Problem OptC-1. An E-MAJ-SAT instance is defined by a CNF Boolean formula $\psi(y, z)$ over Boolean variables $y = [y_1, ..., y_m], m \geq 1$ and $z = [z_1, ..., z_n], n \geq 0$. The task is to decide whether there is a y-instantiation under which the majority of z-instantiations satisfying the propositional formula $\psi(y, z)$. E-MAJ-SAT is $\mathrm{NP}^{\mathrm{PP}}$-complete [174].

From a given $\psi(y, z)$, we construct a DA-PBN as follows:

$$
\begin{aligned}
&V = \{x_1, ..., x_{m+n+1}\}, U = \{u_1, ..., u_m\},\\
&f^{(i)} = u_i \wedge \neg x_{m+n+1}, i \in \mathbb{N}^+_{\leq m},\\
&f_1^{(m+i)} = x_{m+i} \wedge \neg x_{m+n+1}, c_1^{(m+i)} = 0.5, i \in \mathbb{N}^+_{\leq n},\\
&f_2^{(m+i)} = \neg x_{m+i} \wedge \neg x_{m+n+1}, c_2^{(m+i)} = 0.5, i \in \mathbb{N}^+_{\leq n},\\
&f^{(m+n+1)} = \psi[y_1/x_1, ..., y_m/x_m, z_1/x_{m+1}, ..., z_n/x_{m+n}],\\
&p_i = 2, i \in \mathbb{N}^+_{\leq m+n+1}, q_i = 0, i \in \mathbb{N}^+_{\leq m+n}, q_{m+n+1} = 1.
\end{aligned}
$$

In the constructed DA-PBN, both x_i and u_i correspond to y_i for $i \in \mathbb{N}^+_{\leq m}$, x_{m+i} corresponds to z_i for $i \in \mathbb{N}^+_{\leq n}$, and x_{m+n+1} corresponds to $\psi(y, z)$. In addition, the updating pattern is: $x_i, i \in \mathbb{N}^+_{\leq m+n}$ are updated synchronously, then x_{m+n+1} is updated.

We let $x^{ini} = (0, ..., 0)^\top$, $x^{des} = (0, ..., 1)^\top$, and $M = 3$. We prove that the maximum reachability probability of the constructed DA-PBN is larger than 0.5 if and only if the result of E-MAJ-SAT is true.

Suppose that the result of E-MAJ-SAT is true for an assignment of $y = [b_1, ..., b_m]$. Then, it is straightforward to see that by letting $u(0) = (b_1, ..., b_m)^\top$, $x_{m+n+1}(2) = 1$ holds for the majority of choices of probabilistic rules on x_{m+i}, $i = 1, ..., n$. Once $x_{m+n+1}(2) = 1$ holds, the DA-PBN always reaches $(0, ..., 1)^\top$ at time $t = 3$. Hence, the reachability probability of reaching $(0, ..., 1)^\top$ at time $t = 3$ is greater than 0.5. As a consequence, the maximum reachability probability of the constructed DA-PBN is greater than 0.5.

Conversely, suppose that the maximum reachability probability of the constructed DA-PBN is greater than 0.5 with a control sequence $u = \langle u(0), u(1), u(2)\rangle$. At time $t = 2$, $x_{m+n+1}(3) = x_{m+n+1}(2)$ holds. In addition, if $x_{m+n+1}(2) = 1$, then $x_i(3) = 0$ holds for $\forall i \in \mathbb{N}^+_{\leq m+n}$. Hence, the maximum reachability probability is equal to the probability of reaching state $x(2)$ such that $x_{m+n+1}(2) = 1$. From the definition of the reachability probability, the majority of z-instantiations must satisfy $\psi(y, z)$. We can choose an y-instantiation as $u(0)$. Now, the result of E-MAJ-SAT is true.

Since the reduction can be done in time polynomial in m and n, the theorem holds. □

Theorem 7.4.2. *Problem OptC-1 is in PSPACE.*

Proof. We here show an algorithm for solving Problem OptC-1 and this algorithm requires a polynomial amount of space. The algorithm tries all the $2^{m\times M}$ possible control sequences. We use a binary counter of $m \times M$ bits. The counter increases from 0 to $2^{m\times M} - 1$ and the counter value corresponds to a control sequence.

In each control sequence, we use another counter whose value corresponds to a choice of Boolean functions of all the n nodes. Clearly, we need $M \times \sum_{i=1}^{n} \lceil log_2(l_i) \rceil$ bits for this counter. For each choice, we cumulatively calculate the probability such that $x^M = x^{des}$. This computation only needs a polynomial space. We then compare this probability with P_{max}, and then update P_{max} and the maximum control sequence if the current probability is larger than P_{max}.

Since M and $l_i, i = 1, ..., n$ are polynomially bounded, the proposed algorithm only needs a polynomial amount of space. Since PSPACE = NPSPACE, the theorem holds. □

7.4.2 Complexity of Problem OptC-2

Theorem 7.4.3. *Problem OptC-2 is NP^{PP}-hard.*

Proof. Similar to the proof of Theorem 7.4.1, we use a polynomial-time reduction from E-MAJ-SAT to Problem OptC-2.

We modify the constructed DA-PBN for Problem OptC-1 as follows:

$$
\begin{aligned}
&f^{(i)} = u_i \vee \neg x_{m+n+1}, i \in \mathbb{N}^+_{\leq m},\\
&f_1^{(m+i)} = x_{m+i} \vee \neg x_{m+n+1}, c_1^{(m+i)} = 0.5, i \in \mathbb{N}^+_{\leq n},\\
&f_2^{(m+i)} = \neg x_{m+i} \vee \neg x_{m+n+1}, c_2^{(m+i)} = 0.5, i \in \mathbb{N}^+_{\leq n}.
\end{aligned}
$$

We let $x^{ini} = (0, ..., 0, 1)^\top$, $x^{des} = (1, ..., 1, 0)^\top$, and $M = 3$. We prove that the minimum reachability probability of the constructed DA-PBN is less than 0.5 if and only if the result of E-MAJ-SAT is true.

Suppose that the result of E-MAJ-SAT is true for an assignment of $y = [b_1, ..., b_m]$. Then, it is straightforward to see that by letting $u(0) = (b_1, ..., b_m)^\top$, $x_{m+n+1}(2) = 1$ holds for the majority of choices of probabilistic rules on x_{m+i}, $i = 1, ..., n$. If $x_{m+n+1}(2) = 1$ holds, then $x_{m+n+1}(3) = 1$ holds. Then, the probability of reaching $(1, ..., 1, 0)^\top$ at time $t = 3$ is less than 0.5. Hence, the minimum reachability probability of the constructed DA-PBN must be less than 0.5.

Conversely, suppose that the minimum reachability probability of the constructed DA-PBN is less than 0.5 with a control sequence $u = \langle u(0), u(1), u(2) \rangle$. Assume that the result of E-MAJ-SAT is not true. Then, it is straightforward to see that the probability of reaching a state $x(2)$ such that $x_{m+n+1}(2) = 0$ holds is greater than or equal to 0.5 for any control input $u(0)$. Since $u(1)$ and $u(2)$ do not affect the value of x_{m+n+1}, the minimum reachability probability of the constructed DA-PBN must be greater than or equal to 0.5, a contradiction. Hence, the result of E-MAJ-SAT is true.

Since the reduction can be done in time polynomial in m and n, the theorem holds. □

Theorem 7.4.4. *Problem OptC-2 is in PSPACE.*

Proof. The proof is analogous to the proof for the PSPACE-ness of Problem OptC-1 (see Theorem 7.4.2). Specifically, in each control sequence, we only need to compare the cumulatively computed probability with P_{min}, and then update P_{min} and the minimum control sequence if the current probability is less than P_{min}. □

7.4.3 Complexity of Problem OptC-3

Theorem 7.4.5. *Problem OptC-3 is NP^{PP}-hard.*

Proof. Similar to the proof of Theorem 7.4.1, we use a polynomial-time reduction from E-MAJ-SAT to Problem OptC-3.

We modify the constructed DA-PBN for Problem OptC-1 as follows:

$$
\begin{aligned}
&f^{(i)} = u_i, i \in \mathbb{N}^+_{\leq m},\\
&f_1^{(m+i)} = x_{m+i}, c_1^{(m+i)} = 0.5, i \in \mathbb{N}^+_{\leq n},\\
&f_2^{(m+i)} = \neg x_{m+i}, c_2^{(m+i)} = 0.5, i \in \mathbb{N}^+_{\leq n}.
\end{aligned}
$$

We let $x^0 = (0, ..., 0)^\top$, $M = 2$, and $J = 1 - E[x_{m+n+1}(2)]$. We prove that the minimum expected value of the constructed DA-PBN (say J_{min}) is less than 0.5 if and only if the result of E-MAJ-SAT is true.

Suppose that the result of E-MAJ-SAT is true for an assignment of $y = [b_1, ..., b_m]$. Then, it is straightforward to see that by letting $u(0) = (b_1, ..., b_m)^\top$, $x_{m+n+1}(2) = 1$ holds for the majority of assignments on $[x_{m+1}(1), ..., x_{m+n}(1)]$. Hence, $E[x_{m+n+1}(2)] > 0.5$ holds. Then, $J < 0.5$. Since $J_{min} \leq J$, $J_{min} < 0.5$ holds.

Conversely, suppose that $J_{min} < 0.5$ holds with a control sequence $u = \langle u(0), u(1) \rangle$. Then, $E[x_{m+n+1}(2)] > 0.5$ holds. From the definition of the expected value, $x_{m+n+1}(2) = 1$ holds for the majority of assignments on $[x_{m+1}(1), ..., x_{m+n}(1)]$. We can choose an y-instantiation as $u(0)$. Now, the result of E-MAJ-SAT is true.

Since the reduction can be done in time polynomial in m and n, the theorem holds. □

Lemma 7.4.1 ([175])**.** *Consider two binary variables δ_1 and δ_2. Then the following relations hold.*

1. *$\neg\delta_1$ is equivalent to $1 - \delta_1$.*
2. *$\delta_1 \wedge \delta_2$ is equivalent to $\delta_1\delta_2$.*
3. *$\delta_1 \vee \delta_2$ is equivalent to $\delta_1 + \delta_2 - \delta_1\delta_2$.*
4. *δ_1^2 is equivalent to δ_1.*

By Lemma 7.4.1, a given Boolean function can be transformed into a polynomial on the real number field. For example, $(x_2 \vee x_3) \wedge \neg x_1$ is equivalently transformed into $(x_2 + x_3 - x_2x_3)(1 - x_1)$. For simplicity of notation, the condition $E[x(k) \mid *]$ is omitted. $\hat{f}^{(i)}$ denotes the polynomial corresponding to the Boolean function $f^{(i)}$. $\hat{f}_j^{(i)}$ denotes the polynomial corresponding to the Boolean function $f_j^{(i)}$.

Theorem 7.4.6. *Given a DA-PBN $\mathcal{DP}$. The expected value of a node, $E[x_i(k)]$ with $i = 1, ..., n$, is expressed by*

$$
\Sigma_i(k) : E[x_i(k+1)] = \begin{cases} E[x_i(k)] \text{ if } k\%p_i = q_i, \\ \sum_{j=1}^{l_i} c_j^{(i)} \hat{f}_j^{(i)}(E[x(k)], u(k)) \quad \text{otherwise.} \end{cases}
$$

Proof. If $k\%p_i = q_i$, then x_i is not updated at time k. Immediately, $E[x_i(k+1)] = E[x_i(k)]$ holds.

The opposite part follows from [68]. □

Example 7.4.1. *Suppose that* $x^{ini} = (1,0,0)^\top, u_1(0) = u_1(1) = 0$. *Relying on Theorem 7.4.6, we have*

$$\begin{aligned}
&E[x_1(1)] = 0.8[x_3(0)u_1(0)] + 0.2[1 - x_3(0)] = 0.2,\\
&E[x_2(1)] = 1.0x_1(0)[1 - x_3(0)] = 1.0,\\
&E[x_2(2)] = E[x_2(1)] = 1.0,\\
&E[x_3(2)] = 0.7E[x_1(1)](1 - E[x_2(1)]) + 0.3E[x_2(1)]u_1(1) = 0.
\end{aligned}$$

Theorem 7.4.7. *Problem OptC-3 is in PSPACE.*

Proof. We consider the brute-force algorithm that tries all the $2^{m\times M}$ possible control sequences. We use a binary counter of $m \times M$ bits. The counter increases from 0 to $2^{m\times M} - 1$ and the counter value corresponds to a control sequence.

Let J_{min} and u_{min} denote the minimum expected cost and the corresponding control sequence, respectively. For each control sequence, we can recursively calculate $E[x_i(k)], i \in \{1, ..., n\}, k \in \{1, ..., M\}$ by Theorem 7.4.6. Then, J can be directly obtained. If $J < J_{min}$, then $J_{min} \leftarrow J$ and $u_{min} \leftarrow$ the current control sequence.

We need $m \times M$ bits to store u_{min}. In the calculation of $E[x_i(k)]$, we assume that the amount of space for storing and evaluating Boolean functions is polynomial. J_{min} and $E[x_i(k)]$ are real numbers. In addition, the amount of space for storing $E[x_i(k)]$ can be reused in the next control sequence. Since M is polynomially bounded, the amount of space needed for this algorithm is polynomial. Since PSPACE = NPSPACE, the theorem holds. □

7.4.4 Remarks

From the obtained complexity results, we derive several remarks as follows.

First, if the propositional formula φ of E-MAJ-SAT is in 2CNF, then $f^{(m+n+1)}$ of the constructed DA-PBN of Problem OptC-1 is also in 2CNF. In addition, $f^{(i)}, i = 1, ..., m+n$ can be converted into 2CNF in polynomial time. For example, we can add a dummy variable x_{m+n+2} whose value is always 0 to the DA-PBN as follows:

$$\begin{aligned}
f^{(i)} &= (u_i \vee x_{m+n+2}) \wedge (\neg x_{m+n+1} \vee x_{m+n+2}),\\
f_1^{(m+i)} &= (x_{m+i} \vee x_{m+n+2}) \wedge (\neg x_{m+n+1} \vee x_{m+n+2}),\\
f_2^{(m+i)} &= (\neg x_{m+i} \vee x_{m+n+2}) \wedge (\neg x_{m+n+1} \vee x_{m+n+2}).
\end{aligned}$$

Since E-MAJ-SAT with the restriction that the formula φ is in 2CNF is still $\mathrm{NP}^{\mathrm{PP}}$-complete [174], Problem OptC-1 is still $\mathrm{NP}^{\mathrm{PP}}$-hard with the restriction that Boolean functions are in 2CNF. Analogously, we also have the same results for Problems OptC-2 and OptC-3.

