UNIVERSITÉ DU
LUXEMBOURG

PhD-FSTM-2020-58
The Faculty of Science, Technology and Medicine

DISSERTATION

Defence held on 03/11/2020 in Esch-sur-Alzette

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Cui SU
Born on 29 January 1991 in Hebei (China)

SCALABLE CONTROL OF ASYNCHRONOUS BOOLEAN NETWORKS

Dissertation defence committee

Dr. Jun Pang, dissertation supervisor
Senior Researcher, Université du Luxembourg

Dr. Sjouke Mauw, Chairman
Professor, Université du Luxembourg

Dr. Thomas Sauter, Vice Chairman
Professor, Université du Luxembourg

Dr. Jaco van de Pol
Professor, Aarhus University

Dr. Loïc Paulevé
Université de Bordeaux

"As was predicted at the beginning of the Human Genome Project, getting the sequence will be the easy part as only technical issues are involved. The hard part will be finding out what it means, because this poses intellectual problems of how to understand the participation of the genes in the functions of living cells."

Sydney Brenner

Abstract

Scalable Control of Asynchronous Boolean Networks

by Cui SU

Direct cell reprogramming has been garnering attention for its therapeutic potential for treating the most devastating diseases characterised by defective cells or a deficiency of certain cells. It is capable of reprogramming any kind of abundant cells in the body into the desired cells to restore functions of the diseased organ. It has shown promising benefits for clinical applications, such as cell and tissue engineering, regenerative medicine and drug discovery.

A major obstacle in the application of direct cell reprogramming lies in the identification of effective reprogramming factors. Experimental approaches are usually laborious, time-consuming and enormously expensive. Mathematical modelling of biological systems paves the way to study mechanisms of biological processes and identify therapeutic targets with computational reasoning and tools. Among several modelling frameworks, Boolean networks have apparent advantages. They provide a qualitative description of biological systems and thus evade the parametrisation problem, which often occurs in quantitative models.

In this thesis, we focus on the identification of reprogramming factors based on asynchronous Boolean networks. This problem is equivalent to the control of asynchronous Boolean networks: finding a subset of nodes, whose perturbations can drive the dynamics of the network from the source state (the initial cell type) to the target attractor (the desired cell type). Before diving into the control problems, we first develop a near-optimal decomposition method and use this method to improve the scalability of the decomposition-based method for attractor detection. The new decomposition-based attractor detection method can identify all the exact attractors of the network efficiently, such that we can select the proper attractors corresponding to the initial cell type and the desired cell type as the source and target attractors and predict the key nodes for the conversion. Depending on whether the source state is given or not, we can have two control problems: source-target control and target control. We develop several methods to solve the two problems using different control strategies. All the methods are implemented in our software CABEAN. Given a control problem, CABEAN can provide a rich set of realistic solutions that manipulate the dynamics in different ways, such that biologists can select suitable ones to validate with biological experiments. We believe our works can contribute to a better understanding of the regulatory mechanisms of biological processes and greatly facilitate the development of direct cell reprogramming.

Acknowledgements

Upon completion of my Ph.D. study, I would like to express my gratitude to those who have helped me during the past four years.

I would like to express my sincere appreciation to my supervisor Jun. Throughout these years, he provided me with persistent help and guided me to be professional.

Thanks to Sjouke for his encouragement and advice. Thanks to Andrzej and Paul who helped to cosupervise me at different periods of my study. Thanks to Qixia who worked closely with me in my first year. I have learned a lot from them.

Thanks to Dr. Hugues Mandon, Dr. Loïc Paulevé and Prof. Stefan Haar. We collaborated on two interesting papers about the sequential control of Boolean networks. One of the papers has received the best paper award at CMSB 2019.

Thanks to all the members of SaToSS group. Their kind help and support made my study and life a wonderful time in Luxembourg. Thanks in particular to Zach who helped me to proofread my papers, and to Ninghan and Zhiqiang.

Lastly, I would like to thank my family. Without their understanding and support, it would be impossible for me to complete my study.

Contents

1 **Introduction** **1**
- 1.1 Cell reprogramming 1
- 1.2 Mathematical modelling of biological systems 2
- 1.3 Research questions 5
- 1.4 Related work 7
- 1.5 Contributions 9
- 1.6 Layout 11

2 **Preliminaries** **15**
- 2.1 Boolean networks 15
  - 2.1.1 Boolean networks 15
  - 2.1.2 Dynamics of Boolean networks 16
  - 2.1.3 Attractors and basins 17
- 2.2 Decomposition of Boolean networks 19
  - 2.2.1 SCC decomposition 19
  - 2.2.2 Blocks 19
  - 2.2.3 Projection of states and the cross operation 20
  - 2.2.4 Transition system of the blocks 21
- 2.3 Acyclic partitioning of directed acyclic graphs 22
- 2.4 Real-life biological networks 23

3 **Near-optimal Decomposition of Boolean Networks** **27**
- 3.1 Introduction 27
- 3.2 Problem definition 29
- 3.3 Acyclic partitioning of SCC graphs 30
  - 3.3.1 Initial partitioning 31
  - 3.3.2 Topological refinement 33
  - 3.3.3 Iteration 34
- 3.4 Integration with the SCC decomposition 35
- 3.5 Evaluation 36
- 3.6 Conclusion 39

4 **Computation of the Basin of Attraction** **41**
- 4.1 Introduction 41
- 4.2 A global method 42

4.3 A decomposition-based method 45
4.4 Evaluation 50

5 Source-target Control **53**
5.1 Introduction 53
5.2 The source-target control problems 55
5.2.1 Boolean networks under control 55
5.2.2 The control problems 56
5.3 One-step instantaneous control 59
5.4 One-step temporary control 60
5.5 One-step permanent control 63
5.6 Attractor-based sequential instantaneous control 67
5.7 Attractor-based sequential temporary control 70
5.8 Attractor-based sequential permanent control 70
5.9 Evaluation 72
5.9.1 Control of the cardiac gene regulatory network 73
5.9.2 Control of the myeloid differentiation network 74
5.9.3 Control of the tumour network 79
5.9.4 Control of other biological networks 81
5.10 Conclusion 83

6 Target Control **85**
6.1 Introduction 85
6.2 The target control problems 86
6.3 Instantaneous target control 88
6.4 Temporary target control 89
6.5 Permanent target control 92
6.6 Evaluation 93
6.6.1 Control of the cardiac gene regulatory network 94
6.6.2 Control of the myeloid differentiation network 96
6.6.3 Control of the tumour network 96
6.6.4 Control of other biological networks 97
6.7 Conclusion 100

7 CABEAN: a Software for the Control of Asynchronous Boolean Networks **101**
7.1 Introduction 101
7.2 General features 102
7.3 Case study 104

8 Conclusions and Future Work **111**
8.1 Limitations of existing biological networks 111
8.2 Conclusions 112
8.3 Future work 113

Bibliography **115**

Curriculum Vitae **124**

List of Figures

2.1 A toy example 16
2.2 Different types of attractors 17
2.3 Transition systems of blocks 22

3.1 SCC graph of the PC12 cell network 31
3.2 Partition of the SCC graph of the PC12 cell network 35

4.1 Transition system of Example 6 44

5.1 Two source-target control strategies 53
5.2 The original transition system and the transition system under control of Example 7 56
5.3 Minimal OT and OP control of Example 10 66
5.4 Sequential control of Example 11 72
5.5 Structure of the myeloid differentiation network 75
5.6 Source-target control of the myeloid differentiation network 75
5.7 AST control of the myeloid differentiation network 77
5.8 AST control of the myeloid differentiation network with constraints . 78

6.1 Source-target control and target control 85
6.2 Target control of Example 13 93
6.3 Comparison of TTC and SMC on the number of solutions 99

List of Tables

2.1 An overview of the biological networks 24

3.1 Partition of the SCC graph of the PC12 cell network 35
3.2 Attractor detection based on the SCC decomposition 37
3.3 Attractor detection for the HIV1 and pathway networks 38
3.4 Attractor detection based on the new decomposition 39

4.1 Computation of the strong basin 50

5.1 Attractors of the cardiac network 73
5.2 One-step control of the cardiac network 73
5.3 Computational time for one-step control of the cardiac network. . . . 74
5.4 Attractors of the myeloid differentiation network 75
5.5 One-step control of the myeloid differentiation network 76
5.6 Strong basin of granulocytes 76
5.7 Attractors of the tumour network 79
5.8 One-step control of the tumour network 80
5.9 Sequential control of the tumour network 80
5.10 Sequential control of the tumour network with constraints 80
5.11 Source-target control of the biological networks 82
5.12 Computational time for source-target control of the biological networks 82

6.1 Target control of the cardiac network 94
6.2 The number of perturbations required by the target control of the myeloid differentiation network 95
6.3 Target control of granulocytes and monocytes 95
6.4 Target control of megakaryocytes and erythrocytes 95
6.5 Target control of the tumour network 97
6.6 Target control of the biological networks 97
6.7 Computational time for target control of the biological networks . . . 99

7.1 Control methods integrated in CABEAN 102

Layout Notes

Where relevant, boxes such as this one give some context about the research papers written over the course of preparing this thesis.
This information may be useful to reviewers of the thesis, but may not be important for other readers.

Key Statements

At certain locations in this document, key definitions, statements or results will be emphasised in boxes such as this one.

Code Snippets

```
Important code examples are provided in code boxes.
```

Chapter 1

Introduction

1.1 Cell reprogramming

Cell differentiation is the process in which an immature cell becomes a more specialised cell. Previously, researchers think that this process is one-way and could not be reversed. However, this paradigm has been overturned by the discovery of *cell reprogramming*.

In 1962, John Gurgeon proposed the concept of cell reprogramming with his experiment, in which he reprogrammed somatic cells to stem cells [Gur62]. Since then, researchers have been actively working on this field for the last six decades. In their seminal work, Yamanaka *et al.* [Yam07, TTO^{+}07] showed that human somatic cells can be converted to induced pluripotent stem cells (iPSCs) by a cocktail of defined factors: Oct3/4, Sox2, Klf4, and c-Myc. The generated iPSCs have the ability to further propagate and differentiate into many cell types and thus they have great potential for tissue engineering and regenerative medicine. However, the application of iPSC reprogramming is hampered by the following concerns: (1) the generated iPSCs have a risk of tumourigenesis and teratoma formation [Gol19, GD19]; (2) the iPSC reprogramming and the differentiation process usually require lengthy time commitment to generate a sufficient number of desired cells for clinical application, which also results in significant experimental costs [GD19]; and (3) the generated iPSCs often encounter cell cycle arrest after differentiation, which makes it difficult to expand the number of cells for therapeutic transplantation [Gol19].

Limitations of iPSC reprogramming reinforce the need of *direct reprogramming*, also called *transdifferentiation*. Direct reprogramming harnesses the power of somatic cells to regenerate defective or deficient cells by reprogramming abundant somatic cells directly into the desired cells bypassing the pluripotent state. In this way, direct reprogramming can not only reduce the risk of tumourigenesis and teratoma formation, but also shorten the period of time and reduce the experimental costs for generating sufficient desired cells for therapeutic application [GD19].

A key factor in the application of direct reprogramming techniques is to identify effective reprogramming factors, including transcription factors, growth factors, small molecules, RNAs and DNAs [GSPP19]. Experimental approaches usually select promising combinations of targets, perturb them and monitor if the perturbation triggers desired changes. Such "trial and error" approaches can be very expensive and require long-time commitment, which render them inefficient [GSPP19].

Gene regulatory networks (GRNs) are comprehensive networks that encode molecular species (nodes) and their regulation relations (edges) [KS08]. They play a critical role in numerous cellular processes, such as cell differentiation, metabolism and so on. Advances in sequencing techniques and the availability of wealthy data on gene expression profiles promote the shift from experimental approaches to the computational predictions based on mathematical modelling of GRNs. Mathematical models allow us to discover different combinations of targets by providing a broad view on the whole GRNs. In this way, we are able to make predictions in a systematic manner and speed up the development of cell reprogramming, as such predictions are much faster and cheaper than experimental approaches. Moreover, it has great promise to discover novel intervention targets for reprogramming.

1.2 Mathematical modelling of biological systems

"All models are wrong, but some are useful."

George E.P. Box

Mathematical modelling paves the way for the analysis of biological systems and the study of underlying mechanisms. Several modelling frameworks have been developed to model biological systems and each framework has its advantages and limitations. We roughly categorise them into two classes: logical models and continuous models. In this section, we will give a brief and concise review on the popular methodologies. In particular, we will discuss the logical models, including *Boolean networks*, *probabilistic Boolean networks*, and *Bayesian networks*, and the continuous model, *networks of ordinary differential equations (ODEs)*.

Boolean network

Boolean networks, first introduced by Kauffman [Kau69], are a well-established modelling framework for GRNs and their associated signalling pathways. In Boolean networks, molecular species, such as genes and transcription factors, are described as Boolean variables. Each variable is assigned with a Boolean function, which determines the evolution of the variable. Boolean functions characterise activation or inhibition regulations between the molecular species. The states of a Boolean network are binary strings, where every bit of the string represents the state of a molecular species – 0 for inactive (or absent) and 1 for active (or present). The dynamics of a Boolean network is assumed to evolve in discrete time steps, moving from one state to the next, under one of the updating schemes, such as the *synchronous* or *asynchronous* updating scheme. Under the synchronous scheme, all the variables update their values simultaneously at each time step, while under the asynchronous scheme, only one variable is randomly selected to update its value at each time step. The asynchronous updating scheme is considered more realistic than the synchronous one, since it can capture the phenomenon that biological processes occur at different classes of time scales [ZYL$^+$13]. Besides the above mentioned general asynchronous updating scheme, several other non-deterministic updating schemes are also proposed: at each time step, a random selection of m nodes can be updated either synchronously or asynchronously with m being fixed or random.

Boolean networks are restricted by its definition that (1) gene expressions cannot be sufficiently captured by only two states; and (2) the dynamics of Boolean networks cannot depict the stochasticity of biological systems [VCS13]. Still, Boolean networks have apparent advantages compared to other modelling frameworks. They are highly abstract and free from kinetic parameters, thus it is relatively easy to infer Boolean networks from the available data [HLT$^+$09]. The structure of a Boolean network is simple, yet it can capture the important dynamical properties of real-life biological systems [Aku18].