Second, the Σ_2^p-hardness of the counterpart of Problem OptC-3 for SPBNs is proved in [71]. By using similar reductions, we can easily obtain the Σ_2^p-hardness of the counterparts of Problems OptC-1 and OptC-2 for SPBNs. Thus, the optimal control problems of DA-PBNs seem harder than the corresponding optimal control problems of SPBNs, since $\Sigma_2^p \subseteq \mathrm{PH} \subseteq \mathrm{NP}^{\mathrm{PP}}$ [174]. It is reasonable because an SPBN is a special DA-PBN.

Third, both SAT and ILP are efficient techniques and have been applied to control of SBNs [21]. ILP has even been applied to some special variants of optimal control of

SPBNs [65, 71]. It is reasonable to develop similar SAT-based or ILP-based algorithms for the three problems of DA-PBNs. However, it is not plausible that such algorithms exist even in the case of SPBNs because SAT and ILP are NP-complete and NP $\subseteq \Sigma_2^p \subseteq$ PH $\subseteq$ NP$^{\text{PP}}$ [174].

Finally, all the three problems are NP$^{\text{PP}}$-hard. It is known that PH $\subseteq$ NP$^{\text{PP}}$ $\subseteq$ PSPACE [174]. Since all the three problems in PSPACE, their complexity is between NP$^{\text{PP}}$-hard and PSPACE-complete. Hence, these problems are hard to solve in general. Moreover, in a crude sense, NP$^{\text{PP}}$ is very close to PSPACE. Therefore, to solve these problems, we may have to encode them as PSPACE-complete problems.

7.5 Proposed Solution Approaches

We propose two solution approaches for solving Problem OptC-1. The first proposed approach is based on Probabilistic Model Checking (PMC) [171] and can be seen as an extension of that for SPBNs [69] to DA-PBNs. The second proposed approach is a new approach that relies on Stochastic Satisfiability Modulo Theory (SSMT) [172]. With slight modifications, the two approaches can be applied to Problems OptC-2 and OptC-3. For Problem OptC-3, we propose a solution approach that relies on Polynomial Optimization Problem (POP) [173]. The POP-based approach can be seen as an extension of that for SPBNs [68] to DA-PBNs. However, we also design two new reduction rules to reduce the computational burden.

7.5.1 Probabilistic Model Checking-Based Approach

For Problem OptC-1, the general idea of the PMC-based approach is to encode this problem as a PMC problem. We here use PRISM (a probabilistic symbolic model checker) [171] to express and solve the encoded PMC problem. First, we model the DA-PBN with the control setting as a PRISM model (see Definition 7.5.1). Then, we formulate a Probabilistic Computation Tree Logic (PCTL) property φ corresponding to the control aim. In this case, the PCTL is defined as

$$\text{Pmax=?}\left[\text{F} = M(x_1 = x_1^{des}\&...\&x_n = x_n^{des})\right].$$

Note that we can easily describe multiple desired states. By the semantics of PCTL, the result of verifying φ is equivalent to the result of Problem OptC-1.

Definition 7.5.1 (PRISM model for Problem OptC-1)**.** *Given a DA-PBN $\mathcal{DP}$, the initial state x^{ini}, the desired state x^{des}, and the control time M. Each Boolean function $f_j^{(i)}$ of $\mathcal{DP}$ is transformed into a polynomial on the real number field (denoted by $\hat{f}_j^{(i)}$) by using Lemma 7.4.1. First, the given system is described as a Markov Decision Process (MDP).*

Then, module iNode $i, i = 1, 2, ..., n$ *is described by*

$$\begin{aligned}
&\textit{module } iNodei\\
&\ x_i : [0..1];\\
&\ [iNode1](mod(t, p_i) = q_i) \rightarrow c_1^{(i)} : (x_i' = \hat{f}_1^{(i)}(x, u)\\
&\qquad\qquad + ... + c_{l_i}^{(i)} : (x_i' = \hat{f}_{l_i}^{(i)}(x, u));\\
&\ [iNode1](mod(t, p_i)\ != q_i) \rightarrow 1.0 : (x_i' = x_i);\\
&\textit{endmodule.}
\end{aligned}$$

Next, module eNode $i, i = 1, 2, ..., m$ *is described by*

$$\begin{aligned}
&\textit{module } eNodei\\
&\ u_i : [0..1];\\
&\ [iNode1]true \rightarrow (u_i' = 0);\\
&\ [iNode1]true \rightarrow (u_i' = 1);\\
&\textit{endmodule.}
\end{aligned}$$

Next, module time described as

$$\begin{aligned}
&\textit{module time}\\
&\ t : [0..(\gamma - 1)];\\
&\ [iNode1]true \rightarrow (t' = mod(t + 1, \gamma));\\
&\textit{endmodule.}
\end{aligned}$$

Finally, the initial state is described by

$$init\ x_1 = x_1^{ini} \& ... \& x_n = x_n^{ini} \& t = 0\ endinit.$$

For Problem OptC-2, we reuse the PRISM model for Problem OptC-1. However, we slightly modify the PCTL property φ as

$$\text{Pmin=?}\left[\text{F} = M(x_1 = x_1^{des} \& ... \& x_n = x_n^{des})\right].$$

By the semantics of PCTL, the result of verifying φ is equivalent to the result of Problem OptC-2.

For Problem OptC-3, we need substantial modifications. The general idea is to add state rewards to the PRISM model for Problem OptC-1 to express the cost function; then to use the property on reachability rewards to find the minimum expected cost of the DA-PBN. Specifically, the modified model for Problem OptC-3 is shown in Definition 7.5.2. The new PCTL property is

$$\text{Rmin=?}\left[\text{F}(t = M + 1)\right].$$

By the definition of expected values of rewards, R_{min} (i.e., the minimum expected reward) is equivalent to J_{min} of the DA-PBN.

Definition 7.5.2 (PRISM model for Problem OptC-3)**.** *We modify the PRISM model for Problem OptC-1 (see Definition 7.5.1) as follows. First, we replace module time by the new one as*

$$\begin{aligned}
&\textit{module time}\\
&\ t : [0..(M+1)];\\
&\ [iNode1] t < (M+1) \rightarrow (t' = t+1);\\
&\ [iNode1] t = (M+1) \rightarrow (t' = M+1);\\
&\textit{endmodule.}
\end{aligned}$$

Since we need to express states at time t ($t = 0, ..., M$), we do not use γ and % here. We use $M+1$ because we must consider $x(M)$. Then, we add a reward item as

$$\begin{aligned}
&\textit{rewards "cost"}\\
&\ t < M : \{\mathcal{Q}x(t) + \mathcal{R}u(t)\};\\
&\ t = M : \mathcal{Q}_f x(t);\\
&\textit{endmodule.}
\end{aligned}$$

Regarding verifying the PTCL property, PRISM also builds the transition probability matrix of the model. However, it uses more efficient techniques (e.g., multi-terminal decision diagrams, sparse-matrix construction) for this task [176].

7.5.2 Stochastic Satisfiability Modulo Theory-Based Approach

For Problem OptC-1, the general idea of the SSMT-based approach is to encode this problem as an SSMT formula Φ. By the semantics of SSMT [172], the maximum reachability probability is equivalent to the satisfaction probability of Φ (denoted by $Pr(\Phi)$). We then use SiSAT [177], an SSMT solver, to compute $Pr(\Phi)$.

Definition 7.5.3 (SSMT formula for Problem OptC-1)**.** *Given a DA-PBN $\mathcal{DP}$, an initial state x^{ini}, a desired state x^{des}, a target time M. We construct an SSMT formula Φ as follows:*

$$\begin{aligned}
\Phi &:= \mathcal{Q} : \varphi, \varphi := \varphi_{ini} \wedge \varphi_{des} \wedge \varphi_{trans},\\
\varphi_{ini} &:= \bigwedge_{i=1}^{n} \left\{x_i^0 \leftrightarrow v_i^{ini}\right\}, \varphi_{des} := \bigwedge_{i=1}^{n} \left\{x_i^M \leftrightarrow v_i^{des}\right\},\\
\varphi_{trans} &:= \bigwedge_{t=0}^{M-1} \mathcal{T}(x^t, x^{t+1}), \mathcal{T}(x^t, x^{t+1}) := \bigwedge_{i=1}^{n} \mathcal{T}_i(x^t, x^{t+1}),\\
\mathcal{T}_i(x^t, x^{t+1}) &:= \left[(t\%p_i \neq q_i) \wedge (x_i^{t+1} \leftrightarrow x_i^t)\right] \vee\\
&\qquad \left[(t\%p_i = q_i) \wedge \bigvee_{j=1}^{l_i} \{(c_i = j) \wedge (x_i^{t+1} \leftrightarrow f_j^{(i)}(x^t, u^t))\}\right],\\
\mathcal{Q}_\exists &:= \left(\exists u_1^0 \in \mathbb{B} ... \exists u_m^0 \in \mathbb{B}\right) ... \left(\exists u_1^{M-1} \in \mathbb{B} ... \exists u_m^{M-1} \in \mathbb{B}\right),\\
\mathcal{Q}_\mathcal{Y} &:= \mathcal{Y}_{d_{c_1}} c_1 \in \{1, ..., l_1\} ... \mathcal{Y}_{d_{c_n}} c_n \in \{1, ..., l_n\}.
\end{aligned}$$

Definition 7.5.3 shows the encoded SSMT formula for Problem OptC-1. Herein, variable $x_i^j \in \mathbb{B}$ expresses the value of internal node x_i at time j. Variable $u_i^j \in \mathbb{B}$ expresses the value of external node u_i at time j. Variable $c_i \in \{1, ..., l_i\}$ expresses which Boolean function is chosen for the update of internal node x_i. $\exists$, $\forall$, and $\mathfrak{R}$ denote the existential, universal, and randomized quantifiers, respectively [172]. d_{c_i} is the probability distribution of c_i such that $d_{c_i}(j) = c_j^{(i)}, j \in \{1, ..., l_i\}$. φ_{ini} and φ_{des} express the conditions for the initial and desired states, respectively. By the syntax of SSMT, we can express the condition for multiple initial or desired states. φ_{trans} expresses state transitions of the DA-PBN. Note that if $l_i = 1$, then we can remove the randomized variable c_i from $\mathcal{Q}_{\mathfrak{R}}$ and remove the $(c_i = j)\wedge$ part from $\mathcal{T}_i(x^t, x^{t+1})$.

For Problem OptC-2, we only need to slightly modify the encoded SSMT formula for Problem OptC-1. Specifically, we replace only $\mathcal{Q}_\exists$ by $\mathcal{Q}_\forall$ that is defined as

$$\left(\forall u_1^0 \in \mathbb{B} ... \forall u_m^0 \in \mathbb{B}\right) ... \left(\forall u_1^{M-1} \in \mathbb{B} \ ... \ \forall u_m^{M-1} \in \mathbb{B}\right).$$

By the semantics of SSMT, the minimum reachability probability of Problem OptC-2 is equivalent to $Pr(\Phi)$. In addition, if we want only to know whether the system is safe or not, then we can use thresholding to prune the search space. For example, we can use the built-in option of SiSAT as "–ut=ε" [177], i.e., the upper target threshold is ε.

For Problem OptC-3, the general idea is to use the conditional expectation semantics of SSMT. Since only the maximum conditional expectation is defined as well as is supported in SiSAT, we modify the encoded SSMT formula Φ for Problem OptC-1 as follows. First, we add to Φ a new free variable y corresponding to the cost function J of Problem OptC-3. However, we need to set y as $-\gamma$ (where $J = E[\gamma | x(0) = x^{ini}]$) because we need to find the minimum expected cost. This means that

$$y = -\left\{\sum_{k=0}^{M-1} \left(\mathcal{Q}x(k) + \mathcal{R}u(k)\right) + \mathcal{Q}_f x^M\right\}.$$

Let $\sigma^+(X)$ and $\sigma^-(X)$ denote the sum of all positive and negative elements of a vector X of real numbers, respectively. Then, we specify the upper and lower bounds for the cost variable as

$$u_y = -\left[\sigma^-(\mathcal{Q}) \times M + \sigma^-(\mathcal{R}) \times M + \sigma^-(\mathcal{Q}_f)\right]$$

and

$$l_y = -\left[\sigma^+(\mathcal{Q}) \times M + \sigma^+(\mathcal{R}) \times M + \sigma^+(\mathcal{Q}_f)\right],$$

respectively. Next, we also need to remove φ_{des} from φ. By the semantics of the maximum conditional expectation, the minimum expected cost of Problem OptC-3 is equivalent to $-E_y(\Phi)$, where $E_y(\Phi)$ is the maximum conditional expectation of the SSMT formula Φ with respect to variable y. Finally, we use SiSAT [177] to compute $E_y(\Phi)$.

In contrast to the PMC-based approach, the proposed SSMT-based approach builds neither implicit nor explicit transition probability matrices. It exploits the efficient solving techniques developed for SSMT solvers. As mentioned above, SiSAT supports thresholding that may early exclude redundant parts of the search space in Problem OptC-2. PRISM always builds transition probability matrices first. Moreover, PRISM does not

support such kind of thresholding in verifying the PCTL property φ. Hence, this is also another advantage of the SSMT-based approach as compared to the PMC-based approach. Furthermore, if $l_i \leq 2$, $\forall i \in \{1, ..., n\}$, the encoded SSMT formula of Problem OptC-1 can be seen as a Stochastic Satisfiability (SSAT) formula. In this case, we can apply recent advents in SSAT solvers (e.g., see [178]) to Problem OptC-1.