Probabilistic Boolean network

Probabilistic Boolean networks, introduced by *Shmulevich et al.* [SDKZ02], are an extension of Boolean networks to integrate stochasticity and uncertainty of real systems. A probabilistic Boolean network is also made up of binary-valued variables and Boolean functions. Different from Boolean networks, each variable of a probabilistic Boolean network can have several Boolean functions and each function is assigned with a probability based on its compatibility with the microarray gene expression data [KS08, SDKZ02]. The updating of a variable is subject to the Boolean function randomly selected based on the probabilities. Thus, the dynamics of a probabilistic Boolean network is stochastic. The framework of probabilistic Boolean networks enables the analysis of the long-run dynamics with the methods for *discrete-time Markov chains (DTMCs)*. All the long-run characteristics, such as the long-run relative influence and sensitivity, are expressed in terms of steady-state probabilities [MPSY18]. Therefore, the study of probabilistic Boolean networks mainly concentrates on the efficient computation of steady-state probabilities.

Probabilistic Boolean networks preserve appealing features of Boolean networks such as being simple in the structure yet effective at depicting real-life biological systems. The state space of a probabilistic Boolean network is discrete and thus it also suffers from loss of information during the discretisation of the real-time data [KS08]. Due to the high complexity of probabilistic Boolean networks, more temporal data are needed to infer the network [TMP$^+$13].

Bayesian network

Bayesian networks are graphical models that describe probabilistic relationships between variables [SDKZ02]. A Bayesian network is a directed acyclic graph $G = (X, A)$, where $X = \{x_1, x_2, \ldots, x_n\}$ denotes the set of genes and A represents the set of direct edges corresponding to probabilistic dependence interactions between the genes [CLL$^+$14]. The relationships between the genes are represented as a joint probability distribution that can be decomposed into a product of local conditional probabilities and has the following form:

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | Pa(x_i))$$

where $Pa(x_i)$ denotes the values of the parent nodes of x_i [SDKZ02]. In another word, a Bayesian network describes a joint probability distribution that can divide the set of genes into conditional independent subsets that are not overlapped based on their positions [CLL$^+$14].

The advantage of Bayesian networks is that they can handle noise and uncertainty. But they fail to capture temporal information of the time-series data [FLNP00] and their acyclic nature does not not allow feedback loops. Another drawback lies in the computational complexity of inferring the structure of a Bayesian network from gene expression data, especially when a large number of genes are involved [SDKZ02, VCS13].

Network of ordinary differential equations

Network of ODEs is a classic mathematical framework for the modelling and analysis of all kinds of biological networks. It encodes a function for each gene to describe its (instantaneous) changes and therefore provides a quantitative description of the real system. The function is an ordinary differential equation of the following form:

$$\frac{dx(t)}{dt} = f_i(x_1(t), x_2(t), \ldots, x_n(t), u, \theta_i)$$

where $x_i(t), i \in \{1, 2, \ldots n\}$ are the gene expression of gene x_i at time t, $\frac{dx(t)}{dt}$ is the rate of change of gene x_i, u is the external perturbation signal, and θ_i is a set of parameters describing the interactions of the genes [HLT$^+$09, BBAIDB07].

As a continuous model, networks of ODEs provide more details about the dynamics of the system, at a cost that a large amount of high-quality data are needed to find the appropriate parameters [BBAIDB07, KS08, BGMN20].

Comparison of the models

The logical models describe combinatorial regulations of the elements of a particular biological system and provide a qualitative description of the system [SB07]. The plentiful data of microarray gene expression triggers the rapid development of logical models [SB07]. The continuous models are constructed using real-time data and are theoretically more accurate in describing the real systems [KS08]. However, the construction of continuous model usually requires a large amount of data to estimate proper kinetic parameters and the models describing large biological networks are prohibitively complicated [Bor05].

A key criterion for assessing the quality of the model is the similarity of the experimental data and predictions generated based on the model [KS08]. Given the same level of similarity, the simpler model is more appealing. Lack of real-valued data prohibits the modelling of continuous models, such as networks of ODEs [Bor05]. Logical models provide a proper "coarse-grained" level of description of biological networks. They focus on important qualitative properties of the real system and neglect details when it is acceptable. Considering the intrinsic diversity and complexity of biological systems, modelling of large biological networks, such as the probabilistic Boolean networks and the Bayesian networks, is usually prohibited by the incomplete knowledge of the full circuitry, despite of the large amount of available data [Bor05].

Therefore, in this thesis, we select Boolean networks as the representative of biological networks, which is encouraged by its simplicity on even large-scale networks. In particular, we focus on asynchronous Boolean networks, which are considered more realistic than synchronous Boolean networks. Thanks to the rapid expansion of biological databases, a number of large-scale Boolean networks have been constructed to model biological systems. These Boolean networks are interconnected and stable. Thus, they are likely to better withstand the loss of a link and cope with external environmental perturbations, which are important characteristics of biological systems [Sto18].

1.3 Research questions

The principal objective of this thesis is to identify reprogramming factors that can trigger desired changes in biological systems. We study this problem in the context of asynchronous Boolean networks.

In Boolean networks, the steady-state behaviour of the dynamics is described as *attractors*, to one of which the system eventually settles down. Attractors are hypothesised to characterise cellular phenotypes or functional cellular states, such as proliferation, apoptosis and so on [Hua01]. Thus, to reprogram the cells is equivalent to alter the attractor that the Boolean network is in. The prerequisite to this problem is to have a good knowledge of the attractors of a given Boolean network.

Attractor detection

Efficient identification of attractors is one of the major challenges in the analysis of Boolean networks. Mizera *et al.* [MPQY19, YMPQ19] developed a decomposition-based method for the identification of attractors in large synchronous and asynchronous Boolean networks.

Their seminal work uses the simple decomposition to divide the structure of the network into a set of *strongly connected components (SCCs)*. Hence, the efficiency of this SCC decomposition-based attractor detection method will be hampered when a Boolean network has a large number of SCCs, which commonly arises in biological networks. This motivates us to work on the following question:

Research Question 1

Can we optimise the decomposition of Boolean networks to improve the efficiency of attractor detection?

Our goal is to find a better decomposition of Boolean networks, which balances the structure and the dynamics of the networks and improves the efficiency of the decomposition-based method for attractor detection in asynchronous Boolean networks.

Basin computation

The *basin of attraction* of an attractor indicates the commitment of the states to the attractor. In asynchronous Boolean networks, each attractor has a *weak basin* and a *strong basin*, reflecting different levels of commitment to the attractor. The weak basin of an attractor includes the states that can reach this attractor, meanwhile, it is possible that these states can reach some other attractors of the network. Whereas the states in the strong basin of an attractor are fully committed – they can only reach this attractor. An important attractor usually has large basins, which means that this attractor can attract more states [SPRF08]. The computation of the strong basin of an attractor is computationally difficult, which motivates us to work on the following question:

Research Question 2

Can we develop an efficient method for the computation of the strong basin of an attractor?

We aim to develop a decomposition-based method for the strong basin computation, which makes use of the structural and the dynamical properties of Boolean networks.

Source-target control

Once the attractors of the given network are known, we can identify the source and target attractors that correspond to the interested cell phenotypes based on the gene expression profiles. Our next objective is:

Research Question 3

Can we find a set of nodes, the perturbation of which can drive the network from the source attractor to the desired target attractor?

To identify the nodes, whose intervention can drive the network from the source attractor (the initial cell type) to the desired target attractor (the target cell type), is called *source-target control* of Boolean networks. In the application of cell reprogramming, we can reprogram a cell from the initial cell type directly to the desired cell type or via some other cell types. Accordingly, in Boolean networks, the control can be achieved either in one step or in a sequence of steps through other states, called *one-step source-target control* and *sequential source-target control*, respectively. The nodes required to be perturbed are often called *driver nodes* of the associated control. The control can be carried out for different periods of time using *instantaneous*, *temporary* or *permanent perturbations*. We will combine the two categories and explore different control strategies to solve the source-target control problem. Moreover, practical constraints will be taken into consideration to improve the feasibility of the predictions.

To reduce the experimental costs, we are interested in the minimal source-target control. The requirement of minimality is crucial, without which the problem is rendered trivial – simply alter the values of the nodes, that have different values in the source and the target attractors, as the values in the target attractor. Our control methods will be developed based on the concept of basins of attraction, including the weak basin and the strong basin.

Target control

Cells in tissues and in culture normally exist as a population of cells, corresponding to different states [dSB14]. There is a need of *target control* to compute a set of nodes of a network, the control of which can drive the network from any initial state to the desired target attractor. This inspires us to work on the following question:

Research Question 4

Can we find a set of nodes, the perturbation of which can drive the network from any initial state to the desired target attractor?

We aim to develop methods to solve the target control problem with instantaneous, temporary and permanent perturbations.

1.4 Related work

In recent years, several approaches have been developed for the control of complex networks [LSB11, MFKS13, FMKS13, GLDB14, ZnA15, CGC$^+$16, WSH$^+$16, MHP16, MHP17, ZYA17, CKM13]. Among them, the methods [LSB11, GLDB14, CGC$^+$16, CZD$^+$11, NV12, PWHL17] were proposed to tackle the control of networks with linear time-invariant dynamics using either node perturbations or edge perturbations. Specifically, Liu *et al.* [LSB11] developed a structural controllability framework for the full control of complex networks. It computes the minimal set of driver nodes that can modulate the entire dynamics of the network. Later on, Gao *et al.* [GLDB14] adapted this method to solve the the target control problem. They developed a k-walk method and a greedy method to identify a set of driver nodes that can control a target set of nodes. However, Czeizler *et al.* [CGC$^+$16] proved that it is NP-hard to compute a minimal set of driver nodes for the problem of structural target control. They also proposed several heuristics to improve the greedy algorithm [GLDB14]. Benoit *et al.* [CZD$^+$11] first proposed the concept of edge perturbations. Nepusz and Vicsek [NV12] defined a dynamical process on the edges of a network and showed that the controllability of edge dynamics is significantly different from that of node dynamics. Pang *et al.* [PWHL17] generalised the controllability of edge dynamics to all types of complex networks, including directed or undirected networks, weighted or unweighted networks.

The above methods have a common distinctive advantage that they are solely based on the structure of the network, which is exponentially smaller than the number of states in the dynamics. Nevertheless, they are only applicable to systems with linear time-invariant dynamics. While most real systems are nonlinear in nature, linear dynamics cannot depict some of the crucial features of many real-world systems, such as multistability and basins of attraction [CKM13]. Therefore, it is necessary to study the control of nonlinear systems.

The control methods proposed in [MFKS13, FMKS13, ZnA15, WSH$^+$16, MHP16, MHP17, ZYA17, CKM13] are designed for networks governed by non-linear dynamics. Among these methods, the one based on the computation of the feedback vertex set (FVS) [MFKS13, FMKS13, ZYA17] is a purely structure-based method. It drives the network towards a target state by regulating the nodes in a feedback vertex set of the network, which have been proved to be the determining nodes for systems described by ODEs. Cornelius *et al.* [CKM13] proposed a simulation-based method to predict instantaneous perturbations that can reprogram a cell from an undesired phenotype to a desired one for systems of ODEs. Wells *et al.* [WKM15] introduced the optimal least action control (OLAC) method to predict parameter interventions on gene expressions, protein levels or interaction rates for systems described by nonlinear differential equations. Wang *et al.* [WSH$^+$16] developed an experimentally feasible method for the control of nonlinear dynamical networks. They construct the so-called "attractor network" of a system by incorporating all the experimentally validated paths from one attractor to another. However, this method fails to formulate a generic control framework for nonlinear dynamical networks. It also fails to provide a straightforward way to find the control paths for the conversion between two given attractors. All the methods mentioned above are not applicable to Boolean networks.

Several methods based on semi-tensor product (STP) have been proposed to solve different control problems for Boolean control networks (BCNs) under the synchronous updating scheme [LCL17, ZLLC18, LZH$^+$16, ZLK$^+$19, WSZS19, CLW16, YYCJ19, ZKF13]. For synchronous Boolean networks, Kim *et al.* [KPC13] developed a method to compute a small fraction of nodes, called "control kernels", whose modulation can govern the dynamics of the network; and Moradi *el al.* [MGFA19] developed an algorithm guided by forward dynamic programming to solve the control problem. Lin *et al.* [LK12] proposed a Max-SAT based automatic test pattern generate algorithm to identify faulty genes that cause undesired behaviours of GRNs and to identify the best drug selection for cancer treatment. This algorithm considers synchronous Boolean networks under a stuck-at fault model. Murrugarra *el al.* [MVCAL16] proposed a method for the identification of intervention targets based on algebraic techniques for synchronous Boolean networks. However, all these methods are not directly applicable to asynchronous Boolean networks.

Closely related to our work, Mandon *et al.* [MHP16, MHP17] proposed several methods for the control of asynchronous Boolean networks. Particularly, they proposed algorithms to identify reprogramming factors for both existential and inevitable reachability of the target attractor with permanent perturbations [MHP16]. Afterwards, they proposed a method to compute all the control paths between two states with at most k temporary or permanent perturbations [MHP17]. However, these methods do not scale well for large-scale Boolean networks, since they encode all possible control strategies into the transition system of the Boolean network to identify the desired control paths [MHP17]. As a result, the size of the resulting *perturbed transition graph* grows exponentially in the number of perturbations, which renders these methods inefficient. Fontanals *et al.* [FTS20] proposed a method based on trap space to deal with the full network control problem: leading the system to the desired attractor or the phenotype from all possible initial states. Indeed, this full network control problem is equivalent to the target control problem introduced in this thesis. Their method adopts temporal controls while preserving the desired attractors in the transition system under control. However, a temporal control does not necessarily need to preserve the desired attractor because the control will eventually be released to retrieve the original transition system where the desired attractor is in. In Chapter 6, we will develop a target control method to identify a temporary control that may or may not preserve the desired attractor. With this lifted constraint, our method has the potential to identify smaller control sets than the trap space-based method. The stable motifs-based control method (SMC) [ZnA15] predicts a set of transient perturbations that can guide the dynamics from any initial states to the desired target attractor. It takes into account the functional information of the network (network dynamics) and has a substantial improvement in computing the number of driver nodes. This method is very promising, even though it does not guarantee to find the minimal set of driver nodes. In Chapter 6, we will compare this stable motif-based method with our methods for the target control of Boolean networks.

1.5 Contributions

The main contributions of this thesis are summarised as follows:

1. We propose a method towards the optimal decomposition of Boolean networks to balance the relation between the structure and dynamics of a network [SPP19a]. The decomposition problem is transformed into the acyclic partitioning of the directed acyclic graphs (DAGs), which is known to be NP-hard. We plug the new decomposition method into the decomposition-based method for attractor detection [MPQY19] and perform experiments on

a number of biological networks. The results show that the efficiency of attractor detection can be significantly improved with our new decomposition method.