7.5.3 Polynomial Optimization Problem-Based Approach

Since p_i and q_i are fixed, and k is given, $E[x_i(k+1)]$ can be expressed as a polynomial (see Theorem 7.4.6). Thus, Problem OptC-3 is equivalent to Problem 7.5.1. In this encoded POP, we see that $E[x(k+1)] \in [0,1]^n$ automatically holds; therefore, we set $E[x(k+1)] \in \mathbb{R}^n$. Technically, this may accelerate the computational time for solving this problem. In addition, the constraint $u_i(k)(u_i(k)-1) = 0$ guarantees that $u_i(k)$ is a binary variable. Since this constraint is non-convex, its existence may make the solving time longer. In a practical manner, we can use the relaxed constraint $0 \leq u_i(k) \leq 1$ instead. Finally, we use SparsePOP [173] to solve the encoded POP.

Problem 7.5.1.

$$
\begin{aligned}
&\textit{find } E[x(0)] \in \mathbb{R}^n, E[x(k+1)] \in \mathbb{R}^n, u(k) \in \mathbb{R}^m, \\
&\qquad k = 0, 1, ..., M-1, \\
&\textit{min } J, \\
&\textit{subject to } \textit{System } \Sigma_i(k), i = 1, 2, ..., n, \\
&\qquad E[x(0)] = x^{ini}, \\
&\qquad u_j(k)(u_j(k)-1) = 0, j \in \{1, ..., m\}.
\end{aligned}
$$

In contrast to the PMC-based approach, the proposed POP-based approach builds neither implicit nor explicit transition probability matrices. It exploits the efficient solving techniques developed for POP solvers. This is an advantage as compared to the PMC-based approach. However, it is hard to apply the POP-based approach to Problem OptC-1 or OptC-2. Furthermore, we propose two reduction rules to reduce the number of decision variables of the encoded POP, which largely affects the performance of the POP-based approach.

The first rule is based on the cost function J. In the encoded POP, we find the set of decision variables that properly contribute to J. We then remove all decision variables that are not in this set from the encoded POP. Note that we also must remove all the equations related to the removed decision variables. For example, consider the instance of Problem OptC-3 shown in Example 7.3.2. With $J = E[x_3(2)]$, we only need to consider variables $E[x_3(2)]$, $E[x_1(1)]$, $E[x_2(1)]$, $E[x_3(0)]$, and $E[x_1(0)]$. Since the number of decision variables (resp. equations) of Problem 7.5.1 is $n \times (M+1) + m$ (resp. $n \times (M+1) + m$), the time for applying the cost-based reduction rule is polynomial.

The second rule is based on the context of the DA-PBN. Because of introducing the context (i.e., p's and q's), the encoded POP may contain some equations in form $E[x_i(t+1)] - E[x_i(t)] = 0$ (when $t\%p_i \neq q_i$). Now, we can remove decision variable $E[x_i(t+1)]$ and the above equation. Then, we must replace $E[x_i(t+1)]$ by $E[x_i(t)]$ everywhere $E[x_i(t+1)]$ appears in the encoded POP. The time for applying this reduction rule is also polynomial.

7.5.4 Remarks

First, we consider the issue of showing the control sequence when solving the problems. From the output of SparsePOP, we can directly obtain the control sequence that leads to the minimum expected cost. However, this task is difficult for the case of PRISM and SiSAT due to their implementing limitations.

Second, we can consider other forms of the cost function in Problem OptC-3. For example, the weighting vectors can be replaced by functions of x and u [21, 72]. By the expressive power of PRISM, SiSAT, or SparsePOP, we can easily modify the proposed approaches to handle a new form of the cost function.

Finally, we can consider adding hard constraints (i.e., adding an upper bound H for the number of controls that can be applied to the network) into the problems. The number of controls applied to the network is defined in [71] as

$$\sum_{t=0}^{M-1}\sum_{i=1}^{m}|u_i(t)-u_i(t+1)|.$$

The introduction of hard constraints is important for medical applications because the number of treatments such as radiation and chemo-therapy is usually limited [71]. To handle hard constraints, we only need some little modifications to the SSMT-based and POP-based approaches because these approaches consider the values of control nodes at each time step. We have that

$$|u_i(t)-u_i(t+1)| = (u_i(t)-u_i(t+1))^2,$$

since

$$|u_i(t)-u_i(t+1)| \in \mathbb{B}.$$

For the SSMT-based approach, we can, for example, modify φ to

$$\varphi_{ini} \wedge \varphi_{des} \wedge \varphi_{trans} \wedge \varphi_{hard},$$

where

$$\varphi_{hard} \equiv \sum_{t=0}^{M-1}\sum_{i=1}^{m}(u_i(t)-u_i(t+1))^2 \leq H.$$

For the POP-based approach, we can, for example, add the inequality

$$\sum_{t=0}^{M-1}\sum_{i=1}^{m}(u_i(t)-u_i(t+1))^2 \leq H$$

to the encoded POP. On the other hand, it is difficult to modify the PRISM-based approach to handle hard constraints. The reason is due to the expressive power of PCTL.

Table 7.2: DA-PBN models of the WNT5A network.

Gene	Node	Boolean function (Probability)
pirin	u_1	
WNT5A	x_1	$\neg x_5$ (1.0)
S100P	x_2	$\neg x_6$ (0.8), x_2 (0.2)
RET1	x_3	x_3 (1.0)
MART1	x_4	$\neg x_6 \vee u_1$ (1.0)
HADHB	x_5	$x_2 \vee x_3$ (1.0)
STC2	x_6	$x_6 \vee \neg u_1$ (0.8), x_6 (0.2)

7.5.5 Case Study

We consider a WNT5A network [179], which is related to melanoma. The DA-PBN model of this network is given in Table 7.2. Note that the second Boolean function for x_2 or x_6 is obtained by applying the synchronous and asynchronous semantics of BNs [69]. Since it is hard to determine the context of the DA-PBN, we randomly generate the context with varying Λ ($\Lambda = 1$ and $\Lambda = 3$). Now, we can obtain two DA-PBN models expressing the WNT5A network.

In the WNT5A network, it is important to inhibit the concentration level of x_1 (the gene WNT5A) [179]. From the obtained DA-PBN models, we consider solving Problems OptC-1, OptC-2, and OptC-3 with the control setting capturing this fact:

$$
\begin{aligned}
&x^{ini} = (1,1,0,1,0,0)^\top, x^{des} = (0,-,-,-,-,-)^\top,\\
&M = 5,\\
&\mathcal{Q} = (1,0,0,0,0,0), \mathcal{R} = (1), \mathcal{Q}_f = (10,0,0,0,0,0),
\end{aligned}
$$

where $-$ means an arbitrary Boolean value.

We then apply the proposed approaches to optimal control of the WNT5A network. Regarding Problems OptC-1 and OptC-2, the results are the same for both the PMC-based and SSMT-based approaches. The maximum reachability probability (say P_{max}) is always equal to 1 for all the two DA-PBNs. The minimum reachability probability (say P_{min}) is 0.104 (resp. 0.36) for the first DA-PBN with $\Lambda = 1$ (resp. the second DA-PBN with $\Lambda = 3$). Therefore, it is possible to drive the DA-PBN from the initial state to the desired state because P_{max} is always equal to 1 and P_{min} is always greater than 0. This means that the inhibition of the gene WNT5A can be controlled by manipulating the value of the control node (the gene pirin). Regarding Problem OptC-3, the results are the same for all the approaches including the PMC-based approach, the SSMT-based approach, the POP-based approach with reduction, and the POP-based approach without reduction. The minimum expected cost (say J_{min}) is 4 (resp. 3) for the first DA-PBN (resp. the second DA-PBN). Specifically, for all the two DA-PBNs, we obtain $E[x_1(1)] = 1$, $E[x_1(2)] = 1$, $E[x_1(3)] = 0$, $E[x_1(4)] = 0$, and $E[x_1(5)] = 0$. Hence, we see that the concentration level of the gene WNT5A is inhibited with time. In addition, the minimum cost without control is 7 for all the two DA-PBNs. Therefore, the control objective is achieved.

7.6 Experiments

We conducted experiments on random DA-PBNs to evaluate the performance of the proposed approaches for the three optimal control problems. First, we randomly generated DA-PBNs according to the following factors:

$$n \in \{30, 50\}, m \in \{1, 2\}, \Lambda = 4, l \in \{1, 4\},$$

where l is the number of l_i such that $l_i > 1$. The control setting is given as follows:

$$x^{ini} = (0, 0, ..., 0)^\top, \mathcal{Q} = (0, ..., 0)^n, \mathcal{R} = (1, ..., 1)^m,$$
$$M \in \{4, 6, 8, 10, 12, 14, 16\}.$$

For x^{des}, we randomly chose k^{des} internal nodes and set them either 0 or 1, whereas the remaining nodes can receive arbitrary Boolean values. For simplify, we set $k^{des} = 2$. We say the set of k^{des} internal nodes as V^{des}. For $\mathcal{Q}_f$, $\mathcal{Q}_f(x_i) = 10$ if $x_i \in V^{des}$ and $x_i^{des} = 0$. $\mathcal{Q}_f(x_i) = 1$ if $x_i \in V^{des}$ and $x_i^{des} = 1$. Otherwise, $\mathcal{Q}_f(x_i) = 0$.

We then applied the proposed approaches to the randomly generated instances of the three optimal control problems. All the experiments were run on a virtual machine whose environment is CPU: Intel(R) Core(TM) i7-3630QM 2.40GHz x 4, Memory: 8 GB, Ubuntu 18.04.2 64 bit. The time limit is two hours for each instance. For each instance, we reported the running time of each proposed approach.

Note that there are some special cases in the running of SiSAT. For Problem OptC-1 (resp. OptC-2), when the computed probability corresponding to an assignment to all the existential (resp. universal) variables is 1 (resp. 0), SiSAT can immediately stop and return 1 (resp. 0) as the satisfaction probability. For both Problems OptC-1 and OptC-2, if Φ is not satisfiable for all assignments to all the existential or universal variables, SiSAT can quickly return 0 as the satisfaction probability. In all the instances of Problem OptC-1 (resp. Problem OptC-2), P_{max} (resp. P_{min}) is always 0 because Φ is not satisfiable. To avoid this special case, we change Λ to 1 for all the instances of Problems OptC-1 and OptC-2. For all the instances of Problems OptC-3, Λ is still 4.

With the new setting, the results of all the instances of Problem OptC-2 are still 0. Hence, we only analyze the results of the instances of Problem OptC-1 and Problem OptC-3. The obtained insights for Problem OptC-2 can be similarly deduced from those for Problem OptC-1 because in general cases, i.e., $Pr(\Phi) \in (0, 1)$, the SSMT-based approach must traverse all the assignments of all the quantified variables. For the PMC-based approach, the PRISM models for Problem OptC-1 and Problem OptC-2 are the same. In addition, the time complexity for checking the PTCL formula for Problem OptC-1 is the same as that for checking the PCTL formula for Problem OptC-2.

Note that the instances of Problem OptC-1 and Problem OptC-3 used in the experiment can be representative for a vast number of similar case studies because of the following reasons. First, n is large enough, ensuring that the number of reachable states as well as the transition probability matrix is large. Second, the result of Problem OptC-1 is in (0, 1) in most cases, indicating that the special cases of SiSAT did not occur in most cases.

From the experimental results, we shall show several analysis on how the running time of the proposed approaches depends on the factors and a comparison among the proposed approaches.

7.6.1 Experimental Results on Problem OptC-1

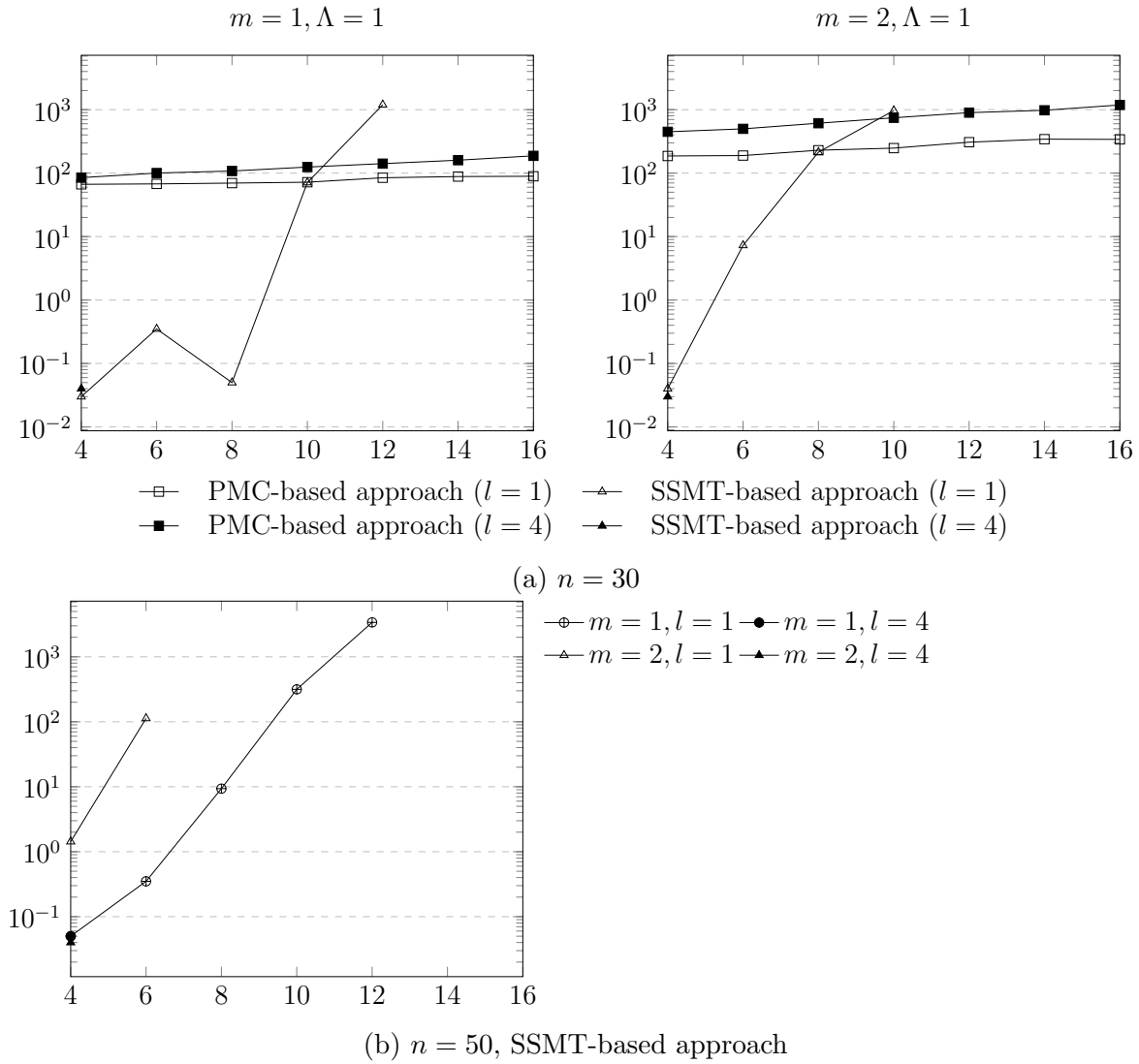


Figure 7.2: Experimental results on Problem OptC-1. The x-axis denotes the target time M, whereas the y-axis denotes the running time (in seconds) with a logarithmic scale of base 10.