2. We develop a decomposition-based method for the computation of the strong basin of an attractor based on the computation of fixed points of set operations [PSPM18, PSPM19]. Compared to the global method that treats the entire network in one-go, the decompose-based method, which employs the "divide and conquer" strategy, has a great improvement in efficiency for networks that have modular structures, which often occurs in real-life biological networks. The decomposition-based computation of the strong basin forms the foundations for the control of Boolean networks.

3. We design several methods for the minimal one-step source-target control of Boolean networks with instantaneous, temporary and permanent perturbations [PSPM18, PSPM19, SPP19b]. Given a Boolean network, our methods can identify the minimal subsets of nodes, whose instantaneous, temporary or permanent perturbation can drive the dynamics of the network from the source attractor to the target attractor. The minimality of the identified perturbations leads to lower experimental costs and also makes the experimental validation easier to conduct. We apply these methods to a number of Boolean networks that model real-life biological systems to show their efficacy and efficiency.

4. We develop methods for the attractor-based sequential source-target control with instantaneous, temporary and permanent perturbations [MSH+19, SP20c]. These methods compute all the sequential control paths from the source attractor to the target attractor with at most k perturbations. Moreover, these methods only adopt biologically observable attractors as intermediate states, which makes them more practical than the general sequential control [MSP+19], where any state (transient states or steady states) can play the role of intermediate states. These methods provide new ways to reprogram Boolean networks and can potentially reduce the number of perturbations compared to the one-step source-target control.

5. We develop methods to compute instantaneous, temporary and permanent target control for a given target attractor of a Boolean network [SP20b]. These methods explore the strong basin or the weak basin of the target attractor by partitioning the basin into a set of schemata. Each schema leads to a candidate control, which is further verified and reduced. We demonstrate that even for target control, the control of only a small fraction of nodes is sufficient to reprogram the dynamics of the networks.

6. We implement all the methods for the source-target control and the target control of asynchronous Boolean networks in our software CABEAN [SP20a]. CABEAN also integrates the feature that allows users to encode practical conditions on the perturbations and on the intermediate attractors (for attractor-based sequential control) as preconditions for source-target control methods,

such that undesired perturbations or intermediate attractors will be avoided during the computation.

In addition to the contributions listed above, I also contributed to the following works, which are not included in this thesis.

1. We present ASSA-PBN, a software toolbox for modelling, simulation, and analysis of probabilistic Boolean networks [MPSY18]. ASSA-PBN provides efficient statistical methods with three parallel techniques to speed up the computation of steady-state probabilities. ASSA-PBN also supports the in-depth analysis of probabilistic Boolean networks, including the parameter estimation, the long-run influence and sensitivity analysis, and the computation of one-parameter profile likelihoods.
2. We develop a general sequential control method, which establishes a set characterisation of sequential reprogramming for Boolean networks [MSP$^+$19]. This method has no restriction on the nature of the intermediate state, which could be either a transient state, a single-state attractor, or a state in a cyclic attractor. This method guarantees the inevitable reprogramming to the target attractor. It has the ability to identify new targets and lower the experimental costs with fewer interventions.
3. We design a method to predict a minimal set of nodes, whose instantaneous perturbation can modulate the dynamics of the network from any state in the source attractor to the target attractor [BPSP19]. We then extend this method to solve the problems of target control and full control of large-scale Boolean networks with instantaneous perturbations.

1.6 Layout

The thesis is structured as follows.

- In Chapter 2, we introduce preliminary notions of Boolean networks, including the notions on the structure and dynamics of Boolean networks. A number of real-life biological networks, modelled as Boolean networks, are also introduced. These networks are of different sizes and describe different real-life biological processes, such as the myeloid differentiation process, the cardiac development and the FHF/SHF determination. These networks will be used throughout the thesis to evaluate the performance of the proposed methods.
- Chapter 3 introduces a multi-level method for the near-optimal decomposition of Boolean networks. The structure of a Boolean network is decomposed into maximal SCCs to form an SCC graph, which is a directed acyclic graph. Then the SCC graph is partitioned via three steps, viz. initial partitioning, topological refinement and iteration, to reduce the number of partitions and balance their weights. This method is integrated with the decomposition-based method for attractor detection [MPQY19] to improve its efficiency.

- After addressing the problem of attractor detection, we shift our focus to the scalable control of asynchronous Boolean networks. A critical foundation for the control of Boolean networks lies in the computation of strong basin of an attractor. To tackle the state-space explosion problem, in Chapter 4, we develop a decomposition-based method to compute the strong basin of an attractor.

- Chapter 5 studies the source-target control of Boolean networks. We first describe a method for the minimal one-step control with instantaneous perturbations based on the decomposition-based computation of the strong basin. Then we describe constraints for the validation of temporary and permanent control. Based on the constraints and the computation of the strong basin, we develop methods for the minimal one-step control with temporary and permanent perturbations. Afterwards, we extend the three one-step control methods to develop methods for attractor-based sequential control with instantaneous, temporary and permanent perturbations.

- Chapter 6 examines the target control of Boolean networks with instantaneous, temporary and permanent perturbations. Instead of driving the network from one initial state, the methods for target control aim to find solutions that can drive the network from any initial state to the target attractor. In this chapter, we define the concept of schemata which gives candidate solutions, that can be minimised and verified based on the constraints for different kinds of target control problems.

- Chapter 7 introduces our software, CABEAN, which integrates all the control methods proposed in this thesis. In this chapter, we present the general features of CABEAN and illustrate its usage with an example. Detailed instructions on how to use CABEAN are referred to the website of the software: *https://satoss.uni.lu/software/CABEAN/*.

- Finally, Chapter 8 consists of discussions, conclusions and future work. During the analysis of a number of biological networks, we spotted some limitations of the inferred biological networks in the literature. These limitations are summarised in Section 8.1. In the end, we draw some conclusions regarding the overall thesis and discuss some thoughts and open issues that should be considered in the future.

- Chapter 3 is based on the paper entitled “Towards optimal decomposition of Boolean networks” [SPP19a], which was published in IEEE/ACM Transactions on Computational Biology and Bioinformatics (IEEE/ACM TCBB).
- Chapter 4 is written using the content of two publications: one published in the proceedings of 9th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM-BCB 2018) [PSPM18] and the other published in IEEE/ACM TCBB [PSPM19].
- Chapter 5 is written using the content of five publications [PSPM18, PSPM19, SPP19b, MSH$^+$19, SP20c]: one was published in the proceedings of 9th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM-BCB 2018), one was published in IEEE/ACM TCBB; one was published in the proceedings of 23rd International Symposium on Formal Methods (FM 2019); one was published in the proceedings of 17th International Conference on Computational Methods in Systems Biology (CMSB 2019); and the last one was published in the proceedings of 18th International Conference on Computational Methods in Systems Biology (CMSB 2020).
- Chapter 6 is based on the paper entitled “A dynamics-based approach for the target control of Boolean networks” [SP20b], which was published in the proceedings of 11th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM-BCB 2020).
- Chapter 7 is based on the paper entitled “CABEAN: a software for the control of asynchronous Boolean networks” [SP20a], accepted by Bioinformatics.

Chapter 2

Preliminaries

2.1 Boolean networks

In this section, we give preliminary notions of Boolean networks.

2.1.1 Boolean networks

A Boolean network (BN) describes elements of a dynamical system with binary-valued nodes and interactions between elements with Boolean functions. It is formally defined as:

Definition 1 (Boolean networks). *A Boolean network is a tuple* $\mathsf{BN} = (X, F)$ *where* $X = \{x_1, x_2, \ldots, x_n\}$, *such that* $x_i \in X$ *is a Boolean variable and* $F = \{f_1, f_2, \ldots, f_n\}$ *is a set of Boolean functions over* X.

The structure of a Boolean network $\mathsf{BN} = (X, F)$ can be viewed as a directed graph $G(V, E)$, called the *dependency graph* of BN, where $V = \{v_1, v_2 \ldots, v_n\}$ is the set of *nodes*. Node $v_i \in V$ corresponds to variable $x_i \in X$. For every $i, j \in \{1, 2, \ldots, n\}$, there is a directed edge from v_j to v_i, denoted as $e(v_j, v_i)$, if and only if f_i depends on x_j. A *strongly connected component (SCC)* of a directed graph is a maximal strongly connected subgraph. An SCC with no outgoing edges is called a *bottom SCC (BSCC)*.

For the rest of the exposition, we assume an arbitrary but fixed network $\mathsf{BN} = (X, F)$ of n variables is given to us. For all occurrences of x_i and f_i, we assume x_i and f_i are elements of X and F, respectively. A *state* s of BN is an element in $\{0, 1\}^n$. Let S be the set of states of BN. For any state $s = (s[1], s[2], \ldots, s[n])$, and for every $i \in \{1, 2, \ldots, n\}$, the value of $s[i]$, represents the value that x_i takes when the network is in state s. For some $i \in \{1, 2, \ldots, n\}$, suppose f_i depends on $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$. Then $f_i(s)$ will denote the value $f_i(s[i_1], s[i_2], \ldots, s[i_k])$ and $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ are called *parent nodes* of x_i, denoted as $par(x_i)$. For two states $s, s' \in S$, the *Hamming distance* between s and s' will be denoted as $hd(s, s')$ and $\arg(hd(s, s')) \subseteq \{1, 2, \ldots, n\}$ will denote the set of indices in which s and s' differ. For two subsets $S', S'' \subseteq S$, the Hamming distance between S' and S'' is defined as the minimum of the Hamming distances between all the states in S' and all the states in S''. That is, $hd(S', S'') = \min_{s' \in S', s'' \in S''} hd(s', s'')$. We let $\arg(hd(S', S''))$ denote the set of subsets of $\{1, 2, \ldots, n\}$ such that $I \in \arg(hd(S', S''))$ if and only if I is a set of indices of the variables that realise this Hamming distance.

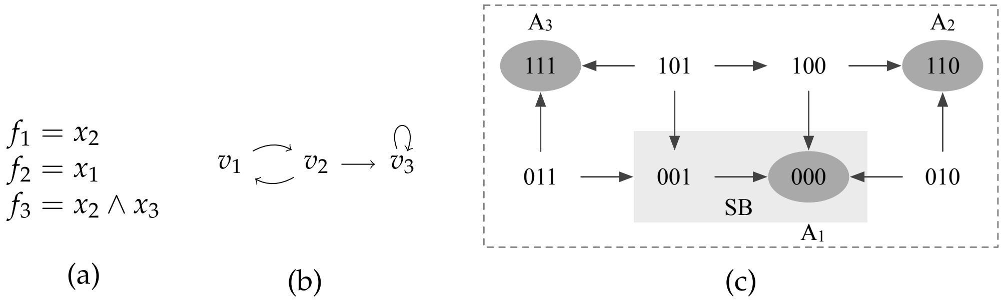


Figure 2.1: (a) Boolean functions, (b) the dependency graph, and (c) the transition system *TS* of Example 1. In (c), we omit selfloops for all the states except for state (101).

2.1.2 Dynamics of Boolean networks

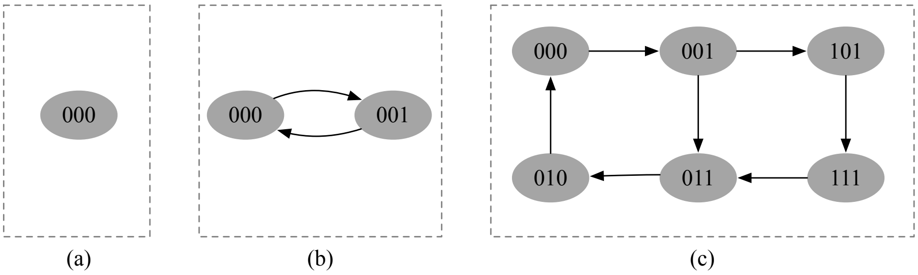
We assume that a Boolean network $\mathsf{BN} = (X, F)$ evolves in discrete time steps. It starts from an initial state s_0 and its state changes in every time step based on the Boolean functions F and the updating schemes. Different updating schemes lead to different dynamics of the network [MPSY18, ZH14]. In this work, we are interested in the *asynchronous updating scheme* as it allows biological processes to happen at different classes of time scales and thus is more realistic. We define *asynchronous dynamics* of Boolean networks as follows.

Definition 2 (Asynchronous dynamics of Boolean networks). *Suppose $s_0 \in S$ is an initial state of* BN. *The asynchronous evolution of* BN *is a function $\xi_{\mathsf{BN}} : \mathbb{N} \to \wp(S)$ such that $\xi_{\mathsf{BN}}(0) = \{s_0\}$ and for every $j \geq 0$, if $s \in \xi_{\mathsf{BN}}(j)$ then $s' \in \xi_{\mathsf{BN}}(j+1)$ is a possible* next state *of s iff either $hd(s, s') = 1$ and there exists i such that $s'[i] = f_i(s) = 1 - s[i]$, or $hd(s, s') = 0$ and there exists i such that $s'[i] = f_i(s) = s[i]$.*

It is worth noting that the asynchronous dynamics is non-deterministic. At each time step, only one node is randomly selected to update its value based on its Boolean function. A different choice may lead to a different next state $s' \in \xi(j + 1)$. Henceforth, when we talk about the dynamics of a Boolean network, we shall explicitly mean the asynchronous dynamics. We describe the dynamics of a Boolean network as a *transition system (TS)*, defined as follows.

Definition 3 (Transition system of Boolean networks). *The transition system of a Boolean network* BN, *denoted as TS, is a tuple $(S, \to_{\mathsf{BN}})$, where the vertices are the set of states S and for any two states s and s' there is a directed edge from s to s', denoted $s \to s'$, iff s' is a possible next state of s according to the asynchronous evolution function ξ of* BN.

Example 1. *Consider a Boolean network $\mathsf{BN} = (X, F)$, where $X = \{x_1, x_2, x_3\}$, $F = \{f_1, f_2, f_3\}$, and $f_1 = x_2$, $f_2 = x_1$ and $f_3 = x_2 \wedge x_3$. The dependency graph of the network* BN *and its associated transition system TS are given in Figure 2.1 (b) and (c). Because the updating of the nodes is non-deterministic, a state can have more than one out-going edge as shown in Figure 2.1 (c).*



(a) (b) (c)

Figure 2.2: Different types of attractors of an asynchronous Boolean network: (a) a singleton attractor, (b) a simple loop, and (c) a complex loop. We omit selfloops for all the states.

2.1.3 Attractors and basins

A *path* ρ from a state s to a state s' is a (possibly empty) sequence of transitions from s to s' in TS. Thus, $\rho = s_0 \to s_1 \to \ldots \to s_k$, where $s_0 = s$ and $s_k = s'$. A path from a state s to a subset S' of S is a path from s to any state $s' \in S'$. An infinite path ρ from s is an infinite sequence of transitions from s. Let $\rho = s_0 \to s_1 \to \ldots$ be an infinite path from s_0. A state $s \in S$ appears *infinitely often* in ρ if for every $i \geq 0$ there exits $j \geq i$ such that $s_j = s$; s appears *finitely often* in ρ otherwise.

Definition 4 (Fairness). *Let $s_0 \in S$. An infinite path $\rho = s_0 \to s_1 \to \ldots$ is said to be unfair if for every state s that occurs infinitely often in ρ, there exists a possible next state s' of s which occurs only finitely often in ρ. ρ is said to be fair otherwise.*

Henceforth, we shall assume that the evolution of BN is always fair, and hence consider only fair paths. We therefore, let $P_\infty(s)$ denote the set of all infinite fair paths from a state $s \in S$.

For any state $s \in S$, let $pre_{TS}(s) = \{s' \in S \mid s' \to s\}$ and let $post_{TS}(s) = \{s' \in S \mid s \to s'\}$, where $s' \to s \in \to_{\mathsf{BN}}$ and $s \to s' \in \to_{\mathsf{BN}}$. $pre_{TS}(s)$ consists of all the states that can reach s by one transition in TS and $post_{TS}(s)$ consists of all the states that can be reached from s by one transition in TS. $pre_{TS}(s)$ and $post_{TS}(s)$ are often called the set of *predecessors* and *successors* of s. Note that, by definition, $hd(s, pre_{TS}(s)) \leq 1$ and $hd(s, post_{TS}(s)) \leq 1$. Operations pre_{TS} and $post_{TS}$ can be lifted to a subset S' of S as: $pre_{TS}(S') = \bigcup_{s \in S'} pre_{TS}(s)$ and $post_{TS}(S') = \bigcup_{s \in S'} post_{TS}(s)$. We define $pre^{i+1}_{TS}(S') = pre_{TS}(pre^{i}_{TS}(S'))$ and $post^{i+1}_{TS}(S') = post_{TS}(post^{i}_{TS}(S'))$ where $pre^{0}_{TS}(S') = post^{0}_{TS}(S') = S'$. For a state $s \in S$, $reach_{TS}(s)$ denotes the set of states s' such that there is a path from s to s' in TS and can be defined as the fixed point of the successor operation which is often denoted as $post^{*}_{TS}$. Thus, $reach_{TS}(s) = post^{*}_{TS}(s)$.

Definition 5 (Attractor). *An attractor A of TS is a minimal non-empty subset of states of S such that for every $s \in A, reach_{TS}(s) = A$.*

Attractors are hypothesised to characterise steady-state behaviours of the network. Any state, which is not in an attractor, is a transient state. An attractor A of TS is said to be *reachable* from a state s if $reach(s) \cap A \neq \emptyset$. The network starting from any initial state $s_0 \in S$ will eventually end up in one of the attractors of TS and remain there forever unless perturbed externally. Thus, it is easy to observe that

Observation 1. Any attractor of TS is a BSCC of TS.

Under the asynchronous updating scheme, there are singleton attractors and cyclic attractors. A singleton attractor contains only one state and a cyclic attractor contains more than one state. Cyclic attractors can be further classified into: (1) a simple loop, in which all the states form a loop and every state appears only once per traversal through the loop; and (2) a complex loop, which has intricate topology and includes several loops. Figure 2.2 (a), (b) and (c) show a singleton attractor, a simple loop and a complex loop, respectively.

Let S' be a subset of states of S. We define the *weak basin* and *strong basin* of S', denoted $bas^W_{TS}(S')$ and $bas^S_{TS}(S')$, as follows.

Definition 6 (Basin)**.** *Let* $S' \subseteq S$.

- ***Weak basin:*** *The weak basin of* S' *with respect to* TS *is defined as* $bas^W_{TS}(S') = \{s \in S \mid reach_{TS}(s) \cap S' \neq \emptyset\}$*, which equals the fixed point of the predecessor operation on* S' *and is often denoted as* $pre^*_{TS}(S')$*. Thus,* $bas^W_{TS}(S') = pre^*_{TS}(S')$*. In other words,*

$$bas^W_{TS}(S') = \{s \in S \mid \exists \rho = s_0 \to s_1 \to \ldots \in P_\infty(s), \exists j \geq 0, s_j \in S'\}$$

- **Strong basin:** *Since all paths in* $P_\infty(s)$ *are fair, the strong basin of* S' *with respect to* TS *is defined as*

$$bas^S_{TS}(S') = \{s \in S \mid \forall \rho = s_0 \to s_1 \to \ldots \in P_\infty(s), \exists j \geq 0, s_j \in S'\}$$

We say that a path $\rho = s_0 \to s_1 \to \ldots$ *eventually* reaches S' if there exists a $j \geq 0$ such that $s_j \in S'$. Intuitively, the weak basin of S' consists of all states from which there is at least one path to S', whereas the strong basin of S' consists of all states from which all paths eventually reach S'. Clearly thus, $bas^S_{TS}(S') \subseteq bas^W_{TS}(S')$.

If S' is an attractor A (say), $bas^W_{TS}(A)$ and $bas^S_{TS}(A)$ will also be referred to as the *weak basin* and *strong basin of attraction* with respect to A, which imply the commitment of states to A. Thus, the weak basin of A includes the states s from which there exists a path to A. It is possible that there also exist paths from s to some other attractor $A' \neq A$ of TS. However, the notion of the strong basin of an attractor does not allow this. Any path from a state s in the strong basin of A will eventually reach A and cannot reach any other distinct attractor $A' \neq A$ of TS. Thus, it is easy to observe that,

Observation 2. *If* $s \in bas^S_{TS}(A)$ *then* $s \notin bas^W_{TS}(A')$ *for any other distinct attractor* A' *of* TS*. Therefore,* $bas^S_{TS}(A) = bas^W_{TS}(A) \setminus (\bigcup_{A'} bas^W_{TS}(A'))$*, where the union is over all attractors* $A' \neq A$ *of* TS*.*

It is worth noting that when S' is an attractor A, if ρ eventually reaches A, the network will get stuck in A for all the following time steps. That is, for every $i \geq 0$, $s_i \in A$ implies $s_j \in A$ for all $j > i$. This follows directly from Definition 5.

Example 2. *The network given in Example 1 has three attractors* $A_1 = \{000\}$*,* $A_2 = \{110\}$ *and* $A_3 = \{111\}$*, indicated as grey nodes in Figure 2.1*(c)*. Taking attractor* A_1 *as* *an example, its strong basin,* $bas_{TS}^{S}(A_1) = \{000, 001\}$, *is shown as the highlighted region; its weak basin,* $bas_{TS}^{W}(A_1) = \{000, 001, 101, 011, 100, 010\}$, *consists of six states.*

2.2 Decomposition of Boolean networks

In this section, we introduce several notions on the decomposition of Boolean networks.

2.2.1 SCC decomposition

The dependency graph of a Boolean network $G(V, E)$ can be decomposed into a set of maximal SCCs, denoted as $\mathcal{H} = \{H_1, H_2, \ldots, H_m\}$, where m is the number of maximal SCCs. We assume that a single node (with or without a self-loop) is always an SCC, although it may not be maximal. The set of maximal SCCs, $\mathcal{H}$, is disjoint and the union of $\mathcal{H}$ equals the whole set of nodes V, namely $H \cap H' = \emptyset$ for any $H \neq H'$ in $\mathcal{H}$, and $\bigcup \mathcal{H} = V$.

The set of SCCs can form a *weighted directed graph* $\mathcal{G}(\mathcal{H}, \mathcal{E})$, which we call the *SCC graph* of a given Boolean network $G(V, E)$. Every vertex $H \in \mathcal{H}$ of $\mathcal{G}$ corresponds to a maximal SCC in the Boolean network. For any pair of SCCs $H,\ H' \in \mathcal{H}$, $H \neq H'$, there is a directed edge from H to H' if and only if there exists a directed edge from a node in SCC H to a node in SCC H' in the dependency graph $G(V, E)$ of BN, denoted $\{e(v, v') \in E | v \in H, v' \in H'\}$. Then, we say SCC H is a *parent SCC* of H' and we let $par(H')$ denote the set of parent SCCs of H'. The *control nodes* of H' is defined as $ctr(H') = (\bigcup_{v \in H'} par(v)) \setminus H'$.

We assign *vertex weight* δ and *edge weight* η, to the SCC graph $\mathcal{G}(\mathcal{H}, \mathcal{E})$ as follows. The vertex weight δ of a vertex $H \in \mathcal{H}$ is a function $\delta : \mathcal{H} \to \mathbb{N}$, such that for every vertex $H \in \mathcal{H}$, $\delta(H) = |H|$, representing the number of nodes contained in the SCC. The edge weight η of $e \in \mathcal{E}$ is a function $\eta : \mathcal{E} \to 2^V$ such that for every edge $e(H, H') \in \mathcal{E}$, $\eta(e)$ is given as $\eta(e) = ctr(H') \cap H$. That is, the weight of an edge $e(H, H')$ corresponds to the set of control nodes of SCC H' from SCC H.

2.2.2 Blocks

Let H be an SCC of BN and $ctr(H)$ be the control nodes of H.

Definition 7 (Basic Block)**.** *A basic block* B *is a subset of the nodes of* V *such that* $B = H \cup ctr(H)$ *for some* $H \in \mathcal{H}$.

Let $\mathcal{B}$ be the set of basic blocks of BN. Since every node of BN belongs to an SCC, we have $\bigcup \mathcal{B} = V$. The union of two or more basic blocks of $\mathcal{B}$ is also called a *block*. For any block $B \in \mathcal{B}$, let $|B|$ denote the number of nodes contained in B.

The set of basic blocks $\mathcal{B}$ can form a directed graph $\mathcal{G}_{\mathcal{B}} = (\mathcal{B}, E_{\mathcal{B}})$, called the *block graph* of BN. In the block graph, the basic blocks are vertices and for any pair of basic blocks $B', B \in \mathcal{B}, B' \neq B$, there is a directed edge from B' to B if and only if $B' \cap B \neq \emptyset$ and for every $v \in (B' \cap B)$, $par(v) \cap B = \emptyset$. In such case, block B' is called a *parent block* of block B and the node v is called a *control node* of B. Let $par(B)$ and $ctr(B)$ denote the set of parent blocks and the set of control nodes of B, respectively.

A block B (basic or non-basic) is called *elementary* if $ctr(B) \subseteq B$. B is called *non-elementary* otherwise. Each block has a topological credit, defined as follows.

Definition 8 (Topological Credit)**.** *Given a* BN*, an elementary block B_i has a credit of* 0*, denoted as $\varphi(B_i) = 0$. Let B_j be a non-elementary block and let $par(B) = \{B_{j1}, B_{j2}, \ldots, B_{jk}\}$ be the set of parent blocks of B_j. The credit of B_j is $\varphi(B_j) = \max_{B_z \in par(B)}(\varphi(B_z)) + 1$.*

Let us assume that BN has m basic blocks and these blocks are topologically sorted based on the topological credits as $\{B_1, B_2, \ldots, B_m\}$. Note that for every $j, 1 \leq j \leq m$, $(\bigcup_{\ell=1}^{j} B_\ell)$ is an elementary block, denoted $\overline{B}_j$. For two basic blocks B and B', where B is non-elementary, B' is said to be an *ancestor* of B if there is a path from B' to B in the block graph $\mathcal{G}_\mathcal{B}$. The *ancestor-closure* of a basic block B (elementary or non-elementary), denoted $\mathsf{ac}(B)$ is defined as the union of B with all its ancestors. Note that $\mathsf{ac}(B)$ is an elementary block and so is $\{\mathsf{ac}(B') \mid B' \in par(B)\}$, which we denote as $\mathsf{ac}(B)^-$.

2.2.3 Projection of states and the cross operation

We assume the nodes $\{v_1, v_2, \ldots, v_n\}$ of the dependency graph G inherit the ordering of the variables X of BN. Let B be a block of BN. Since B is a subset of V, its state space is $\{0,1\}^{|B|}$ and is denoted as S_B.

Next, we define the *projection* of a state $s \in S$ to a subset B of $\{1, 2, \ldots, n\}$, which represents the indices of a subset of nodes $X' \subseteq X$ as follows.

Definition 9 (Projection)**.** *For any state $s \in S$, where $s = (s[1], s[2], \ldots, s[n])$, the projection of s to B, denoted $s|_B$, is the tuple obtained from s by suppressing the values of the variables not in B. Thus, if $B = \{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ then $s|_B = (s[i_1], s[i_2], \ldots, s[i_k])$. Clearly $s|_B \in S_B$. For a subset S' of S, $S'|_B$ is defined as $\{s|_B \mid s \in S'\}$.*

Definition 10 (Cross Operation)**.** *Let B_1 and B_2 be two blocks of* BN *and let s_1 and s_2 be states of B_1 and B_2, respectively. $s_1 \otimes s_2$ is defined (called* crossable*) if there exists a state $s \in S_{B_1 \cup B_2}$ such that $s|_{B_1} = s_1$ and $s|_{B_2} = s_2$. $s_1 \otimes s_2$ is then defined to be this unique state s. For any subsets S_1 and S_2 of S_{B_1} and S_{B_2}, respectively, $S_1 \otimes S_2$ is a subset of $S_{B_1 \cup B_2}$ and is defined as:*

$$S_1 \otimes S_2 = \{s_1 \otimes s_2 \mid s_1 \in S_1, s_2 \in S_2 \textit{ and } s_1 \textit{ and } s_2 \textit{ are crossable}\}$$

Note that $S_1 \otimes S_2$ can be the empty set. The cross operation is easily seen to be associative. Hence for more than two states $s_1, s_2, \ldots, s_k$, $s_1 \otimes s_2 \otimes \ldots s_k$ can be defined as $(((s_1 \otimes s_2) \otimes \ldots) \otimes s_k)$. We have a similar definition for the cross operation on more than two sets of states.

2.2.4 Transition system of the blocks

We define the 'local' transition system of a block inductively, starting from the elementary blocks and moving to the non-elementary blocks further down the topological order. The transition system TS_B of an elementary block B, being either basic or non-basic, is defined exactly as the transition system of a Boolean network (see Definition 3) with the nodes being S_B. This is well-defined since by the definition of an elementary block, the Boolean functions of the nodes of block B do not depend on the values of the nodes outside B. Whereas, the transition system of a non-elementary block B is defined based on the transition systems of its parent blocks because the transition system of B depends on the transition systems of its parent blocks (or its control nodes in its parent blocks).

Let B be a non-elementary basic block of a BN. Let A be an attractor of the transition system of the elementary block $\mathsf{ac}(B)^-$ and let $bas^S_{TS_{\mathsf{ac}(B)^-}}(A)$ be its (strong) basin of attraction in $TS_{\mathsf{ac}(B)^-}$. Then, the transition system of a non-elementary block is defined as follows.