Figure 7.2 shows the results on Problem OptC-1. First, the running time of the PMC-based approach linearly increases as M increases. In the PRISM model of Problem OptC-1 (see Subsection 7.5.1), M does not appear; thus, M does not affect the time for building the transition probability matrix. The complexity of the algorithm for solving a reachability formula is linear in M [180]. Hence, we have that the running time of the PMC-based approach is linear in M.

Second, the running time of the PMC-based approach for the case $l = 1$ is less than that for the case $l = 4$. When l increases, the number of possible Boolean functions for node updating also increases. As a consequence, the number of reachable states of the model may increase; leading the running time may increase.

Third, for the case $n = 50$, the PMC-based approach met the OutOfMemory (OOM) error in all instances. This is the reason why we only show the results for the SSMT-

based approach in Figure 7.2b. The number of reachable states of the PRISM model may increase exponentially as n increases. When $n = 50$ (a large number), the number of reachable states of the PRISM model as well as the size of the transition probability matrices may be very large; leading OOM. Practically, we can increase the memory size but the time for building the transition probability matrices and verifying the PCTL property may be extremely long.

Last, the running time of the SSMT-based approach exponentially increases as M increases. Even for the cases $m = 1, l = 4$ and $m = 2, l = 4$, the SSMT-based approach met timeout when M is only 6. The reason is as follows. The quantified variables of the encoded SSMT formula include $m \times M$ existential variables and $l \times M$ randomized variables. In general cases, the time for solving the SSMT formula is exponential in its number of quantified variables [172]. The maximum probability is in (0, 1) in most cases. Moreover, the algorithmic enhancements for SiSAT may be not effective. For example, since we want to get the maximum satisfaction probability, thresholing is not effective in this case. Due to the above reason, the SSMT-based approach can only handle the case of small number of quantified variables.

In addition, we also obtain a comparison between the PMC-based approach and the SSMT-based approach in terms of Problem OptC-1. In the case of small number of quantified variables (e.g., $m = 1, l = 1, M \leq 10$ or $m = 2, l = 1, M \leq 8$), the SSMT-based method is better than the PMC-based approach. In this case, the number of quantified variables is moderate; thus, the SSMT-based approach can obtain better performance because of the efficiency of solving algorithms for SSMT. When $n = 50$, the SSMT-based approach outperforms the PMC-based approach. In this case, the PMC-based approach meets OOM because the number of reachable states is too large. Whereas, the SSMT-based approach can work well because the number of quantified variables is small and in particular does not depends on n. When $n = 30$ and the number of quantified variables is large, the PMC-based approach outperforms the SSMT-based approach. In this case, the PMC-based approach did not meet OOM and its running time is only linear in M. Whereas, the running time of the SSMT-based method is exponential in M.

7.6.2 Experimental Results on Problem OptC-3

Figures 7.3 and 7.4 show the results on Problem OptC-3 with $n = 30$ and $n = 50$, respectively. Regarding the PMC-based approach for Problem OptC-3, we first see that the running time of the PMC-based approach polynomially increases as M increases. The reason is as follows. In the PRISM model of Problem OptC-3 (see Subsection 7.5.1), t takes a value from 0 to $M+1$. Hence, the number of reachable states linearly increases as M increases. In addition, the complexity of the method for solving a reward reachability formula is cubic in the size of the system [180]. Second, for the case $n = 30, m = 2, l = 4, \Lambda = 4$, the PMC-based approach met OOM in all the cases of M (except $M = 4$). The number of reachable states of the PRISM model is exponential in m and linear in M. With these parameters, the number of reachable states as well as the size of the transition probability matrix may be very large; leading OOM. Third, the running time of the PMC-based approach for the case $l = 1$ is less than that for the case $l = 4$. Fourth, for the case $n = 50$, the PMC-based approach met OOM in all instances. Note that the two later observations are similar to those for Problem OptC-1.

Regarding the SSMT-based approach for Problem OptC-3, we first see that the running

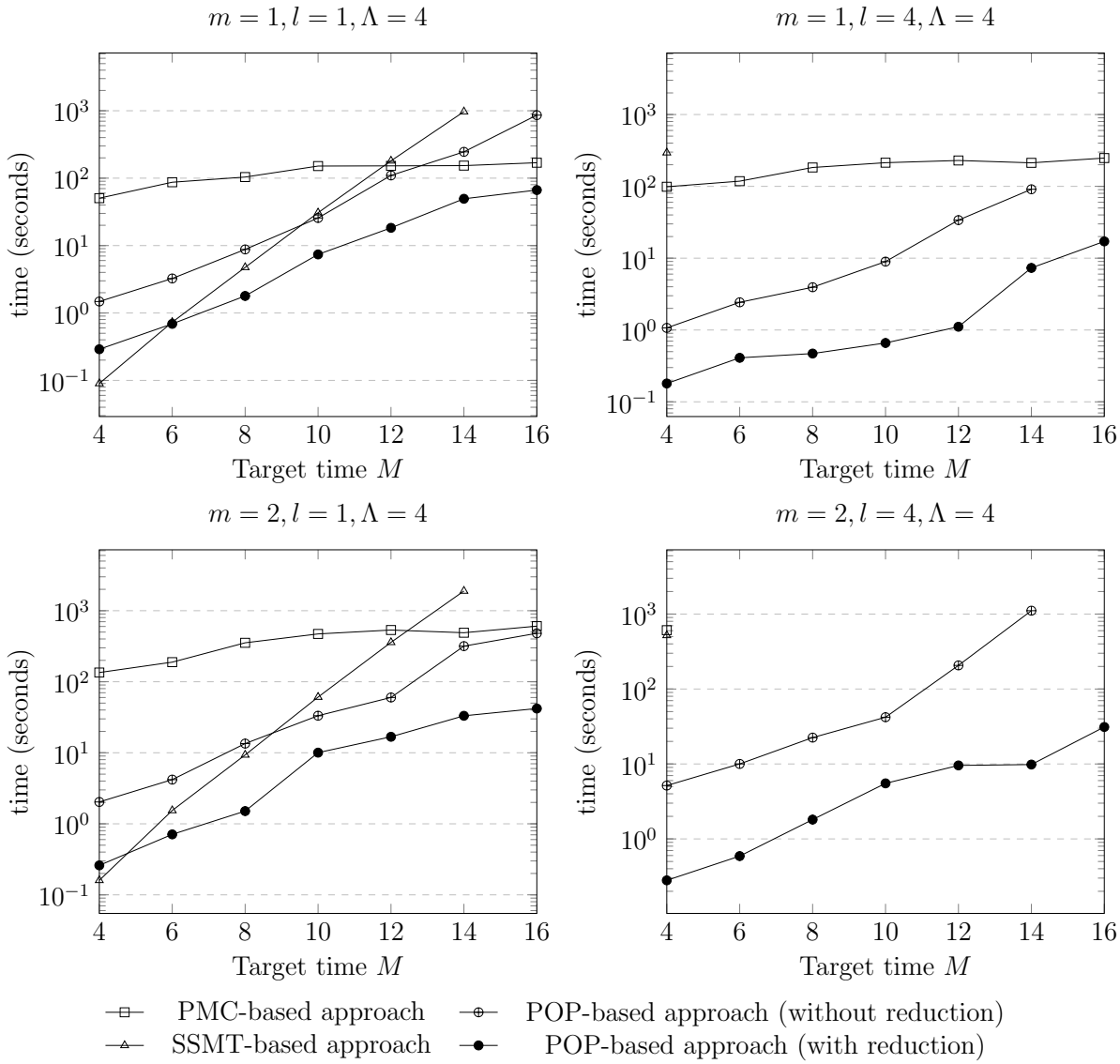


Figure 7.3: Experimental results on Problem OptC-3 with $n = 30$.

time of the SSMT-based approach exponentially increases as M increases. Second, the SSMT-based approach can only handle the case of small number of quantified variables. For the cases $m = 1, l = 1$ and $m = 2, l = 1$, the maximum M that the SSMT-based approach can handle is 14. The cases $m = 1, l = 4$ and $m = 2, l = 4$ are only 4. However, in this case, the SSMT-based approach can still handle the instances with $n = 50$. The reasons for the above observations are similar to those for Problem OptC-1.

Regarding the POP-based approach for Problem OptC-3, we first see that the running time of the POP-based approach without reduction exponentially increases as M increases. The reason is that the number of decision variables of the encoded POP is linear in M and the running time of SparsePOP may exponentially increase as this number increases [173]. Second, the POP-based approach with reduction is much better than the POP-based approach without reduction (the speedup is significant). However, the running time of the POP-based approach with reduction still exponentially increases as M increases. Clearly, the two reduction rules reduce significantly the number of decision

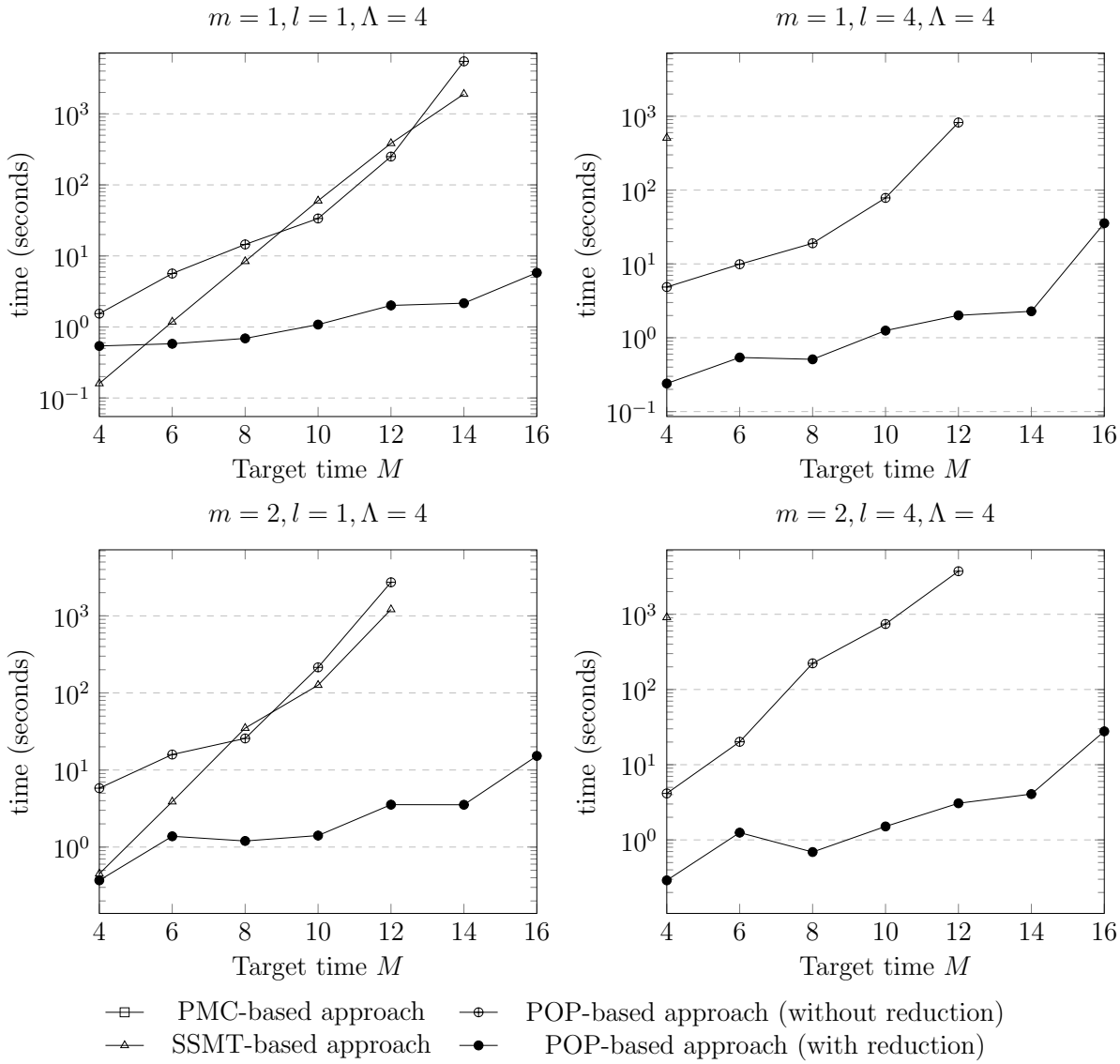


Figure 7.4: Experimental results on Problem OptC-3 with $n = 50$.

variables as well as the number of equations of the encoded POP. However, the number of decision variables of the encoded POP may be still linear in M. We also note that the result of the POP-based approach is different from that of the PMC-based or SSMT-based approach in some cases. The reason is that PRISM and SiSAT are exact tools; whereas, SparsePOP is only an approximation tool [173].