Definition 11 (TS of non-elementary blocks)**.** *The transition system of B generated by $bas^S_{TS_{\mathsf{ac}(B)^-}}(A)$ is defined as a tuple $TS_B = (S, \rightarrow)$ where the set of states S of TS_B is a subset of $S_{\mathsf{ac}(B)}$ such that $s \in S$ if and only if $s|_{\mathsf{ac}(B)^-} \in bas^S_{TS_{\mathsf{ac}(B)^-}}(A)$ and for any two states $s, s' \in S_{\mathsf{ac}(B)}$ there is a transition $s \rightarrow s'$ if and only if either $hd(s,s') = 1$ and $s'[i] = f_i(s) = 1 - s[i]$ where $i = \arg(hd(s,s'))$ or $hd(s,s') = 0$ and there exists an i such that $s'[i] = f_i(s) = s[i]$.*

Note that the construction of the transition system of the non-elementary blocks is different from that introduced in [MPQY19]. There, given a non-elementary block B, the set of states of its 'local' transition system TS_B is a subset of the state space S_B and the transition relations for the control nodes of B were derived by projecting the transitions in the attractor of the parent block of B to these control nodes. In other words, in [MPQY19], the transition system of a non-elementary block B is constructed based on the behaviour of the attractor of the parent blocks of B, whereas here we need the full behaviour of the basin of the attractor of the ancestors of B to generate the transition system of B.

Example 3. *Consider a Boolean network $\mathsf{BN} = (X, F)$, where $X = \{x_1, x_2, x_3, x_4, x_5\}$, $F = \{f_1, f_2, f_3, f_4, f_5\}$, and $f_1 = x_2, f_2 = x_1, f_3 = x_2 \wedge x_3, f_4 = (\neg x_1) \vee x_4 \vee x_5, f_5 = x_1 \vee x_4$. The Boolean functions and the dependency graph of the network are given in Figure 2.3 (a) and (b). This network has three maximal SCCs $\{v_1, v_2\}$, $\{v_3\}$ and $\{v_4, v_5\}$, which give rise to three blocks: $B_1 = \{v_1, v_2\}$, $B_2 = \{v_2, v_3\}$, and $B_3 = \{v_1, v_4, v_5\}$. Block B_1 is an elementary block. Its 'local' transition system TS_{B_1} is given in Figure 2.3 (c). In TS_{B_1}, there are two single-state attractors $\{00\}$ and $\{11\}$. The strong basin of attractor $\{00\}$ in TS_{B_1} is shown as the highlighted region. Block B_3 is a non-elementary block with block B_1 as its parent block and node $v_1 \in B_1$ as the corresponding control node. The transition system of B_3 based on the strong basin of attractor $\{00\}$ in TS_{B_1} is given in Figure 2.3 (d). Although block B_3 consists of three nodes, the state space S_{B_3} of TS_{B_3} is a subset of $\mathsf{ac}(B_3)$. Hence, a state in S_{B_3} is made up of the values of x_1, x_2, x_4 and x_5.*

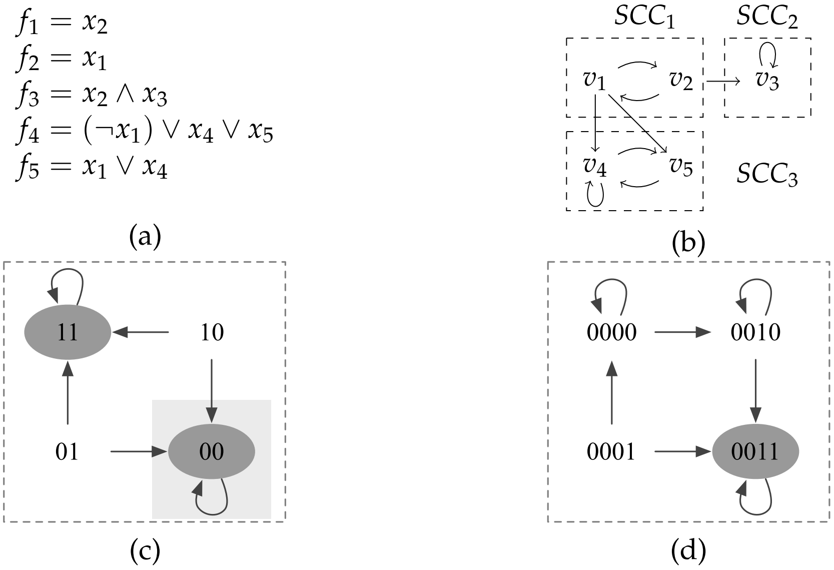


Figure 2.3: (a) Boolean functions, (b) the decomposition of the dependency graph, (c) the transition system of block B_1 and (d) the transition system of block B_3 based on the strong basin of attractor $\{00\}$ of TS_{B_1}.

2.3 Acyclic partitioning of directed acyclic graphs

Let $G(V, E)$ be a directed graph and let $n = |V|$ and $m = |E|$ represent the number of vertices and edges contained in G, respectively. A directed graph is called a *directed acyclic graph* (DAG) if it does not contain any directed cycles. Every DAG has a topological ordering, i.e. an ordering of the vertices V that for every edge $e(u, v), u, v \in V$, vertex u comes before vertex v in the ordering. Similar to Definition 8, a vertex u with no parent has a topological credit of 0, denoted as $\varphi(u) = 0$. A vertex v with a set of parents $par(v)$ has a topological credit of $\varphi(v) = \max_{v' \in par(v)}(\varphi(v')) + 1$.

A DAG is said to be *weighted* if it has associated *vertex weight* and *edge weight*. The vertex and edge weights of G are defined as functions $\delta : V \to \mathbb{N}$ and $\eta : E \to \mathbb{N}$. It is easy to observe that

Observation 3. *The SCC graph $\mathcal{G}_{\mathcal{H}}$ and the block graph $\mathcal{G}_{\mathcal{B}}$ are DAGs.*

In the SCC graph $\mathcal{G}_{\mathcal{H}}$ and the block graph $\mathcal{G}_{\mathcal{B}}$, the directed edges are all from the parent SCCs (blocks) to the child SCCs (blocks), and there are no directed cycles.

A *k-way partitioning* of a weighted DAG $G(V, E)$ divides the vertices V into k disjoint subsets $\{V_1, V_2, \ldots, V_k\}$. For any partition $V_i, i \in [1, k]$, its weight equals $\sum_{v \in V_i} \delta(v)$. A *cut set* is a set of edges whose endpoints are in different partitions, i.e. $\{e(u, v) \in E \mid u \in V_i, v \in V_j\}$, where $V_i, V_j \subseteq V$ and $V_i \neq V_j$. The *weight of a cut set* Γ is the sum of the weights of the edges in the set, $\sum_{e \in \Gamma} \eta(e)$. The formal definition of a cut set is given below.

Definition 12 (Cut set)**.** *A cut set of a weighted DAG $G(V, E)$ is a set of edges with the following properties:*

- *the removal of all edges in the set disconnects G;*

- *the removal of some (but not all) of the edges in the set does not disconnect G.*

The partitioning of a weighted DAG that maintains the acyclic property between different partitions, is called *acyclic partitioning*. In general, the maximal weight of a partition is limited by an upper bound L_{max}. Here, we give the formal definition of the acyclic partitioning problem of a weighted DAG.

Definition 13 (Acyclic partitioning of a weighted DAG). *Given a weighted DAG* $G(V, E)$ *and an upper bound* L_{max}*, find a partitioning* $P = \{V_1, V_2, \ldots, V_k\}$*, where* $V_1 \cup \ldots \cup V_k = V$ *and* $V_i \cap V_j = \emptyset$ $(i \neq j)$*, such that the maximal weight of each partition is not greater than* L_{max} *and the weight of the cut set is minimised. Moreover, the relation between* $V_1, V_2, \ldots, V_k$ *is acyclic.*

2.4 Real-life biological networks

In this section, we introduce several real-life biological networks that model different biological systems/processes. In Chapters 3, 4, 5 and 6, we use these networks to evaluate the performance of our methods for the new decomposition, the computation of the strong basin, source-target control and target control of Boolean networks. Moreover, we give an in-depth analysis of some of the networks to demonstrate the consistency of the identified predictions with wet-lab observations. During the analysis of these networks, we gained a better understanding of the properties of biological networks and observed flaws of the constructed networks. In Section 8.1, we will discuss these flaws. Moreover, we will propose some suggestions for the refinement of the networks.

Now we introduce the biological systems/processes described by the networks and give an overview of the number of nodes, the number of edges and the number of singleton and cyclic attractors contained in each network in Table 2.1. We refer details on the networks, such as the Boolean functions, to their original works.

- The cell cycle network of the fission yeast is constructed based on known biochemical interactions to recap regulations of the cell cycle of the fission yeast [DB08]. This is a small network with 10 nodes. It has 12 singleton attractors and 1 cyclic attractor.

- The myeloid differentiation network is designed to model myeloid differentiation from common myeloid progenitors to megakaryocytes, erythrocytes, granulocytes and monocytes [KMST11]. This network has 11 nodes and 6 attractors, 4 of which agree with microarray expression profiles of two different studies.

- The cardiac gene regulatory network integrates major genes that play essential roles in early cardiac development and FHF/SHF determination [HGZ$^+$12]. It has 15 nodes and 3 attractors under the initial condition defined in [HGZ$^+$12].

- The ERBB receptor-regulated G1/S transition protein network combines ERBB signalling with G1/S transition of the mammalian cell cycle to identify new

Network	# nodes	# edges	# singleton attractors	# cyclic attractors
yeast	10	28	12	1
myeloid	11	30	6	0
cardiac	15	39	3	0
ERBB	20	52	3	0
HSPC-MSC	26	81	2	2
tumour	32	158	9	0
hematopoiesis	33	88	5	0
PC12	33	62	7	0
bladder	35	116	3	1
psc-bFA	36	237	4	0
co-infection	52	136	30	0
MAPK	53	105	12	0
CREB	64	159	8	0
HGF	66	103	10	0
bortezomib	67	135	5	0
T-diff	68	175	12	0
HIV1	136	327	8	0
CD4+	188	380	6	0
pathway	321	381	3	1

Table 2.1: An overview of the biological networks.

targets for breast cancer treatment [SFL$^+$09]. It consists of 20 nodes and 3 singleton attractors.

- The HSPC-MSC network of 26 nodes describes intercommunication pathways between hematopoietic stem and progenitor cells (HSPCs) and mesenchymal stromal cells (MSCs) in bone marrow (BM) [EMMP16]. It has 2 singleton attractors and 2 cyclic attractors.

- The tumour network is built to study the role of individual mutations or their combinations in the metastatic process [CMR$^+$15]. This network contains 32 nodes and 9 attractors, which are consistent with [CMR$^+$15].

- The network of hematopoietic cell specification covers major transcription factors and signalling pathways for the development of lymphoid and myeloid [CvOO$^+$17]. This network is made up of 33 nodes and contains 5 steady states.

- The PC12 cell differentiation network [OKS$^+$16] is a comprehensive model for the clarification of the cellular decisions towards proliferation or differentiation. It models the temporal sequence of protein signalling, transcriptional responses as well as the subsequent autocrine feedbacks [OKS$^+$16]. It has 33 nodes and 7 singleton attractors.

- The bladder cancer network of 35 nodes allows one to identify the deregulated pathways and their influence on bladder tumourigenesis [RRC$^+$15].

This network consists of 4 attractors, corresponding to growth arrest and cell proliferation.

- The model of mouse embryonic stem cells can capture the signal-dependent emergence of cell subpopulations under different initial conditions [YKOO+18]. It has 36 nodes and 4 singleton attractors under the bFGF+Activin (bF+A) condition.

- The model of immune responses is constructed to study the immune responses against single and co-infections with the respiratory bacterium and the gastrointestinal helminth [TPM+12]. This network consists of 52 nodes, one of which is an input node. We detect 30 singleton attractors, when there are no restrictions on the initial values of the input nodes.

- The MAPK network is constructed to study the MAPK responses to different stimuli and their contributions to cell fates [GCBP+13]. In this paper, we use the MPAK mutant r4, which has 53 nodes and 12 attractors.

- The CREB network is a complex neuronal network, whose output node is the transcription factor adenosine 3′, 5′-monophosphate (cAMP) response element–binding protein (CREB) [ATE08]. It is composed of 64 nodes and 8 singleton attractors.

- The model for HGF-induced keratinocyte migration captures the onset and maintenance of hepatocyte growth factor-induced migration of primary human keratinocytes [SNK+12]. It has 66 nodes and 10 attractors.

- The model of bortezomib responses can predict responses to the lower bortezomib exposure and the dose-response curve for bortezomib [COAM15]. It has 67 nodes and 5 singleton attractors.

- The Th-cell differentiation network models regulatory elements and signalling pathways controlling Th-cell differentiation [NCCT10]. It consists of 68 nodes and 12 attractors.

- The HIV-1 network models dynamic interactions between human immunodeficiency virus type 1 (HIV-1) proteins and human signal-transduction pathways that are essential for activation of CD4+ T lymphocytes [ODRS14]. This network contains 136 nodes and 8 attractors.

- The $CD4^+$ T-cell network allows us to study downstream effects of $CAV1^{+/+}$, $CAV1^{+/-}$ and $CAV1^{-/-}$ on cell signalling and intracellular networks [CHS+14]. This network is comprised of 188 nodes and 6 attractors under certain initial conditions.

- The model of signalling pathways central to macrophage activation integrates four crucial signalling pathways that are triggered early-on in the innate immune response [RRL+08]. It consists of 321 networks and has 4 attractors under certain initial condition.

Chapter 3

Near-optimal Decomposition of Boolean Networks

3.1 Introduction

Efficient detection of attractors is one of the major challenges in the analysis of large-scale biological networks. This problem is non-trivial because attractors of a Boolean network are determined based on the states in the transition system, the number of which is exponential in the size of the network. In asynchronous Boolean networks, this problem gets even worse as the non-determinism property results in a more complex transition system compared to synchronous Boolean networks.

It has been known for a while that to detect the attractors of a Boolean network efficiently, an algorithm should exploit both the structure and the dynamics of the network [GR16]. In this spirit, Mizera *et al.* [MPQY19] developed an SCC decomposition-based attractor detection method for asynchronous Boolean networks. The main idea of this method is to decompose the dependency graph of a given Boolean network into a set of maximal SCCs and construct an SCC graph, which is a weighted DAG. In the SCC graph, every vertex corresponds to an SCC. There is a directed edge between two vertices if and only if one of the SCC depends on the nodes of the other SCC. After the SCC decomposition, blocks formed by SCCs and their control nodes, are sorted based on the topological ordering. Then, attractor detection is performed in every block, whose 'local' transition system is constructed based on the behaviour of the control nodes in the attractors of parent blocks. In the end, all the 'local' attractors are combined block-wise to derive the attractors of the entire network. The vital principle that guarantees the correctness of this method is that the decomposed components of the network should form a DAG [MPQY19]. It is worth noting that the efficiency of this method is hindered when a Boolean network has a large number of SCCs, which commonly arises in biological networks. This motivates the need for the optimal decomposition of Boolean networks, which balances the structure and the dynamics of the blocks and can potentially boost the efficiency of the decomposition-based attractor detection.

In this chapter, we address the optimal decomposition of Boolean networks by the acyclic partitioning of SCC graphs, resulting from the SCC decomposition of Boolean networks. In this way, we can balance the sizes of different components, while maintaining the acyclic property between the components. There are a few requirements when searching for the optimal partitioning of the SCC graphs. The first is that the weight of each partition should not exceed an upper bound. This upper bound is the core in balancing the relationship between the structural and the dynamical properties of Boolean networks and therefore should be determined cautiously. If the upper bound is too large, the partitioned graph might have partitions that are almost equal to the entire graph. It can be costly to analyse the dynamics of such partitions in terms of computational time and memory. On the other hand, if the upper bound is too small, there will be a large number of partitions. In such case, even though the state space of each partition is small, it might still be very time-consuming to traverse all the partitions. It is worth noting that biological networks usually have large SCCs that cover a large portion of the nodes, thus, we allow the upper bound to be smaller than the weights of SCCs and let the SCC, whose weight overruns the upper bound, constitute a single partition. The second requirement is that the number of control nodes of each partition needs to be minimised. This is mainly due to the fact that the attractor detection is conducted in blocks, which are formed by partitions together with their control nodes. By minimising the number of control nodes, the sizes of blocks will be reduced as well, leading to smaller 'local' transition systems. Last but not least, the number of partitions should be minimised. Fewer partitions lead to fewer blocks. As a consequence, we analyse fewer local transition systems of reasonable size.

The problem of acyclic partitioning of SCC graphs is similar to that of acyclic partitioning of DAGs. It partitions a weighted DAG into disjoint subsets and minimises the weight of the cut set and the number of partitions. Acyclic partitioning of DAGs is known to be NP-hard [GJ79]. Hence, efficient algorithms for general cases are highly unlikely. Extensive works have been done in the literature to address the acyclic partitioning of special classes of graphs and to find efficient but non-optimal solutions [Ker71, CLB94, FER$^+$13, MPS17]. The most relevant work is a multi-level graph partitioning method [HKU$^+$17]. This method consists of three phases: coarsening, initial partitioning and refinement. It has been demonstrated that this method can compute partitions of higher quality than similar tools and algorithms in the literature [HKU$^+$17]. However, this method is not applicable to the acyclic partitioning of SCC graphs of Boolean networks. In an SCC graph, the weight of an edge is represented as a set of nodes rather than an integer, such that the weight of a cut set is the union of the weights of the edges in the set without duplicate elements. Furthermore, different from the work in [HKU$^+$17], where the number of partitions is given as a predefined value, we aim to compute a minimal number of partitions, the size of which is within a threshold determined by the Boolean network at hand.

In this chapter, we introduce our method towards the optimal decomposition of Boolean networks. The rest of this chapter is organised as follows:

- In Section 3.2, we give the formal definition of acyclic partitioning of an SCC graph.

- In Section 3.3, we introduce our multi-level partitioning of SCC graphs, which includes three steps: initial partitioning, topological refinement and iteration.
- In Section 3.4, we combine the new decomposition method with the SCC decomposition-based attractor detection.
- In Section 3.5, we compare the efficiency of the attractor detection integrated with the new decomposition and the SCC decomposition on several biological networks introduced in Section 2.4.

3.2 Problem definition

As discussed in Section 2.2.1, the SCC decomposition of a Boolean network results in a set of maximal SCCs, which form a weighted DAG $\mathcal{G}(\mathcal{H}, \mathcal{E})$ called the SCC graph. In the SCC graph, every vertex corresponds to an SCC. The weight of a vertex $H \in \mathcal{H}$ is an integer representing the number of nodes contained in the SCC. The weight of an edge $e(H, H') \in \mathcal{E}$ represents the set of control nodes of the child SCC H' from its parent SCC H.

The acyclic partitioning of an SCC graph is an instance of the acyclic partitioning of a weighted DAG (Definition 13). Given a Boolean network $G(V, E)$, the SCC decomposition results in an SCC graph $\mathcal{G}(\mathcal{H}, \mathcal{E})$. The acyclic partitioning of $\mathcal{G}(\mathcal{H}, \mathcal{E})$ divides the vertices $\mathcal{H}$ into a set of disjoint subsets $\{V_1, V_2, \ldots, V_k\}$. These subsets form a directed graph $G(P, E_P)$, where $P = \{V_1, V_2, \ldots, V_k\}$ is the set of nodes and E_P is the set of edges. For every $\{1, 2, \ldots, k\}$, there is a directed edge from V_i to V_j if and only if there is a directed edge from $H \in V_i$ to $H' \in V_j$ in $\mathcal{G}(\mathcal{H}, \mathcal{E})$. The weight of a vertex $V_i \in P$ equals the number of nodes contained in V_i, denoted $\delta(V_i)$. The weight of an edge e from V_i to V_j, denoted $\eta(e)$, is a set $\{u \mid e(u, v) \in E, u \in V_i, v \in V_j\}$, where E is the set of edges in $G(V, E)$.

Similar to other DAG partitioning problems, the partitioning of an SCC graph sets an upper bound $L_{\max}$ on the partition weight. The upper bound is the key to balancing the structure and dynamics of the partitions. However, biological networks often contain large SCCs that cannot be split any further [AB02]. Thus, we allow the upper bound $L_{\max}$ to be smaller than the vertex weight. In this way, each SCC, whose weight is equal or greater than the upper bound, forms a single partition and the remaining SCCs, whose weights are smaller than the upper bound, are split into a minimal set of partitions without exceeding the upper bound of the partition weight. This is motivated by the fact that after the decomposition of the Boolean network, we perform analysis on the local transition systems of the blocks, which can be very time-consuming if there exist too many blocks. Besides that, the size of the cut set weight, i.e. the number of control nodes, should be minimised. Since a block is formed by a partition and its control nodes, fewer control nodes lead to smaller blocks with smaller 'local' transition systems. The above constraints naturally lead to the formal definition of the acyclic partitioning of an SCC graph.

Definition 14 (Acyclic partitioning of an SCC graph)**.** *Given an upper bound $L_{\max}$ and an SCC graph $\mathcal{G}(\mathcal{H}, \mathcal{E})$, which is a weighted DAG with vertex weights δ and edge weights*

η, find a partitioning $P = \{V_1, V_2, \ldots, V_k\}$, where $V_1 \cup \ldots \cup V_k = \mathcal{H}$ and $V_i \cap V_j = \emptyset (i \neq j)$, satisfying the following constraints.

1. *The SCC $H_i \in \mathcal{H}$, whose vertex weight $\delta(H_i) \geq L_{\max}$, forms a partition. The other SCCs are divided into partitions, such that the weight of each partition is equal or smaller than $L_{\max}$.*
2. *The weight of the cut set $\eta(\Gamma)$ is minimised.*
3. *The number of partitions k is minimised.*
4. *The directed graph formed by $P = \{V_1, V_2, \ldots, V_k\}$ is still a DAG.*

The acyclic partitioning of an SCC graph differs from the acyclic partitioning of a DAG mainly in two aspects.

1. The weight of a cut set Γ is different. In the SCC graph partitioning, the weight of an edge is a set of nodes. There may exist duplicate nodes in the weights of different edges. The weight of a cut set Γ is the union of the weights of edges in Γ. In the DAG partitioning, the weight of an edge is an integer, hence, the weight of a cut set Γ is the sum of the weights of the edges in Γ, i.e. $\eta(\Gamma) = \sum_{e \in \Gamma} \eta(e)$.
2. The number of partitions k is not predefined in the SCC graph partitioning. With the constraint on the partition weight, we want to get as few partitions as possible.

In this chapter, we propose a new decomposition method consisting of two steps: the SCC decomposition of a given network and the acyclic partitioning of the resulting SCC graph. The problem of SCC decomposition has been solved in [MPQY19]. Thus, the course of this chapter is to develop a method for the acyclic partitioning of an SCC graph.

Example 4. *The PC12 cell differentiation network consists of* 33 *nodes and* 7 *singleton attractors [OKS$^+$16]. We refer to the original work [OKS$^+$16] for the structure of the network derived from the Boolean functions. The SCC decomposition decomposes the structure of the network into* 19 *maximal SCCs. These SCCs form an SCC graph given in Figure 3.1, where each vertex represents an SCC.* 16 *out of the* 19 *SCCs are single-node SCCs, thus, their vertex weights equal* 1*. The rest of the SCCs, including SCCs* 3, 15 *and* 16, *contain* 3, 2 *and* 12 *nodes, respectively. Thus, the weights of vertices* 3, 15 *and* 16 *in the SCC graph are* 3, 2 *and* 12, *respectively. According to the definition of the edge weight of an SCC graph (see Section 2.2.1), the weight of an edge from H to H' equals the set of control nodes, $ctr(H') \cap H$. We can see that in order to analyse this* 33*-node network, we need to loop through* 19 *blocks, which hampers the efficiency of the SCC decomposition-based methods.*

3.3 Acyclic partitioning of SCC graphs

We propose a method for the acyclic SCC graph partitioning based on the method in [HKU$^+$17]. Our method contains three phases: initial partitioning, topological

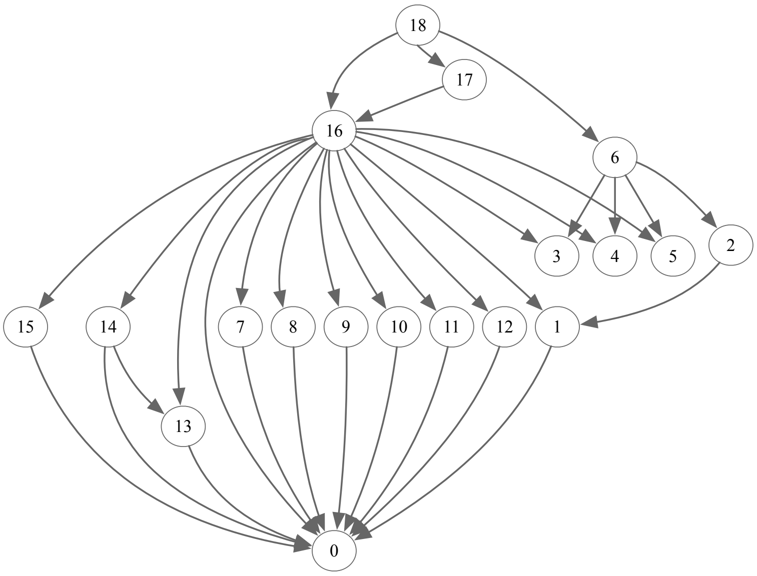


Figure 3.1: The SCC graph of the PC12 cell network. Each vertex represents an SCC.

refinement and iteration. The initial partitioning performs a preliminary partitioning of the vertices based on the topological ordering. In the topological refinement phase, we refine the preliminary solution by adjusting the boundary vertices as well as the partitions at the same topological level. Furthermore, we iterate the initial partitioning and the topological refinement until there is no more improvement in the partitioning results.

3.3.1 Initial partitioning

We adapt the greedy algorithm in [HKU$^+$17] to perform the initial partitioning of an SCC graph. In this phase, we compute an initial solution based on the topological ordering of SCCs.

The inputs of our algorithm include an SCC graph $\mathcal{G}(\mathcal{H}, \mathcal{E})$ and an upper bound of the partition weight $L_{\max}$. The upper bound $L_{\max}$ is an important parameter of the algorithm. The SCC graph is a weighted DAG, constructed from the SCC decomposition of a given Boolean network (see Section 2.2.1). The vertices $\mathcal{H}$ are sorted according to their topological credits, denoted $\varphi(u), u \in \mathcal{H}$. Each vertex u in $\mathcal{H}$ is assigned with two properties: *part* and *free*, representing the partition that u belongs to and the mobility of u, respectively. The mobility of a vertex u reflects if u is movable or not. We consider a vertex is *movable* if it has no predecessors or the mobility *free* of its direct predecessors are *false* as in [HKU$^+$17].

The *gain* of moving a vertex u to a partition j is computed based on the status of all its predecessors and successors after the move, as described in [HKU$^+$17]. Among a number of movable vertices, the best vertex to be moved is the one with the lowest topological credit and the highest gain. It is worth noting that, to maintain the acyclic dependency of the graph, a vertex u should not be moved to a partition $j, j \in [1, k]$, if $\varphi(u) - max_{v \in V_j}(\varphi(v)) > 1$ or $min_{v \in V_j}(\varphi(v)) - \varphi(u) > 1$.

To perform the initial partitioning, we first initialise the number of partitions $k = 0$ and set all the vertices to movable. Then, we repeat the following steps as long as there exist movable vertices.

1. Save movable vertices to a set *set*.
2. For every vertex $u \in set$, compute the gain of assigning u to partition k and insert a tuple $(u, \varphi(u), gain)$ to a queue *queue*.
3. Sort the tuples in *queue* with respect to the topological credits in ascending order first and then again, in each credit level, according to *gain* in descending order. Note that when we sort *queue* based on *gain*, the sorted topological ordering is preserved since the sorting is done separately for each credit level.
4. If the total weight of partition k is less than the upper bound $L_{\max}$, we traverse the tuples in *queue* based on the sorted order and perform the following actions.
    (a) Release the first item $(u, \varphi(u), gain)$ in *queue*.
    (b) If partition k is empty, we assign vertex u to partition k and set the maximum and the minimum topological credits of partition k as $\varphi(u)$. Otherwise, vertex u can be assigned to partition k if (1) the weight of partition k will not overrun the upper bound $L_{\max}$ after the assignment; and (2) $\varphi(u) - max_{v \in V_k}(\varphi(v)) \leq 1$ and $min_{v \in V_k}(\varphi(v)) - \varphi(u) \leq 1$.
    (c) If vertex u is assigned to partition k, we set $free(u)$ as *false* and check if its direct successors are movable or not. If a direct successor of u, denoted v, is movable, we compute the gain of assigning v to partition k, insert $(v, \varphi(v), gain)$ to *queue* and sort *queue* as Step 3.
5. If *queue* is empty, we are done. Otherwise, we increase k by one and go to step 1.