Finally, we make a comparison among the three proposed approaches in terms of Problem OptC-3. In the case of small M (e.g., $M \leq 12$), the POP-based approach (even without reduction) is much better than the PMC-based approach. In this case, the size of POP is moderate. Thus, the POP-based approach (even without reduction) may be much better than the PMC-based approach because it avoids building the transition probability matrix that may be very large and time-consuming to compute when n is large. In the case of large M (e.g., $M \geq 14$), the PMC-based approach gradually becomes better than the POP-based approach (even with reduction). The reason is, as explained above, that the running time of the PMC-based approach linearly increases as M increases, whereas

the running time of the POP-based approach (with or without reduction) exponentially increases as M increases. In the case of small number of quantified variables (e.g., $m = 1, l = 1, M \leq 10$ or $m = 2, l = 1, M \leq 8$), the SSMT-based approach is better than the PMC-based approach, is comparable to the POP-based approach without reduction, and is worse than the POP-based approach with reduction. In this case, the number of quantified variables of the encoded SSMT formula is moderate; thus, the SSMT-based approach may get better performance because of the efficient solving algorithms for SSMT.

7.6.3 Summary of the Experimental Results

To sum up, the experimental results confirm the advantages and disadvantages of each proposed approach. For the PMC-based approach, the running time is linear or polynomial in M. However, it may meet OOM or take extremely long time when the number of reachable states of the PRISM model is too large. Moreover, the number of reachable states is exponential in n and m. For the SSMT-based approach, the running time is exponential in M. However, it can handle the case of large n when the number of quantified variables is small. For the POP-based approach, the POP-based method with reduction gives the best performance overall, but the running time is still exponential in M. Moreover, the result of the POP-based method is not exact in some cases. These insights suggest that the proposed approaches can complement each other.

7.7 Discussion

In this chapter, we have formulated three meaningful optimal control problems of DA-PBNs (i.e., Problems OptC-1, OptC-2, and OptC-3) based on different aims of control. For theoretical aspects, we have shown the hardness of the problems. Specifically, we have proved that the three optimal control problems are NP^{PP}-hard even with the restriction that Boolean functions of the DA-PBN are in 2CNF. In addition, we have also proved that all three problems are in PSPACE. For practical aspects, we have proposed three approaches to solve the problems. The PMC-based and SSMT-based approaches can be applicable for all the problems, whereas the POP-based approach can only be applicable for Problem OptC-3. In particular, the SSMT-based and POP-based approaches avoid building transition probability matrices of the considered DA-PBN. For the POP-based approach, we have also proposed two reduction rules to reduce the number of decision variables of the encoded POP. We note that our proposed approaches open a chance to apply various recent advents in many research fields (e.g., probabilistic model checking [181, 182], stochastic satisfiability [178, 183], polynomial optimization [184, 185]) to optimal control of DA-PBNs.

We have then applied the proposed approaches to optimal control of a real biological network to show their applications. We have also conducted experiments to evaluate the performance of the proposed approaches. From the experimental results, we have presented theoretical and experimental analysis on the effects of some factors (e.g., n, M) to the performance of the proposed approaches as well as a comprehensive comparison among them. The obtained analysis suggests that each of the proposed approaches works well for specific cases of control settings; hence, they can complement each other.

In the future, we plan to improve the proposed approaches to handle large-scale problem instances (e.g., the case of large networks or the case of long target time M). Poten-

tially, we can use the assume-guarantee verification technique [181] for the PMC-based approach, the Craig interpolation technique [183] for the SSMT-based approach, or linear programming formulation technique [184] for the POP-based approach. We also plan to extend the proposed approaches for DA-PBNs to those for hybrid models of DA-PBNs [68, 186]. It seems promising because both PRISM and SiSAT support probabilistic hybrid systems. Finally, it is important to consider the problem of calculating a steady-state distribution on attractors of a DA-PBN [187].

Chapter 8

Conclusions and Future Work

8.1 Conclusions

Boolean networks are simple but efficient mathematical formalism that has widely been applied in various research fields, such as, systems biology, mathematics, neural networks, social modeling, robotics, and computer science. Attractor detection and optimal control of BNs are crucial but challenging problems. Hence, they have recently attracted much attention from many research communities. However, we are still facing profound challenges such as scalability or high computation cost for large-scale networks, the lack of practical methods for complex types of BNs (e.g., GABNs, DGABNs, DA-PBNs), the lack of theoretical results linking dynamics between different types of BNs. In this dissertation, we have worked on fulfilling these challenges.

In theory, we have introduced a number of new theoretical results that contribute to the understanding of the dynamics of BNs. Specifically, in Chapter 3, we have formally stated and proved the relations between the dynamics of a GABN and that of its SBN (or ABN) counterpart. In Chapter 5, we have introduced a new concept called extended state transition graph for capturing the whole dynamics of a DGABN. Based on this new concept, we have formally stated and proved several relations in dynamics between DGABNs and other conventional models including DA models, BSBNs, GABNs, and MxBNs. In particular, we have shown that the relations presented in Chapter 3 and 5 pave the potential ways to analyze different types of BNs as well as many other popular models. In Chapter 4, we have discovered several relations between the dynamics of a BN and its network structures. More specifically, we have formally stated and proved several lemmas and theorems on relations between the dynamics of a BN and an FVS of its interaction graph. Furthermore, we have also obtained a new theorem on relations between the dynamics of an ABN and an NFVS of its interaction graph. Note that these new findings are the theoretical foundations for our efficient methods for finding attractors of an ABN. Finally, in Chapter 7, we have discovered the computational complexity of three meaningful optimal control problems of DA-PBNs. Specifically, we have proved that all the three problems (also their restricted versions) are NP^{PP}-hard and in PSPACE.

In practice, we have developed several algorithms and methods for attractor detection and optimal control of different typical types of BNs. Note that these algorithms and methods are mainly based on the new theoretical results obtained in this research. In Chapter 3, we have proposed three BDD-based algorithms and one SAT-based algorithm for finding attractors of a GABN based on the attractors of the SBN counterpart of the

GABN. These algorithms are first analytical and practical methods for analyzing GABNs. In particular, the experimental results on various classes of networks show that the SAT-based algorithm outperforms the three BDD-based methods and can handle large GABNs. Inspiring by the obtained relations between GABNs and ABNs as well as the observation that the number of attractors of an ABN is equal to that of its GABN counterpart in most cases, we have developed an efficient method for approximating attractors of an ABN, which uses the SAT-based algorithm as a subroutine. The experimental results on real biological networks are promising because the approximation method outperforms the two state-of-the-art methods as well as can handle networks of up to 101 nodes.

In Chapter 4, we have proposed an efficient method for exactly computing all the attractors of an ABN. The theoretical foundation of this method is the obtained relations between the dynamics of a BN and an FVS of its interaction graph (Theorems 4.3.1 and 4.3.2). This method is then enhanced with two substantial improvements. In the first improvement, we have explored several techniques for checking the reachability in ABNs to develop an efficient combination. In the second improvement, the improved method uses an NFVS instead of an FVS to get a candidate set of states. The theoretical foundation of this improvement is the obtained relation between the dynamics of an ABN and an NFVS of its interaction graph (Theorem 4.6.2). Now, the improved method outperforms the state-of-the-art methods and can handle very large networks with up to 1000 nodes in terms of randomly generated networks and more than 300 nodes in terms of real biological networks. In particular, the principle of this method can be applied to many other types of BNs and pave potential ways to improve itself.

In Chapters 5 and 6, relying on our proposed concept (i.e., extended state transition graph), we have proposed one SMT-based method and two SMT-based methods for attractor detection and optimal control of DGABNs, respectively. Note that there is no previous method specifically designed for DGABNs. Although they are extensions of the corresponding previous SAT-based methods for SBNs, we have demonstrated that they contain substantial differences from the previous ones. Furthermore, the experimental results on randomly generated networks and artificial networks show that these proposed methods are efficient and can handle large-scale networks. Inspiring by the developed methods for optimal control of DGABNs, in Chapter 7, we have proposed various approaches for solving the three optimal control problems of DA-PBNs, a stochastic extension of DGABNs and a contextual extension of SPBNs. The SSMT-based approach is completely new, whereas the PMC-based and POP-based approaches are (non-trivial) extensions of the corresponding previous approaches for optimal control of SPBNs. In particular, the POP-based approach is enhanced by the two proposed reduction rules. The experimental results on artificial networks show that (1) the two reduction rules are effective, (2) these proposed approaches can handle large problem instances in terms of optimal control of DA-PBNs, (3) their performance is impacted by multiple parameters, and (4) they can complement each other. Finally, we have developed software tools for all the proposed algorithms, methods, and approaches for attractor detection and optimal control of different typical types of BNs.

Especially, the principle that we developed in Chapter 4 for attractor detection in ABNs can be generalized as a blueprint for attractor detection in various types of BNs beyond ABNs. Figure 8.1 shows the description of this blueprint. First, we try to get a candidate set of states that covers all or nearly all the attractors of a BN. There may be many ways to get a candidate set. For example, in Chapter 4, we use an FVS or an

NFVS to get the candidate set for the ABN, whereas in Chapter 3, we use the attractors of an SBN to get the candidate set for its GABN counterpart. Second, we should shrink the candidate set to reduce the computational burden. This step has been proved very useful in **FVS-ABN**. Third, we use reachability analysis on the BN to filter out the candidate set to obtain the result set that is expected to one-to-one correspond to the set of attractors.

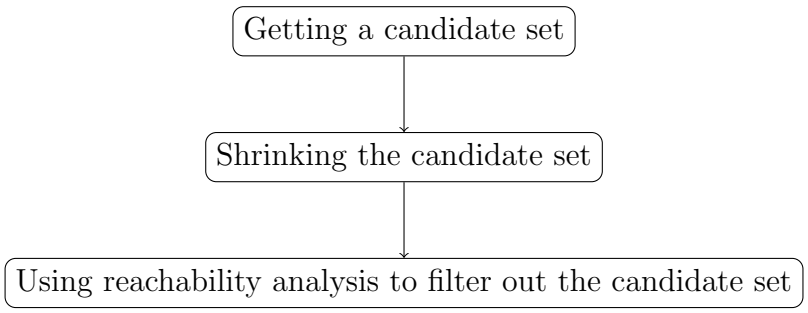


Figure 8.1: Blueprint for attractor detection in various types of BNs.

Finally, although systems biology has served as the main motivation for our research, applications of this dissertation are by no means limited to biological systems. For example, conjunctive BNs [94] are suitable to model water quality networks [188], in which each Boolean variable can be viewed as the water quality within a pipe. The Boolean variable takes value 1 if the water is not polluted, whereas value 0 if the water is polluted. As another example, ABNs are useful in modeling, studying, and controlling nonlinear dynamics in multivariate systems [35]. In addition, BNs provide a convenient modeling framework to explore general properties of complex systems in general, such as, self-organization, criticality, causality, canalization, robustness, and evolvability [36]. Other systems that can be modeled by BNs include multi-agent systems (modeled by SBNs) [189], social networks (e.g., information flow on Twitter or Facebook) [190], smart grids (modeled by SPBNs) [191], and supply chain networks (e.g., movement of materials) [192]. Since we consider general BNs (i.e., there is no restriction in Boolean functions) as well as different types of BNs (GABNs, ABNs, DGABNs, DA-PBNs), the results (theoretical results and computational methods) introduced in this dissertation can be applied to a wide range of other systems.

8.2 Future Work

There are a number of research directions on BNs that could be pursued in the future, and we mention here a few of them.

First, the theoretical results obtained in this dissertation could be further explored to contribute more insights into the dynamics of BNs as well as pave potential ways for developing more efficient methods for attractor detection and optimal control of BNs. For example, although we have explored the relations in dynamics between GABNs and SBNs (or ABNs) in Chapter 3, there is no previous work linking the cyclic attractors of an SBN and those of its ABN counterpart to our best knowledge. Hence, exploring the connection between SBNs and ABNs is theoretically interesting and one of our future work. In addition, we have shown a counter example (Example 3.6.1) in which the number

of attractors of an ABN is greater than that of its GABN counterpart. However, the experimental results on real biological networks show that the numbers of ABN and GABN attractors are identical in most cases. A possible future direction would be to investigate what condition in which the number of attractors of an ABN is identical to that of its GABN counterpart. It is potentially possible because there are some preliminary results [11, 78, 148] following this direction. In Theorem 4.6.2 of Chapter 4, we have introduced the relation between the dynamics of an ABN and an NFVS of its interaction graph. One natural question is that whether this theorem still holds for other types of BNs, such as, SBNs, GABNs, and ROABNs. This question is interesting because the obtained relations in the case of FVSs do not depend on updating schemes and is important because the size of a minimum NFVS is less than or equal to the size of a minimum FVS. In Chapter 7, we have provided several complexity analysis on three optimal control problems of DA-PBNs. However, the precise complexity of these problems is so far open to our best knowledge. Moreover, the precise complexity of attractor detection and optimal control problems of other types of BNs (e.g., ABNs, GABNs, DGABNs) is also still open so far. Hence, one more possible direction is to study the computational complexity of problems on BNs.

Second, the methods proposed in this dissertation could be further improved to handle larger networks (targeting genome-scale networks that can possess thousands of components [49, 54]). Although these proposed methods outperform the state-of-the-art corresponding methods in the literature, their applicable ranges are so far from genome-scale networks. Hence, it is important but challenging to improve the proposed methods. One possible direction is to improve substituent steps in each method. For instance, as in Chapter 3, most of the running time of **filtSAT** or **ApproABN** is spent for computing the attractors of the SBN counterpart of the GABN. Then, we can improve **filtSAT** or **ApproABN** by developing a more efficient method or efficiently combining multiple previous methods for computing SBN attractors. In addition, excluding redundant SBN attractors before the filtering process of **filtSAT** may be a potential improvement. For the case of the **iFVS-ABN** method presented in Chapter 4, we may improve this method by considering several other techniques for checking the reachability in ABNs [141, 144, 145, 146] or an exact method for finding a minimum NFVS. One more possible direction is to combine the proposed methods with previously efficient approximation methods. For instance, we may combine **iFVS-ABN** with the efficient approximation methods for computing attractors of ABNs [53, 54]. It is promising because these approximation methods can handle very large networks as reported by the authors. Furthermore, combining the proposed methods with the decomposition-based [49, 181] or reduction-based [52, 67, 129] approaches (especially in the case of DA-PBNs) may be of interest.

Third, both theoretical and practical results of this dissertation could be extended to those for other less popular but more complex types of BNs such as ROABNs or generalized to those for more general models such as multi-valued networks and hybrid models. ROABNs are a non-standard type of asynchronous BNs. Initially, this type was used in various biological research work [79, 158, 193]. However, ROABNs have gradually been less popular due to their high complexity. Hence, research on the dynamics of ROABNs as well as the development of efficient methods for attractor detection and optimal control of ROABNs are interesting and challenging. Since the obtained relations in the case of FVSs presented in Chapter 4 still hold for the case of ROABNs, it is potential to extend the theoretical and practical results of Chapter 4 to those for ROABNs. In some

cases, BNs are not expressive enough and we need to use their generalization, multi-valued networks [167] or hybrid models [186]. In a multi-valued network, each node can receive more than two expression levels. In a hybrid model, each node can receive a discrete or a continuous value. Many methods have been proposed for converting a multi-valued network into a BN whose dynamics is equivalent to that of the multi-valued network. Then, we can apply directly the proposed methods to the encoded BN. However, the size of the encoded BN may be unmanageable by the proposed methods. Hence, we need more direct and efficient methods for multi-valued networks. It is also similar in the case of hybrid models.

Last, but not least, research on the dynamics of several special classes of BNs such as canalyzing and nested canalyzing BNs [90, 91], AND-OR BNs [92, 93], conjunctive BNs [94], as well as their attractor detection and optimal control problems, may be of interest. Since we in this dissertation consider general Boolean networks, the obtained theoretical results and the developed methods can directly be applied to these special classes of BNs. However, because of their special structures, deeper theoretical results and more efficient methods may be obtained. This intuition is justified by various efficient work on attractor detection of special classes of SBNs [21].

Bibliography

[1] Stuart A Kauffman. *The origins of order: Self-organization and selection in evolution*. OUP USA, 1993.

[2] Guy Karlebach and Ron Shamir. Modelling and analysis of gene regulatory networks. *Nature Reviews Molecular Cell Biology*, 9(10):770–780, 2008.

[3] Shubhank Sherekar and Ganesh A Viswanathan. Boolean dynamic modeling of cancer signaling networks: Prognosis, progression, and therapeutics. *Computational and Systems Oncology*, 1(2):e1017, 2021.

[4] 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.

[5] Carlos Gershenson. Introduction to random Boolean networks. In *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems (ALife IX)*, page 160–173. MIT Press, 2004.

[6] Julian D Schwab, Silke D Kühlwein, Nensi Ikonomi, Michael Kühl, and Hans A Kestler. Concepts in Boolean network modeling: What do they all mean? *Computational and Structural Biotechnology Journal*, 18:571–582, 2020.

[7] Jose C Valverde, Henning S Mortveit, Carlos Gershenson, and Yongtang Shi. Boolean networks and their applications in science and engineering. *Complexity*, 2020:6183798:1–6183798:3, 2020.

[8] Stuart A Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437–467, 1969.

[9] Assieh Saadatpour, István Albert, and Réka Albert. Attractor analysis of asynchronous Boolean models of signal transduction networks. *Journal of Theoretical Biology*, 266(4):641–656, 2010.

[10] Inman Harvey and Terry Bossomaier. Time out of joint: Attractors in asynchronous random Boolean networks. In *Proceedings of the Fourth European Conference on Artificial Life*, pages 67–75. MIT Press, Cambridge, 1997.

[11] Mathilde Noual. General iteration graphs and Boolean automata circuits. *arXiv preprint arXiv:1104.4044*, 2011.

[12] Thomas Chatain, Stefan Haar, and Loïc Paulevé. Boolean networks: beyond generalized asynchronicity. In *International Workshop on Cellular Automata and Discrete Complex Systems*, pages 29–42. Springer, 2018.

[13] Carlos Gershenson, Jan Broekaert, and Diederik Aerts. Contextual random Boolean networks. In *European Conference on Artificial Life*, pages 615–624. Springer, 2003.

[14] Carlos Gershenson. Classification of random Boolean networks. In *Proceedings of the Eighth International Conference on Artificial Life*, pages 1–8. MIT Press, 2002.

[15] Babak Faryabi, Jean-Francois Chamberland, Golnaz Vahedi, Aniruddha Datta, and Edward R Dougherty. Optimal intervention in semi-Markov-based asynchronous genetic regulatory networks. In *American Control Conference*, pages 1388–1393. IEEE, 2008.

[16] Ilya Shmulevich and Edward R Dougherty. *Probabilistic Boolean networks: the modeling and control of gene regulatory networks*. SIAM, 2010.

[17] Panuwat Trairatphisan, Andrzej Mizera, Jun Pang, Alexandru Adrian Tantar, Jochen Schneider, and Thomas Sauter. Recent development and biomedical applications of probabilistic Boolean networks. *Cell Communication and Signaling*, 11(1):1–25, 2013.

[18] Koichi Kobayashi and Kunihiko Hiraishi. Optimization-based approaches to control of probabilistic Boolean networks. *Algorithms*, 10(1):31, 2017.

[19] Peter Bloomingdale, Van Anh Nguyen, Jin Niu, and Donald E Mager. Boolean network modeling in systems pharmacology. *Journal of Pharmacokinetics and Pharmacodynamics*, 45(1):159–180, 2018.

[20] Pedro J Rivera Torres, EI Serrano Mercado, and Luis Anido Rifón. Probabilistic Boolean network modeling of an industrial machine. *Journal of Intelligent Manufacturing*, 29(4):875–890, 2018.

[21] Tatsuya Akutsu. *Algorithms for analysis, inference, and control of Boolean networks*. World Scientific, 2018.

[22] Stuart A Kauffman. The origins of order: Self-organization and selection in evolution. In *Spin Glasses and Biology*, pages 61–100. World Scientific, 1992.

[23] Sui Huang. Genomics, complexity and drug discovery: insights from Boolean network models of cellular regulation. *Pharmacogenomics*, 2(3):203–222, 2001.

[24] Reka Albert and Juilee Thakar. Boolean modeling: a logic-based dynamic approach for understanding signaling and regulatory networks and for making useful predictions. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 6(5):353–369, 2014.

[25] Shinya Yamanaka. Elite and stochastic models for induced pluripotent stem cell generation. *Nature*, 460(7251):49, 2009.

[26] Minsoo Choi, Jue Shi, Yanting Zhu, Ruizhen Yang, and Kwang-Hyun Cho. Network dynamics-based cancer panel stratification for systemic prediction of anticancer drug response. *Nature Communications*, 8(1):1–12, 2017.

[27] Jonas Béal, Lorenzo Pantolini, Vincent Noël, Emmanuel Barillot, and Laurence Calzone. Personalized logical models to investigate cancer response to BRAF treatments in melanomas and colorectal cancers. *PLoS Computational Biology*, 17(1):e1007900, 2021.

[28] Vincent Noël, Jose Carbonell, Miguel Ponce de Leon, Sylvain Soliman, Anna Niarakis, Laurence Calzone, Emmanuel Barillot, Alfonso Valencia, and Arnau Montagud. PhysiBoSS-COVID: the Boolean modelling of COVID-19 signalling pathways in a multicellular simulation framework allows for the uncovering of mechanistic insights, November 2020. European Commission grants: INFORE project(H2020-ICT-825070) and PerMedCoE project(H2020-ICT-951773).

[29] Manuel Azaid Ordaz Arias, Mariana Martinez Sanchez, and Yalbi Balderas Martinez. A Boolean network for studying the macrophages differentiation in SARS-CoV-2 infection, 2021.

[30] Mahmoud AA Ibrahim, Alaa HM Abdelrahman, Tarik A Mohamed, Mohamed AM Atia, Montaser AM Al-Hammady, Khlood AA Abdeljawaad, Eman M Elkady, Mahmoud F Moustafa, Faris Alrumaihi, Khaled S Allemailem, et al. In silico mining of terpenes from red-sea invertebrates for SARS-CoV-2 main protease (Mpro) inhibitors. *Molecules*, 26(7):2082, 2021.

[31] Oyebode J. Oyeyemi, Oluwafemi Davies, David L. Robertson, and Jean-Marc Schwartz. A logical model of HIV-1 interactions with the T-cell activation signalling pathway. *Bioinformatics*, 31(7):1075–1083, November 2014.

[32] Kyaw Tun, Marta Menghini, Lina D'Andrea, Pawan Dhar, Hiroshi Tanaka, and Alessandro Giuliani. Why so few drug targets: a mathematical explanation? *Current Computer-Aided Drug Design*, 7(3):206–213, 2011.

[33] Itziar Irurzun-Arana, José Martín Pastor, Iñaki F Trocóniz, and José David Gómez-Mantilla. Advanced Boolean modeling of biological networks applied to systems pharmacology. *Bioinformatics*, 33(7):1040–1048, 2017.

[34] Matthew Putnins and Ioannis P Androulakis. Boolean modeling in quantitative systems pharmacology: Challenges and opportunities. *Critical Reviews™ in Biomedical Engineering*, 47(6), 2019.

[35] Xiao Yang, Nilam Ram, Peter CM Molenaar, and Pamela M Cole. Describing and controlling multivariate nonlinear dynamics: A Boolean network approach. *Multivariate Behavioral Research*, pages 1–30, 2021.

[36] Alexander J. Gates, Rion Brattig Correia, Xuan Wang, and Luis M. Rocha. The effective graph reveals redundancy, canalization, and control pathways in biochemical regulation and signaling. *Proceedings of the National Academy of Sciences*, 118(12):e2022598118, March 2021.

[37] Daizhan Cheng and Hongsheng Qi. Controllability and observability of Boolean control networks. *Automatica*, 45(7):1659–1667, 2009.

[38] Hiroaki Kitano. Cancer as a robust system: implications for anticancer therapy. *Nature Reviews Cancer*, 4(3):227–235, 2004.

[39] Célia Biane and Franck Delaplace. Causal reasoning on Boolean control networks based on abduction: theory and application to cancer drug discovery. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16(5):1574–1585, 2018.

[40] Nadia S Taou, David W Corne, and Michael A Lones. Investigating the use of Boolean networks for the control of gene regulatory networks. *Journal of Computational Ccience*, 26:147–156, 2018.

[41] Rihab Gam, Minkyung Sung, and Arun Prasad Pandurangan. Experimental and computational approaches to direct cell reprogramming: Recent advancement and future challenges. *Cells*, 8(10):1189, 2019.

[42] Ettore Fornasini and Maria Elena Valcher. Fault detection analysis of Boolean control networks. *IEEE Transactions on Automatic Control*, 60(10):2734–2739, 2015.

[43] Abhishek Garg, Alessandro Di Cara, Ioannis Xenarios, Luis Mendoza, and Giovanni De Micheli. Synchronous versus asynchronous modeling of gene regulatory networks. *Bioinformatics*, 24(17):1917–1925, 2008.

[44] Elena Dubrova and Maxim Teslenko. A SAT-based algorithm for finding attractors in synchronous Boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(5):1393–1399, 2011.

[45] Thomas Skodawessely and Konstantin Klemm. Finding attractors in asynchronous Boolean dynamics. *Advances in Complex Systems*, 14(03):439–449, 2011.

[46] Wensheng Guo, Guowu Yang, Wei Wu, Lei He, and Mingyu Sun. A parallel attractor finding algorithm based on Boolean satisfiability for genetic regulatory networks. *PloS One*, 9(4):e94258, 2014.

[47] Qixia Yuan, Hongyang Qu, Jun Pang, and Andrzej Mizera. Improving BDD-based attractor detection for synchronous Boolean networks. *Science China Information Sciences*, 59(8):080101, 2016.

[48] Qinbin He, Zhile Xia, and Bin Lin. P_UNSAT approach of attractor calculation for Boolean gene regulatory networks. *Journal of Theoretical Biology*, 447:171–177, 2018.

[49] Andrzej Mizera, Jun Pang, Hongyang Qu, and Qixia Yuan. Taming asynchrony for attractor detection in large Boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16(1):31–42, 2018.

[50] Qixia Yuan, Andrzej Mizera, Jun Pang, and Hongyang Qu. A new decomposition-based method for detecting attractors in synchronous Boolean networks. *Science of Computer Programming*, 180:18–35, 2019.

[51] René Thomas. Regulatory networks seen as asynchronous automata: a logical description. *Journal of Theoretical Biology*, 153:1–23, 1991.

[52] Jorge GT Zañudo and Réka Albert. An effective network reduction approach to find the dynamical repertoire of discrete dynamic networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 23(2):025111, 2013.

[53] Hannes Klarner, Adam Streck, and Heike Siebert. PyBoolNet: a python package for the generation, analysis and visualization of Boolean networks. *Bioinformatics*, 33(5):770–772, 2017.

[54] Jordan C Rozum, Jorge Gómez Tejeda Zañudo, Xiao Gan, Dávid Deritei, and Réka Albert. Parity and time reversal elucidate both decision-making in empirical models and attractor scaling in critical Boolean networks. *Science Advances*, 7(29):eabf8124, 2021.

[55] Christopher James Langmead and Sumit Kumar Jha. Symbolic approaches for finding control strategies in Boolean networks. *Journal of Bioinformatics and Computational Biology*, 7(02):323–338, 2009.

[56] 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.

[57] Koichi Kobayashi and Kunihiko Hiraishi. Optimal control of Boolean biological networks modeled by Petri nets. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 96(2):532–539, 2013.

[58] Hugues Mandon, Cui Su, Stefan Haar, Jun Pang, and Loïc Paulevé. Sequential reprogramming of Boolean networks made practical. In *International Conference on Computational Methods in Systems Biology*, pages 3–19. Springer, 2019.

[59] Soumya Paul, Cui Su, Jun Pang, and Andrzej Mizera. An efficient approach towards the source-target control of Boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 17(6):1932–1945, 2019.

[60] Cui Su, Soumya Paul, and Jun Pang. Controlling large Boolean networks with temporary and permanent perturbations. In *International Symposium on Formal Methods*, pages 707–724. Springer, 2019.

[61] Laura Cifuentes Fontanals, Elisa Tonello, and Heike Siebert. Control strategy identification via trap spaces in Boolean networks. In *International Conference on Computational Methods in Systems Biology*, pages 159–175. Springer, 2020.

[62] Cui Su and Jun Pang. Sequential temporary and permanent control of Boolean networks. In *International Conference on Computational Methods in Systems Biology*, pages 234–251. Springer, 2020.