After the initial partitioning, the value of k is the number of partitions we computed so far. The results might be improved in the phases of topological refinement and iteration.

Remark. The differences of our initial partitioning with the one in [HKU$^+$17] mainly lie in three aspects.

1. The computation of the gain to assign a vertex u to a partition j is different. In the SCC graph, we extend an edge weight from an integer to a set of nodes. The cut set weight is the union of the weights of the edges in the set without duplicate elements. We consider the initial size of the cut set weight as the number of all the control nodes in the Boolean network. The gain indicates the decrease in the size of the cut set weight. Since the weights of two edges may have duplicate nodes, to compute the cut set weight, a simple sum of the edge weights as [HKU$^+$17] is not applicable. Let M denote the set of predecessors of u that belongs to partition j, and Q denote the set of successors of u. The gain of assigning the vertex u to a partition j is $|\bigcup_{v \in M} \eta(e(v, u))| -$

$|\bigcup_{p \in Q} \eta(e(u,p))|$. A larger gain indicates a larger decrease in the size of the cut set weight.

2. We take both the topological credit and the gain into account to decide the assignment. The vertices are traversed according to the ascending order of their topological credits. When a vertex is assigned to some partition, the mobility of its child vertices may become movable. More importantly, merely considering the gain may violate the acyclic property. Given two vertices $u, v \in \mathcal{H}$, $\varphi(u) < \varphi(v)$, it may happen that v is assigned to a partition i and u is assigned to a partition j that has a higher credit. This may induce loops in the partitioned graph. Hence, we propose the heuristic that the topological credit has a higher priority than the gain. All the movable vertices are processed first according to their topological credits. With the same credits, the assignment of the vertex with the highest gain will be handled first.

3. We only assign a vertex u to a partition j, if $\varphi(u) - max_{v \in V_j}(\varphi(v)) \leq 1$ and $min_{v \in V_j}(\varphi(v)) - \varphi(u) \leq 1$. This condition is adapted to guarantee the acyclic property of the partitioning.

3.3.2 Topological refinement

The objective of the topological refinement is to refine the preliminary solution based on the topological information. The refinement is achieved in two steps: the refinement of the *boundary vertices* and the refinement of the partitions with the same topological credits.

Refinement of the boundary vertices. Boundary vertices include *incoming boundary vertices* and *outgoing boundary vertices*. A vertex is an incoming boundary vertex if it has no predecessors or all its predecessors are in other partitions. Similarly, a vertex is an outgoing boundary vertex if it has no successors or all its successors are in other partitions. Note that after the initial partitioning, only the boundary vertices can be moved without violating the acyclic dependency. An incoming boundary u can be moved to a partition $j, j \in [1, k]$ if (1) partition j contains at least one predecessor of vertex u; and (2) partition j has the maximum topological credit among the partitions that meet the first constraint. In a similar way, an outgoing boundary v can be moved to a partition j if (1) partition j contains at least one successor of v; and (2) partition j has the minimum topological credit among the partitions satisfying the previous constraint. It is worth noting that the violation of either constraint may lead to cycles in the partitioned graph. For the cases, where multiple partitions satisfy the constraints, the one with the largest gain will be considered.

We adapt the Fiduccia-Mattheyses (FM)-like, move-based direct k-way refinement algorithm in [HKU$^+$17] to refine the initial partitioning. We propose two modifications.

1. Motivated by the same reason explained in Section 3.3.1, we modify the computation of the gain when moving a vertex u from the current partition $part(u)$ to another partition j in this phase. The gain denotes the increase/decrease in the size of the cut set weight $\eta(\Gamma)$ after the movement. Let M denote the set of predecessors/successors of u that are in partition j, Q denote the set of predecessors/successors of u that are not in partition j. If $part(u) > j$, the gain is $|\bigcup_{v \in M} \eta(v,u)| - |\bigcup_{p \in Q} \eta(u,p)|$. Otherwise, the gain is $|\bigcup_{v \in M} \eta(u,v)| - |\bigcup_{p \in Q} \eta(p,u)|$. We drop the movements that cause an overrun on the total weight of a partition j.

2. The refinement may cause empty partitions, thus, the property *part* of vertices and the number of parts k will be updated after all possible movements.

Refinement of the partitions with the same topological credits. Suppose the SCCs are divided into k partitions. All the partitions are topologically sorted and the maximal credit is φ_{max}. If $k > \varphi_{\max} + 1$[1], we perform refinement of the partitions with the same topological credits to minimise the number of partitions. This step is straightforward yet effective. We group the partitions with the same topological credits, if the weight of the new partition is not greater than the upper bound $L_{\max}$. In this way, every partition is considered as a whole and the partitioning is refined at the topological level.

3.3.3 Iteration

Despite the effectiveness of the initial partitioning and the topological refinement, there is still room for further improvement. In the initial partitioning and the refinement of the boundary vertices, we compute solutions at the level of vertices. For large graphs with complex structures, to compute a near-optimal partitioning, it is insufficient to traverse the vertices only once. The refinement of the partitions with the same topological credits can only improve the partitioning at the same topological level.

In order to achieve a near-optimal partitioning, we iteratively apply the initial partitioning and the topological refinement to the partitioned graph. Suppose we have a solution $P = \{V_1, V_2, \ldots, V_k\}$ after the initial partitioning and the refinement. Considering each partition $V_i \in P$ as a vertex, we can form a DAG $G(P, E_P)$ as described in Section 3.2. We perform the initial partitioning and the topological refinement to the partitioned graph $G(P, E_P)$, convert the new solution to a DAG and repeat this process, until the number of partitions k stays the same. During the iteration, every partition is considered as a vertex. We utilise the initial partitioning and the topological refinement to group these partitions. In this way, we can improve the partitioning crossing different topological levels.

Example 5. *For the purpose of illustration, we describe the partitioning results of the PC12 cell differentiation network in Figure 3.2 and Table 3.1. The upper bound of a partition weight is* 12*, which is the maximal size of the SCCs of the PC12 cell network. In Table 3.1,*

---

[1] The topological credit $\varphi(v)$ counts from zero.

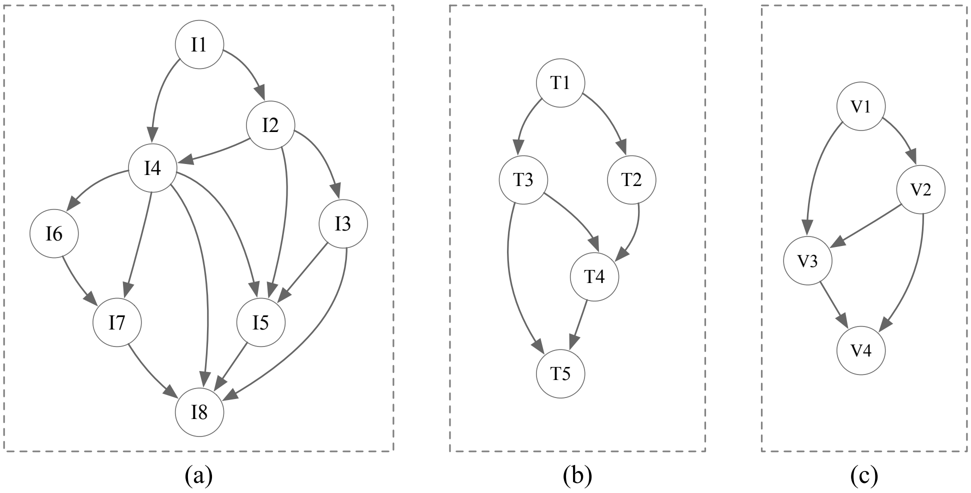


Figure 3.2: Partition of the SCC graph of the PC12 cell network. (a), (b) and (c) show the results of initial partitioning, topological refinement and iteration, respectively.

Initial partitioning		Topological refinement		Iteration	
Partitions	SCCs	Partitions	SCCs	Partitions	SCCs
$I1$	$[18]$	$T1$	$[17,18]$	$V1$	$[17,18,2,6]$
$I2$	$[6,17]$	$T2$	$[2,6]$	$V2$	$[16]$
$I3$	$[2]$	$T3$	$[16]$	$V3$	$[1,3-5,7-12]$
$I4$	$[16]$	$T4$	$[1,3-5,7-12]$	$V4$	$[0,13,14,15]$
$I5$	$[1,3-5,7-12]$	$T5$	$[0,13,14,15]$		
$I6$	$[14,15]$				
$I7$	$[13]$				
$I8$	$[0]$				

Table 3.1: Partition of the SCC graph of the PC12 cell network.

the indices of SCCs are consistent with Figure 3.1. Figure 3.2 (a) shows the results of the initial partitioning phase: 19 *SCCs are partitioned into* 8 *partitions. Figure 3.2 (b) gives the results of the topological refinement. In this phase, by adjusting the boundary vertices, the number of partitions is reduced to* 5*. Among the five partitions, partitions T_2 and T_3 are at the same topological level. However, they cannot be merged into one group because the total weight of these two partitions exceeds the upper bound. The two phases (the initial partitioning and the refinement) are iterated three times to improve the partitioning. In the second iteration, the two partitions T_1 and T_2 in Figure 3.2 (b) are merged. The third iteration shows that the results cannot be improved any further. In the end, the dependency graph of the PC12 cell network is decomposed into* 4 *partitions as shown in Figure 3.2 (c).*

3.4 Integration with the SCC decomposition

As discussed in Section 3.1, the SCC decomposition of a Boolean network often leads to a large number of SCCs as illustrated by the PC12 cell network given in

Example 4. To analyse the SCCs one by one can be time-consuming, which slows down the efficiency of the SCC decomposition-based attractor detection. In Section 3.3, we introduced our method for the acyclic partitioning of SCC graphs. Here, we integrate it with the SCC decomposition to form a near-optimal decomposition of Boolean networks, formulated as the function "FORM_BLOCK($\mathcal{G}$)". The overall procedures of the new decomposition are described below.

1. Decompose the structure of a given Boolean network into a set of maximal SCCs and construct the associated SCC graph $\mathcal{G}(\mathcal{H}, \mathcal{E})$ as described in Section 2.2.1.
2. Apply the acyclic partitioning of the SCC graph to optimise the decomposition as described in Section 3.3.
3. Form blocks by combing partitions with their control nodes.
4. Sort the blocks in ascending order according to their topological credits.

We replace the SCC decomposition with our near-optimal decomposition for the decomposition-based attractor detection method [MPQY19] to improve its efficiency.

3.5 Evaluation

To demonstrate the efficiency of our new decomposition method, we apply attractor detection with the SCC decomposition and the new decomposition on several real-life biological networks introduced in Section 2.4. The SCC decomposition-based attractor detection decomposes the structure of the network into maximal SCCs, form blocks based on the maximal SCCs, compute local attractors in each block and then merge all the local attractors to obtain the attractors of the entire network. The attractor detection based on the new decomposition follows similar procedures except that it optimises the decomposition by partitioning the SCC graph and by forming blocks based on the optimised decomposition. The new decomposition reduces the number of components and balances the weights of the components. Note that both methods identify the exact attractors of Boolean networks, hence, the comparison focuses on the computational efficiency.

The decomposition-based attractor detection is implemented in our software tool ASSA-PBN [MPY15, MPSY18], based on the model checker MCMAS [LQR17] to encode Boolean networks into the efficient data structure - binary decision diagrams (BDDs). The SCC graph partitioning is implemented in Python based on the Python module graph-tool [Pei14]. All the experiments are performed on a high-performance computing (HPC) platform, which contains CPUs of Intel Xeon Gold 6132 @2.6 GHz.

We first describe the results of SCC decomposition-based attractor detection in Table 3.2. The second and the third columns describe the number of nodes and the

Network	# nodes	# edges	# SCCs	# singleton attractors	# cyclic attractors	Time (seconds)
yeast	10	28	3	12	1	0.011
myeloid	11	30	2	6	0	0.001
cardiac	15	39	13	3	0	0.004
ERBB	20	52	14	3	0	0.004
tumour	32	158	13	9	0	0.509
hematopoiesis	33	88	11	5	0	0.664
PC12	33	62	19	7	0	0.012
bladder	35	116	10	3	1	0.909
psc-bFA	36	237	10	4	0	16.780
co-infection	52	136	7	30	0	100.329
MAPK	53	105	18	12	0	2.911
CREB	64	159	39	8	0	1.302
HGF	66	103	26	10	0	1.803
bortezomib	67	135	28	5	0	7.256
T-diff	68	175	34	12	0	5.305
HIV1	136	327	62	8	0	67.177
CD4+	188	380	86	6	0	188.537
pathway	321	381	302	3	1	258.461

Table 3.2: Results of the SCC decomposition-based attractor detection.

number of edges contained in each network. The fourth column describes the number of maximal SCCs derived from the SCC decomposition. The fifth and the sixth columns describe the number of singleton attractors and cyclic attractors of each network. The last column gives the computational time for the SCC decomposition-based attractor detection. We can see that even for the largest network with 321 nodes, the computation finishes within few minutes.

For the new decomposition, the upper bound $L_{\max}$ of the partition weight is an important parameter for balancing the compositions. To analyse its influence on the efficiency of the new decomposition-based attractor detection, we perform the SCC graph partitioning with a list of upper bounds, computed with the formula $L_{\max} = \max(|H|) * scale$, where $\max(|H|)$ is the maximal size of SCCs and the variable *scale* takes values from 0.1 to 1 with an increment of 0.1. The attractors identified by the new decomposition-based attractor detection with different upper bounds are the same with the attractors computed by the SCC decomposition-based attractor detection, since both methods decompose the networks into DAGs.

Table 3.3 shows the results of the new decomposition-based attractor detection on the HIV1 network [ODRS14] and the network of signalling pathways central to macrophage activation (pathway for short) [RRL$^+$08]. The second column gives a list of scales, the third column gives the maximal size of the SCCs, and the fourth columns gives the number of partitions returned by the new decomposition. The last two columns describe the computational time of the new decomposition-based attractor detection and its speedups computed with the formula $speedup = T/T_{new}$, where T and T_{new} represent the time for attractor detection with the SCC decomposition and the new decomposition, respectively. Different scales (different upper

Network	Scale	max($\|H\|$)	# partitions	Time (seconds)	Speedups
HIV1	0.1	35	12	36.359	1.8
	0.2	35	6	24.245	2.8
	0.3	35	5	31.626	2.1
	0.4	35	4	40.115	1.7
	0.5	35	3	49.077	1.4
	0.6	35	3	49.286	1.4
	0.7	35	3	49.508	1.4
	0.8	35	3	48.477	1.4
	0.9	35	3	47.406	1.4
	1.0	35	3	47.876	1.4
pathway	0.1	15	302	16.759	15.4
	0.2	15	110	7.803	33.1
	0.3	15	88	6.388	40.5
	0.4	15	63	4.932	52.4
	0.5	15	55	4.799	53.9
	0.6	15	43	4.877	53.0
	0.7	15	40	4.93	52.4
	0.8	15	31	4.844	53.4
	0.9	15	32	151.912	1.7
	1.0	15	27	3.025	85.4

Table 3.3: Results of the new decomposition-based attractor detection for the HIV1 and pathway networks.

bounds $L_{\max}$) result in different number of partitions. In general, a larger upper bound leads to fewer partitions.