[63] Cui Su and Jun Pang. A dynamics-based approach for the target control of Boolean networks. In *BCB '20: 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics, Virtual Event, USA, September 21-24, 2020*, pages 50:1–50:8. ACM, 2020.

[64] Aniruddha Datta, Ashish Choudhary, Michael L Bittner, and Edward R Dougherty. External control in Markovian genetic regulatory networks. *Machine Learning*, 52(1):169–191, 2003.

[65] Koichi Kobayashi and Kunihiko Hiraishi. An integer programming approach to control problems in probabilistic Boolean networks. In *Proceedings of the 2010 American Control Conference*, pages 6710–6715. IEEE, 2010.

[66] Koichi Kobayashi and Kunihiko Hiraishi. An integer programming approach to optimal control problems in context-sensitive probabilistic Boolean networks. *Automatica*, 47(6):1260–1264, 2011.

[67] Xi Chen, Hao Jiang, Yushan Qiu, and Wai-Ki Ching. On optimal control policy for probabilistic Boolean network: a state reduction approach. *BMC Systems Biology*, 6(1):1–8, 2012.

[68] Koichi Kobayashi and Kunihiko Hiraishi. Optimal control of probabilistic Boolean networks using polynomial optimization. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 95(9):1512–1517, 2012.

[69] Koichi Kobayashi and Kunihiko Hiraishi. Symbolic approach to verification and control of deterministic/probabilistic Boolean networks. *IET Systems Biology*, 6(6):215–222, 2012.

[70] Qiuli Liu. An optimal control approach to probabilistic Boolean networks. *Physica A: Statistical Mechanics and its Applications*, 391(24):6682–6689, 2012.

[71] Xi Chen, Tatsuya Akutsu, Takeyuki Tamura, and Wai-Ki Ching. Finding optimal control policy in probabilistic Boolean networks with hard constraints by using integer programming and dynamic programming. *International Journal of Data Mining and Bioinformatics*, 7(3):1–22, 2013.

[72] Ou Wei, Zonghao Guo, Yun Niu, and Wenyuan Liao. Model checking optimal finite-horizon control for probabilistic gene regulatory networks. *BMC Systems Biology*, 11(6):75–88, 2017.

[73] Qiuli Liu, Yu He, and Junwei Wang. Optimal control for probabilistic Boolean networks using discrete-time Markov decision processes. *Physica A: Statistical Mechanics and its Applications*, 503:1297–1307, 2018.

[74] Mitsuru Toyoda and Yuhu Wu. Mayer-type optimal control of probabilistic Boolean control network with uncertain selection probabilities. *IEEE Transactions on Cybernetics*, 2019.

[75] Florian Greil, Barbara Drossel, and Joost Sattler. Critical kauffman networks under deterministic asynchronous update. *New Journal of Physics*, 9(10):373, 2007.

[76] Yalu Li and Haitao Li. Controllability and stabilization of periodic switched Boolean control networks with application to asynchronous updating. *Nonlinear Analysis: Hybrid Systems*, 41:101054, 2021.

[77] Babak Faryabi, Jean-François Chamberland, Golnaz Vahedi, Aniruddha Datta, and Edward R Dougherty. Optimal intervention in asynchronous genetic regulatory networks. *IEEE Journal of Selected Topics in Signal Processing*, 2(3):412–423, 2008.

[78] Loïc Paulevé and Adrien Richard. Static analysis of Boolean networks based on interaction graphs: A survey. *Electronic Notes in Theoretical Computer Science*, 284:93–104, 2012.

[79] Madalena Chaves, Eduardo D Sontag, and Réka Albert. Methods of robustness analysis for Boolean models of gene control networks. *IEE Proceedings-Systems Biology*, 153(4):154–167, 2006.

[80] István Albert, Juilee Thakar, Song Li, Ranran Zhang, and Reka Albert. Boolean network simulations for life scientists. *Source Code for Biology and Medicine*, 3(1):1–8, 2008.

[81] Julio Aracena, Jacques Demongeot, Eric Fanchon, and Marco Montalva. On the number of different dynamics in Boolean networks with deterministic update schedules. *Mathematical Biosciences*, 242(2):188–194, 2013.

[82] Gonzalo A Ruz, Eric Goles, Marco Montalva, and Gary B Fogel. Dynamical and topological robustness of the mammalian cell cycle network: a reverse engineering approach. *BioSystems*, 115:23–32, 2014.

[83] Trinh Van Giang and Kunihiko Hiraishi. Algorithms for finding attractors of generalized asynchronous random Boolean networks. In *2019 12th Asian Control Conference (ASCC)*, pages 67–72. IEEE, 2019.

[84] Trinh Van Giang and Kunihiko Hiraishi. An efficient method for approximating attractors in large-scale asynchronous Boolean models. In *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1820–1826. IEEE, 2020.

[85] Van Giang TRINH and Kunihiko HIRAISHI. A study on attractors of generalized asynchronous random Boolean networks. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 103(8):987–994, 2020.

[86] G. V. Trinh, T. Akutsu, and K. Hiraishi. An FVS-based approach to attractor detection in asynchronous random Boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2020. in press.

[87] Trinh Van-Giang and Kunihiko Hiraishi. An improved method for finding attractors of large-scale asynchronous Boolean networks. In *2021 IEEE International Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE, 2021.

[88] G. V. Trinh and K. Hiraishi. On attractor detection and optimal control of deterministic generalized asynchronous random Boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2020. in press.

[89] G. V. Trinh and K. Hiraishi. On optimal control of deterministic asynchronous probabilistic Boolean networks. 2021. in preparation.

[90] Stuart Kauffman, Carsten Peterson, Björn Samuelsson, and Carl Troein. Genetic networks with canalyzing Boolean rules are always stable. *Proceedings of the National Academy of Sciences*, 101(49):17102–17107, 2004.

[91] Tatsuya Akutsu, Avraham A Melkman, Takeyuki Tamura, and Masaki Yamamoto. Determining a singleton attractor of a Boolean network with nested canalyzing functions. *Journal of Computational Biology*, 18(10):1275–1290, 2011.

[92] Avraham A Melkman, Takeyuki Tamura, and Tatsuya Akutsu. Determining a singleton attractor of an AND/OR Boolean network in $O(1.587^{\mathrm{n}})$ time. *Information Processing Letters*, 110(14-15):565–569, 2010.

[93] Tatsuya Akutsu, Sven Kosub, Avraham A Melkman, and Takeyuki Tamura. Finding a periodic attractor of a Boolean network. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(5):1410–1421, 2012.

[94] Eyal Weiss, Michael Margaliot, and Guy Even. Minimal controllability of conjunctive Boolean networks is NP-complete. *Automatica*, 92:56–62, 2018.

[95] Julio Aracena, Eric Goles, Andrés Moreira, and Lilian Salinas. On the robustness of update schedules in Boolean networks. *BioSystems*, 97(1):1–8, 2009.

[96] Jason A Papin, Tony Hunter, Bernhard O Palsson, and Shankar Subramaniam. Reconstruction of cellular signalling networks and analysis of their properties. *Nature Reviews Molecular Cell Biology*, 6(2):99–111, 2005.

[97] Calin Guet, Thomas A Henzinger, Claudia Igler, Tatjana Petrov, and Ali Sezgin. Transient memory in gene regulation. In *International Conference on Computational Methods in Systems Biology*, pages 155–187. Springer, 2019.

[98] Abhishek Garg, Ioannis Xenarios, Luis Mendoza, and Giovanni DeMicheli. An efficient method for dynamic analysis of gene regulatory networks and in silico gene perturbation experiments. In *Annual International Conference on Research in Computational Molecular Biology*, pages 62–76. Springer, 2007.

[99] Marco Montalva, Julio Aracena, and Anahí Gajardo. On the complexity of feedback set problems in signed digraphs. *Electronic Notes in Discrete Mathematics*, 30:249–254, 2008.

[100] David S Johnson and Michael R Garey. *Computers and intractability: A guide to the theory of NP-completeness*. WH Freeman, New York, 1979.

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

[102] Javier Esparza and Keijo Heljanko. *Unfoldings: a partial-order approach to model checking*. Springer Science & Business Media, 2008.

[103] Thomas Chatain, Stefan Haar, Loïg Jezequel, Loïc Paulevé, and Stefan Schwoon. Characterization of reachable attractors using Petri net unfoldings. In *International Conference on Computational Methods in Systems Biology*, pages 129–142. Springer, 2014.

[104] Stefan Schwoon and S Romer. Mole—a Petri net unfolder. http://www.lsv.fr/~schwoon/tools/mole/, 2016.

[105] Thomas Chatain, Stefan Haar, Juraj Kolčák, Loïc Paulevé, and Aalok Thakkar. Concurrency in Boolean networks. *Natural Computing*, pages 1–19, 2019.

[106] Javier Esparza and Claus Schröter. Unfolding based algorithms for the reachability problem. *Fundamenta Informaticae*, 47(3-4):231–245, 2001.

[107] Barbara Drossel, Tamara Mihaljev, and Florian Greil. Number and length of attractors in a critical Kauffman model with connectivity one. *Physical Review Letters*, 94(8):088701, 2005.

[108] Desheng Zheng, Guowu Yang, Xiaoyu Li, Zhicai Wang, Feng Liu, and Lei He. An efficient algorithm for computing attractors of synchronous and asynchronous Boolean networks. *PloS One*, 8(4):e60593, 2013.

[109] Randal E Bryant. Graph-based algorithms for Boolean function manipulation. *Computers, IEEE Transactions on*, 100(8):677–691, 1986.

[110] Armin Biere, Alessandro Cimatti, Edmund M Clarke, Masahiro Fujita, and Yunshan Zhu. Symbolic model checking using SAT procedures instead of BDDs. In *Proceedings 1999 Design Automation Conference (Cat. No. 99CH36361)*, pages 317–320. IEEE, 1999.

[111] Aiguo Xie and Peter A Beerel. Efficient state classification of finite-state Markov chains. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(12):1334–1339, 1998.

[112] Arash Vahidi. JDD: a pure Java BDD and Z-BDD library. https://bitbucket.org/vahidi/jdd, 2015.

[113] Fangting Li, Tao Long, Ying Lu, Qi Ouyang, and Chao Tang. The yeast cell-cycle network is robustly designed. *Proceedings of the National Academy of Sciences*, 101(14):4781–4786, 2004.

[114] Adrien Fauré, Aurélien Naldi, Claudine Chaouiya, and Denis Thieffry. Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. *Bioinformatics*, 22(14):e124–e131, 2006.

[115] Luis Mendoza and Ioannis Xenarios. A method for the generation of standardized qualitative dynamical systems of regulatory networks. *Theoretical Biology and Medical Modelling*, 3(1):13, 2006.

[116] Steffen Klamt, Julio Saez-Rodriguez, Jonathan A Lindquist, Luca Simeoni, and Ernst D Gilles. A methodology for the structural and functional analysis of signaling and regulatory networks. *BMC Bioinformatics*, 7(1):56, 2006.

[117] Christoph Müssel, Martin Hopfensitz, and Hans A Kestler. BoolNet—an R package for generation, reconstruction and analysis of Boolean networks. *Bioinformatics*, 26(10):1378–1380, 2010.

[118] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[119] Tomáš Helikar, Bryan Kowal, Sean McClenathan, Mitchell Bruckner, Thaine Rowley, Alex Madrahimov, Ben Wicks, Manish Shrestha, Kahani Limbu, and Jim A Rogers. The Cell Collective: toward an open and collaborative approach to systems biology. *BMC Systems Biology*, 6(1):96, 2012.

[120] Maximino Aldana. Boolean dynamics of networks with scale-free topology. *Physica D: Nonlinear Phenomena*, 185(1):45–66, 2003.

[121] Tatsuya Akutsu, Satoru Kuhara, Osamu Maruyama, and Satoru Miyano. A system for identifying genetic networks from gene expression patterns produced by gene disruptions and overexpressions. *Genome Informatics*, 9:151–160, 1998.

[122] Aurélien Naldi, Pedro T Monteiro, and Claudine Chaouiya. Efficient handling of large signalling-regulatory networks by focusing on their core control. In *International Conference on Computational Methods in Systems Biology*, pages 288–306. Springer, 2012.

[123] Pritha Dutta, Lichun Ma, Yusuf Ali, Peter MA Sloot, and Jie Zheng. Boolean network modeling of β-cell apoptosis and insulin resistance in type 2 diabetes mellitus. *BMC Systems Biology*, 13(2):36, 2019.

[124] Meike Dahlhaus, Andre Burkovski, Falk Hertwig, Christoph Mussel, Ruth Volland, Matthias Fischer, Klaus-Michael Debatin, Hans A Kestler, and Christian Beltinger. Boolean modeling identifies Greatwall/MASTL as an important regulator in the AURKA network of neuroblastoma. *Cancer Letters*, 371(1):79–89, 2016.

[125] Martín Rodríguez Vega. Analyzing toys models of Arabidopsis and Drosphila using Z3 SMT-LIB. In *Independent Component Analyses, Compressive Sampling, Wavelets, Neural Net, Biosystems, and Nanoengineering XII*, volume 9118, page 911813. International Society for Optics and Photonics, 2014.

[126] Elisabeth Remy, Sandra Rebouissou, Claudine Chaouiya, Andrei Zinovyev, François Radvanyi, and Laurence Calzone. A modeling approach to explain mutually exclusive and co-occurring genetic alterations in bladder tumorigenesis. *Cancer Research*, 75(19):4042–4052, 2015.

[127] Shu-Qin Zhang, Morihiro Hayashida, Tatsuya Akutsu, Wai-Ki Ching, and Michael K Ng. Algorithms for finding small attractors in Boolean networks. *EURASIP Journal on Bioinformatics and Systems Biology*, 2007:1–13, 2007.

[128] Tatsuya Akutsu, Morihiro Hayashida, and Takeyuki Tamura. Integer programming-based methods for attractor detection and control of Boolean networks. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 5610–5617. IEEE, 2009.

[129] Alan Veliz-Cuba, Boris Aguilar, Franziska Hinkelmann, and Reinhard Laubenbacher. Steady state analysis of Boolean molecular network models via model reduction and computational algebra. *BMC Bioinformatics*, 15(1):1–8, 2014.

[130] Julio Aracena, Luis Cabrera-Crot, and Lilian Salinas. Finding the fixed points of a Boolean network from a positive feedback vertex set. *Bioinformatics*, 37(8):1148–1155, 2021.

[131] Roland Somogyi and Larry D Greller. The dynamics of molecular networks: applications to therapeutic discovery. *Drug Discovery Today*, 6(24):1267–1277, 2001.

[132] Qinbin He, Zhile Xia, and Bin Lin. An efficient approach of attractor calculation for large-scale Boolean gene regulatory networks. *Journal of Theoretical Biology*, 408:137–144, 2016.

[133] Yin Zhao, Jongrae Kim, and Maurizio Filippone. Aggregation algorithm towards large-scale Boolean network analysis. *IEEE Transactions on Automatic Control*, 58(8):1976–1985, 2013.

[134] Nikola Beneš, Luboš Brim, Jakub Kadlecaj, Samuel Pastva, and David Šafránek. AEON: Attractor bifurcation analysis of parametrised Boolean networks. In *Computer Aided Verification*, pages 569–581. Springer International Publishing, 2020.

[135] Nikola Beneš, Luboš Brim, Samuel Pastva, and David Šafránek. Computing bottom SCCs symbolically using transition guided reduction. In *Computer Aided Verification*, pages 505–528. Springer International Publishing, 2021.

[136] Cui Su, Jun Pang, and Soumya Paul. Towards optimal decomposition of Boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2019. in press.

[137] 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.

[138] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[139] Tomoyuki Yamakami. Quantum optimization problems, 2002.

[140] Loïc Paulevé. Pint: A static analyzer for transient dynamics of qualitative networks with IPython interface. In *International Conference on Computational Methods in Systems Biology*, pages 309–316. Springer, 2017.

[141] Xinwei Chai, Tony Ribeiro, Morgan Magnin, Olivier Roux, and Katsumi Inoue. Static analysis and stochastic search for reachability problem. *Electronic Notes in Theoretical Computer Science*, 350:139–158, 2020.

[142] Adrien Richard. Negative circuits and sustained oscillations in asynchronous automata networks. *Advances in Applied Mathematics*, 44(4):378–392, 2010.

[143] Adrien Richard. Positive and negative cycles in Boolean networks. *Journal of Theoretical Biology*, 463:67–76, 2019.

[144] Paolo Baldan, Alessandro Bruni, Andrea Corradini, Barbara König, César Rodríguez, and Stefan Schwoon. Efficient unfolding of contextual Petri nets. *Theoretical Computer Science*, 449:2–22, 2012.

[145] César Rodríguez, Stefan Schwoon, and Victor Khomenko. Contextual merged processes. In *International Conference on Applications and Theory of Petri Nets and Concurrency*, pages 29–48. Springer, 2013.

[146] Giovanni Casu and G Michele Pinna. Flow unfolding of multi-clock nets. In *International Conference on Applications and Theory of Petri Nets and Concurrency*, pages 170–189. Springer, 2014.

[147] Björn Samuelsson and Carl Troein. Superpolynomial growth in the number of attractors in Kauffman networks. *Physical Review Letters*, 90(9):098701, 2003.

[148] Amer Shreim, Andrew Berdahl, Florian Greil, Jörn Davidsen, and Maya Paczuski. Attractor and basin entropies of random Boolean networks under asynchronous stochastic update. *Physical Review E*, 82(3):035102, 2010.

[149] Meng Yang and Tianguang Chu. Evaluation of attractors and basins of asynchronous random Boolean networks. *Physical Review E*, 85(5):056105, 2012.

[150] Masado Ishii, Jacob Gores, and Christof Teuscher. On the sparse percolation of damage in finite non-synchronous random Boolean networks. *Physica D: Nonlinear Phenomena*, 398:84–91, 2019.

[151] Aravind Karanam, David He, Po-Kai Hsu, Sebastian Schulze, Guillaume Dubeaux, Richa Karmakar, Julian I Schroeder, and Wouter-Jan Rappel. BoolSim, a graphical interface for open access Boolean network simulations and use in guard cell CO2 signaling. *bioRxiv*, 2021.

[152] François Robert. *Discrete iterations: a metric study*, volume 6. Springer Science & Business Media, 2012.

[153] Eric Goles, Marco Montalva, and Gonzalo A Ruz. Deconstruction and dynamical robustness of regulatory networks: application to the yeast cell cycle networks. *Bulletin of Mathematical Biology*, 75(6):939–966, 2013.

[154] Carlos Grilo and Luís Correia. The influence of asynchronous dynamics in the spatial prisoner's dilemma game. In *International Conference on Simulation of Adaptive Behavior*, pages 362–371. Springer, 2008.

[155] Martin Schneiter, Jaroslav Rička, and Martin Frenz. Self-organization of self-clearing beating patterns in an array of locally interacting ciliated cells formulated as an adaptive Boolean network. *Theory in Biosciences*, 139(1):21–45, 2020.

[156] Pramuditha Waidyarathne and Sandhya Samarasinghe. Boolean calcium signalling model predicts calcium role in acceleration and stability of abscisic acid-mediated stomatal closure. *Scientific Reports*, 8(1):1–16, 2018.

[157] Zhiqiang Li, Huimin Xiao, and Jinli Song. Algebraic approach to asynchronous Boolean networks. In *2011 Chinese Control and Decision Conference (CCDC)*, pages 769–773. IEEE, 2011.

[158] Rui-Sheng Wang, Assieh Saadatpour, and Reka Albert. Boolean modeling in systems biology: an overview of methodology and applications. *Physical Biology*, 9(5):055001, 2012.

[159] Clark Barrett. "Decision procedures: An algorithmic point of view," by Daniel Kroening and Ofer Strichman, Springer-Verlag, 2008. *Journal of Automated Reasoning*, 51(4):453–456, 2013.

[160] Jose C Nacher and Tatsuya Akutsu. Minimum dominating set-based methods for analyzing biological networks. *Methods*, 102:57–63, 2016.

[161] 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.

[162] Laurent Tournier and Madalena Chaves. Uncovering operational interactions in genetic networks using asynchronous Boolean dynamics. *Journal of Theoretical Biology*, 260(2):196–209, 2009.

[163] Suzanne Lenhart and John T Workman. *Optimal control applied to biological models*. CRC press, 2007.

[164] Qinqin Chai and Wu Wang. A computational method for free terminal time optimal control problem governed by nonlinear time delayed systems. *Applied Mathematical Modelling*, 53:242–250, 2018.

[165] Babak Faryabi, Jean-François Chamberland, Golnaz Vahedi, Aniruddha Datta, and Edward R Dougherty. Optimal intervention in asynchronous genetic regulatory networks. *IEEE Journal of Selected Topics in Signal Processing*, 2(3):412–423, 2008.

[166] Nikolaj Bjørner, Anh-Dung Phan, and Lars Fleckenstein. νZ - an optimizing SMT solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 194–199. Springer, 2015.

[167] Chao Luo and Xingyuan Wang. Algebraic representation of asynchronous multiple-valued networks and its dynamics. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(4):927–938, 2013.

[168] Yin Zhao and DaiZhan Cheng. On controllability and stabilizability of probabilistic Boolean control networks. *Science China Information Sciences*, 57(1):1–14, 2014.

[169] Mitsuru Toyoda and Yuhu Wu. On optimal time-varying feedback controllability for probabilistic Boolean control networks. *IEEE Transactions on Neural Networks and Learning Systems*, 31(6):2202–2208, 2019.

[170] Dai-zhan Cheng and Li-jun Zhang. On semi-tensor product of matrices and its applications. *Acta Mathematicae Applicatae Sinica*, 19(2):219–228, 2003.

[171] Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *International Conference on Computer Aided Verification*, pages 585–591. Springer, 2011.

[172] Tino Teige. *Stochastic satisfiability modulo theories: a symbolic technique for the analysis of probabilistic hybrid systems*. PhD thesis, Universität Oldenburg, 2012.

[173] Hayato Waki, Sunyoung Kim, Masakazu Kojima, Masakazu Muramatsu, and Hiroshi Sugimoto. Algorithm 883: SparsePOP—a sparse semidefinite programming relaxation of polynomial optimization problems. *ACM Transactions on Mathematical Software (TOMS)*, 35(2):1–13, 2008.

[174] Michael L Littman, Judy Goldsmith, and Martin Mundhenk. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research*, 9:1–36, 1998.

[175] H Paul Williams. *Model building in mathematical programming*. John Wiley & Sons, 2013.

[176] Vojtěch Forejt, Marta Kwiatkowska, Gethin Norman, and David Parker. Automated verification techniques for probabilistic systems. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems*, pages 53–113. Springer, 2011.

[177] Tino Teige. Quick start guide and tutorial. 2012.

[178] Pei-Wei Chen, Yu-Ching Huang, and Jie-Hong R Jiang. A sharp leap from quantified Boolean formula to stochastic Boolean satisfiability solving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3697–3706, 2021.

[179] Ashani T Weeraratna, Yuan Jiang, Galen Hostetter, Kevin Rosenblatt, Paul Duray, Michael Bittner, and Jeffrey M Trent. WNT5A signaling directly affects cell motility and invasion of metastatic melanoma. *Cancer Cell*, 1(3):279–288, 2002.

[180] Marta Kwiatkowska, Gethin Norman, and David Parker. Stochastic model checking. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems*, pages 220–270. Springer, 2007.

[181] Marta Kwiatkowska, Gethin Norman, David Parker, and Hongyang Qu. Compositional probabilistic verification through multi-objective model checking. *Information and Computation*, 232:38–65, 2013.

[182] Marta Kwiatkowska, Gethin Norman, and David Parker. Probabilistic model checking: Advances and applications. In *Formal System Verification*, pages 73–121. Springer, 2018.

[183] Ahmed Mahdi and Martin Fränzle. Generalized Craig interpolation for stochastic satisfiability modulo theory problems. In *International Workshop on Reachability Problems*, pages 203–215. Springer, 2014.

[184] Daniel Bienstock and Gonzalo Munoz. LP formulations for polynomial optimization problems. *SIAM Journal on Optimization*, 28(2):1121–1150, 2018.

[185] Jie Wang, Victor Magron, and Jean-Bernard Lasserre. Chordal-TSSOS: a moment-SOS hierarchy that exploits term sparsity with chordal extension. *SIAM Journal on Optimization*, 31(1):114–141, 2021.

[186] Calin Belta, Jonathan Schug, Thao Dang, Vijay Kumar, George J Pappas, Harvey Rubin, and Paul Dunlap. Stability and reachability analysis of a hybrid model of luminescence in the marine bacterium Vibrio fischeri. In *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228)*, volume 1, pages 869–874. IEEE, 2001.

[187] Andrzej Mizera, Jun Pang, Cui Su, and Qixia Yuan. ASSA-PBN: A toolbox for probabilistic Boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 15(4):1203–1216, 2017.

[188] Zuguang Gao, Xudong Chen, and Tamer Başar. Controllability of conjunctive Boolean networks with application to gene regulation. *IEEE Transactions on Control of Network Systems*, 5(2):770–781, 2017.

[189] Stepan Kochemazov and Alexander Semenov. Using synchronous Boolean networks to model several phenomena of collective behavior. *PLoS ONE*, 9(12):e115156, December 2014.

[190] 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.

[191] Pedro J Rivera-Torres and Orestes Llanes Santiago. Fault detection and isolation in smart grid devices using probabilistic Boolean networks. In *Computational Intelligence in Emerging Technologies for Engineering Applications*, pages 165–185. Springer, 2020.

[192] Geoff Easton, Roger J Brooks, Kristina Georgieva, and Ian Wilkinson. Understanding the dynamics of industrial networks using Kauffman Boolean networks. *Advances in Complex Systems*, 11(01):139–164, 2008.

[193] Juilee Thakar, Ashutosh K Pathak, Lisa Murphy, Réka Albert, and Isabella M Cattadori. Network model of immune responses reveals key effectors to single and co-infection dynamics by a respiratory bacterium and a gastrointestinal helminth. *PLoS Computational Biology*, 8(1):e1002345, 2012.

Publications and Awards

Journals

[1] Trinh Van Giang, Tatsuya Akutsu and Kunihiko Hiraishi: "On dynamics of random order asynchronous Boolean networks and an efficient FVS-based method for approximating their attractors," 2021, in preparation (not included in this dissertation).

[2] Trinh Van Giang and Kunihiko Hiraishi: "Computing attractors of large-scale asynchronous Boolean networks using minimal trap spaces," 2021, in preparation (not included in this dissertation).

[3] Trinh Van Giang and Kunihiko Hiraishi: "On optimal control of deterministic asynchronous probabilistic Boolean networks," 2021, in preparation.

[4] Trinh Van Giang and Kunihiko Hiraishi: "On attractor detection and optimal control of deterministic generalized asynchronous random Boolean networks," IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2020, in press. https://doi.org/10.1109/TCBB.2020.3043785.

[5] Trinh Van Giang, Tatsuya Akutsu and Kunihiko Hiraishi: "An FVS-based approach to attractor detection in asynchronous random Boolean networks," IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2020, in press. https://doi.org/10.1109/TCBB.2020.3028862.

[6] Trinh Van Giang and Kunihiko Hiraishi: "A study on attractors of generalized asynchronous random Boolean networks," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 103(8), 987-994, 2020. https://doi.org/10.1587/transfun.2019EAP1163.

Conference papers

[7] Trinh Van Giang and Kunihiko Hiraishi: "An improved method for finding attractors of large-scale asynchronous Boolean networks", accepted to 18th IEEE International Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB 2021).

[8] Trinh Van Giang and Kunihiko Hiraishi: "An efficient method for approximating attractors in large-scale asynchronous Boolean models," 13th International Workshop

on Biological Network Analysis and Integrative Graph-Based Approaches (IWBNA 2020), in Proc. IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2020), 1820-1826, 2020. https://doi.org/10.1109/BIBM49941.2020.9313230.

[9] Trinh Van Giang and Kunihiko Hiraishi: "Algorithms for finding attractors of generalized asynchronous random Boolean networks", in Proc. 12th Asian Control Conference (ASCC 2019), 67-72, 2019. http://ieeexplore.ieee.org/document/8765169.

Awards

- JAIST President Award, September 2019.