According to Table 3.2, the HIV1 network has in total 62 maximal SCCs. Among the SCCs, there is one huge SCC with 75 nodes and the remaining 61 SCCs are all single-node SCCs. The new decomposition greatly reduces the number of partitions as shown in Table 3.3. Even when the scale is 0.1 ($L_{\max} = 7.5$), the new decomposition method reduces the number of components from 62 to 12. When scale is 0.2 ($L_{\max} = 15$), the new decomposition obtains the highest speedup 2.8. The pathway network has 321 nodes and its structure is decomposed into 302 maximal SCCs. This network does not contain any huge SCCs. The largest SCC is made up of 15 nodes. Hence, its upper bounds are much smaller than those of the HIV1 network. When the scale equals 1.0, the new decomposition gains the highest speedup on the attractor detection. This network also shows that a larger upper bound somehow does not guarantee fewer partitions. This is mainly due to that SCCs with lower topological credits have a higher priority to be assigned to a partition, which might influence the partitioning. For these two networks, with the new decomposition method, the efficiency of the attractor detection is greatly improved.

It is obvious that the scale or the upper bound has an influence on the speedups: the trend of the speedups fluctuates with the increase of the scale. For the other networks in Table 3.2, we noticed that the new decomposition method usually gains higher speedups, when the scale is around 0.6. Table 3.4 summaries the results of the new decomposition-based attractor detection on these networks with

Network	Scale	$\max(\lvert H \rvert)$	# partitions	Time (seconds)	Speedups
yeast	0.6	8	2	0.009	1.2
myeloid	0.6	8	2	0.001	1.0
cardiac	0.6	3	13	0.002	2.0
ERBB	0.6	4	10	0.003	1.3
tumour	0.6	21	3	0.737	0.7
hematopoiesis	0.6	23	3	0.566	1.2
PC12	0.6	12	5	0.008	1.5
bladder	0.6	22	3	0.396	2.3
psc-bFA	0.6	27	3	11.405	1.3
co-infection	0.6	47	3	104.64	1.0
MAPK	0.6	37	3	1.75	1.7
CREB	0.6	26	4	0.097	13.4
HGF	0.6	41	3	1.86	1.0
bortezomib	0.6	38	3	3.486	2.1
T-diff	0.6	35	4	4.63	1.1
HIV1	0.6	75	3	49.286	1.4
CD4+	0.6	103	4	78.363	2.4
pathway	0.6	15	43	4.877	53.0

Table 3.4: Results of the new decomposition-based attractor detection for the biological networks.

$scale = 0.6$. We can see that when the scale equals 0.6, the new decomposition can obtain efficiency gains for most of the networks, except for the tumour network. The tumour network has 8 components with the SCC decomposition. When the scale is 0.2, the number of components is reduced to 5 and the new decomposition obtains the highest speedups of 1.1. When the scale is equal to or greater than 0.5, the number of components stabilises at 3 and the new decomposition method underperforms the SCC decomposition. This example shows that a higher scale does not always lead to a better performance. We do not give the detailed results regarding to the number of iterations to obtain the final decomposition, while the experiments indicate that the iteration improves the partitioning in terms of the number of partitions. To compute the final results, the initial partitioning and the topological refinement are usually iterated for two to three times.

3.6 Conclusion

In this chapter, we proposed a new decomposition of Boolean networks to improve the efficiency of the decomposition-based attractor detection. The new decomposition consists of two steps: the SCC decomposition and the acyclic partitioning of the SCC graphs. The partitioning of the SCC graphs is considered as the acyclic DAG partitioning while minimising (1) the size of the weight of the cut set and (2) the number of partitions without exceeding (3) the upper bound of the partition weight if the partition consists of more than one SCC. We developed a multi-level method to partition the SCC graphs in three phases: initial partitioning, refinement and iteration. We showed that with the new decomposition of Boolean networks, the number of components is reduced and the efficiency of the decomposition-based attractor detection is much improved.

The efficiency of the new decomposition-based attractor detection are influenced by many factors, including the network density, the upper bound of a partition weight, the weight of the cut set. We plan to perform intensive experiments on biological networks to eventually find the optimal decomposition of Boolean networks that suits well for the decomposition-based method for attractor detection.

Chapter 4

Computation of the Basin of Attraction

4.1 Introduction

In Boolean networks, attractors are hypothesised to characterise cell phenotypes or cell fates. The concept of basin of attraction implies the commitment of the states to the attractor. A larger basin implies a large number of states converging to the attractor. We hypothesise that the more important the attractor is, the larger the basin gets [SKI$^+$20].

In synchronous dynamics of a Boolean network, all the nodes update their values simultaneously at each time step. Thus, the synchronous dynamics is deterministic: each state has only one successor and can only reach one of the attractors in the transition system. This indicates that the basins of attractors are disjoint in synchronous dynamics, which is not the case in asynchronous dynamics. In asynchronous dynamics, at each time step, only one node is randomly selected to update its value based on the Boolean function. The asynchronous dynamics of a Boolean network is non-deterministic: a state can have multiple successors and can reach one or more attractors of the network. To distinguish the level of commitment to an attractor, the basin of attraction is subdivided into the *weak basin* and the *strong basin*. The weak basin of an attractor includes the states that can reach this attractor. It is possible that a state in the weak basin can also reach other attractors of the network. The concept of strong basin does not allow this. The strong basin implies the absolute commitment of the states to the attractor. A state in the strong basin of an attractor can only reach this attractor.

Due to the non-deterministic nature, it is known that the computation of the strong basin of an attractor in asynchronous dynamics is computationally difficult and does not scale well to large-scale networks. In this chapter, we develop an algorithm for the strong basin computation based on the fixed point computation. This algorithm starts from the weak basin, WB, and recursively removes states that can reach some state outside WB until WB reaches a fixed point. The remaining states form the strong basin of the attractor. However, owing to the infamous state-space explosion problem, a simple global method that treats the entire network in one-go is highly inefficient. Since most real-life biological networks are large, there is a strong need for an algorithm that can address this problem efficiently. One strategy is to exploit both the structural and dynamical properties of the network at the same time. In this spirit, we develop a decomposition-based method for the computation of the strong basin of an attractor. This method employs the "divide and conquer" strategy to solve the problem in a block-wise manner. It divides the structure of the network into a set of blocks, computes the 'local' strong basin of the projection of the attractor in each block using the fixed point computation, and merges all the 'local' strong basins to obtain the entire strong basin of the attractor. This method can be plugged into any analysis methods that requires the computation of strong basins.

The rest of the chapter is organised as follows:

- In Section 4.2, we introduce the global method for the computation of the strong basin of an attractor based on the fixed point computation.
- In Section 4.3, we introduce the decomposition-based method for the computation of the strong basin of an attractor.
- In Section 4.4, we apply the two methods to a number of biological networks to evaluate their performance.

4.2 A global method

We first describe a procedure for computing the strong basin of an attractor based on the computation of fixed point. This algorithm will act as a reference for comparing the decomposition-based algorithm which we shall later develop.

We define an operator $\mathcal{F}$ on the states S as follows. For any subset T of states:

$$\mathcal{F}(T) = T \setminus (pre_{TS}(post_{TS}(T) \setminus T) \cap T)$$

It is easy to see that $\mathcal{F}$ is monotonically decreasing and hence its greatest fixed point exists. We want to show that for any attractor A of TS, $\mathcal{F}^{\infty}(bas^{W}_{TS}(A)) = bas^{S}_{TS}(A)$. That is, to compute the strong basin of A, one can start with its weak basin and apply the operator $\mathcal{F}$ repeatedly till a fixed point is reached which gives its strong basin. The operation has to be repeated m times where m is the index of $\mathcal{F}^{\infty}(bas^{W}_{TS}(A))$. To prove it, we first prove the following lemmas.

Lemma 1. *For any state $s \in S$, if $s \notin bas^{S}_{TS}(A)$ then $s \notin \mathcal{F}^{\infty}(bas^{W}_{TS}(A))$.*

Proof. Suppose for some $s \in S$, $s \notin bas^{S}_{TS}(A)$. Then either (i) there is no path from s to A or (ii) there is a path from s to another attractor $A' \neq A$ of TS. If (i) holds then $s \notin bas^{W}_{TS}(A)$ and hence $s \notin \mathcal{F}^{\infty}(bas^{W}_{TS}(A))$. Now suppose (ii) holds and there is a path from s to another attractor $A' \neq A$. Consider the shortest such path $s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s_n$, where $s_0 = s$ and $s_n \in A'$ and let $s_i \rightarrow s_{(i+1)}$, $0 \leq i < n$ be the first transition along this path that moves out of $bas^{W}_{TS}(A)$. That is, $s_i \in bas^{W}_{TS}(A)$ but $s_{(i+1)} \notin bas^{W}_{TS}(A)$. We claim that $s \notin \mathcal{F}^{j}(bas^{W}_{TS}(A))$ for all $j \geq (i+1)$. That is, s

is removed in the $(i+1)$th step in the inductive construction of $\mathcal{F}^{\infty}(bas^{W}_{TS}(A))$. We prove this by induction on i.

Suppose $i = 0$. Then there is already a transition from s out of $bas^{W}_{TS}(A)$ and hence $s \in pre_{TS}(post_{TS}(bas^{W}_{TS}(A)) \setminus A) \cap bas^{W}_{TS}(A)$. Thus $s \notin \mathcal{F}(bas^{W}_{TS}(A))$. Next, suppose $i > 0$ and the premise holds for all $j : 0 \leq j < i$. Then by induction hypothesis we have $s_1 \notin \mathcal{F}^{i}(bas^{W}_{TS}(A))$. Hence $s \in (pre_{TS}(post_{TS}(\mathcal{F}^{i}(bas^{W}_{TS}(A))) \setminus \mathcal{F}^{i}(bas^{W}_{TS}(A))) \cap \mathcal{F}^{i}(bas^{W}_{TS}(A)))$ and s will be removed in the $(i+1)$th step of the inductive construction. □

For the converse direction, first, we can easily observe from the definition of weak and strong basins (Definition 6) that:

Lemma 2. *Let A be an attractor of TS. Then*

- $bas^{S}_{TS}(A) \subseteq bas^{W}_{TS}(A)$,
- *for any state $s \in S$, $s \in bas^{S}_{TS}(A)$ iff, for all transitions $s \to s'$, we have $s' \in bas^{S}_{TS}(A)$.*

Thus, we have:

Lemma 3. *For any state $s \in S$, if $s \notin \mathcal{F}^{\infty}(bas^{W}_{TS}(A))$ then $s \notin bas^{S}_{TS}(A)$.*

Proof. For some state $s \in S$, if $s \notin \mathcal{F}^{\infty}(bas^{W}_{TS}(A))$ then either $s \notin bas^{W}_{TS}(A)$, in which case $s \notin bas^{S}_{TS}(A)$ by Lemma 2 or $s \in bas^{W}_{TS}(A)$ but gets removed from $\mathcal{F}^{\infty}(bas^{W}_{TS}(A))$ at the ith step of the inductive construction for some $i \geq 1$. We do an induction on i to show that in that case $s \notin bas^{S}_{TS}(A)$. Suppose $i = 1$. Then by definition $s \in (pre_{TS}(post_{TS}(bas^{W}_{TS}(A)) \setminus bas^{W}_{TS}(A)) \cap bas^{W}_{TS}(A))$ which means there is a transition from s to some $s' \notin bas^{W}_{TS}(A)$. Thus $s \notin bas^{S}_{TS}(A)$ by Lemma 2. Next suppose $i > 1$ and the premise holds for all $j : 1 \leq j < i$. Then, $s \in (pre_{TS}(post_{TS}(\mathcal{F}^{(i-1)}(bas^{W}_{TS}(A))) \setminus \mathcal{F}^{(i-1)}(bas^{W}_{TS}(A))) \cap \mathcal{F}^{(i-1)}(bas^{W}_{TS}(A)))$. There is a state $s' \in \mathcal{F}^{(i-1)}(bas^{W}_{TS}(A))$ such that there is a transition from s to s'. But since by induction hypothesis $s' \notin bas^{S}_{TS}(A)$ we must have $s \notin bas^{S}_{TS}(A)$ by Lemma 2. □

Given an attractor A, we can compute the weak basin $bas^{W}_{TS}(A)$ by a simple iterative fixed-point procedure. Indeed, $bas^{W}_{TS}(A)$ is the smallest subset WB of S such that $A \subseteq \mathrm{WB}$ and $pre_{TS}(\mathrm{WB}) \subseteq \mathrm{WB}$. We call this procedure COMP_WEAK_BASIN which will take as arguments the tuple update functions F and the attractor A.

Now we introduce our algorithm called COMP_STRONG_BASIN, described in Algorithm 1, for the computation of the strong basin of an attractor A based on the fixed point computation. Initially WB is equal to the weak basin of A (Line 2). In each iteration of Line 6, we take the current set WB, which is a subset of the weak basin of A, and remove from it all the states that have transitions to any state outside the current WB (line 8). These are the states from which there are paths to some other attractor $A' \neq A$ and hence they cannot be in the strong basin of A. Finally, when SB stabilises, the remaining part is the strong basin of A. The most important step of this algorithm is the while loop (lines $4-9$), which is repeated till the set SB

Algorithm 1 Fixed point computation of the strong basin

```
procedure COMP_STRONG_BASIN($F, A$)
    WB :=COMP_WEAK_BASIN($F, A$)
    SB := $\emptyset$
    while SB $\neq$ WB do
        if SB $\neq \emptyset$ then
            WB := SB
        end if
        SB := WB $\setminus$ ($pre_{TS}(post_{TS}(\mathsf{WB}) \setminus \mathsf{WB}) \cap \mathsf{WB}$)
    end while
end procedure
```

settles down to a fixed point, which is the strong basin of A. Combining Lemma 1 and Lemma 3, we can prove the correctness of Algorithm 1.

Theorem 1 (Correctness of Algorithm 1). *For any attractor A of TS we have*

$$bas^S_{TS}(A) = F^{\infty}(bas^W_{TS}(A))$$

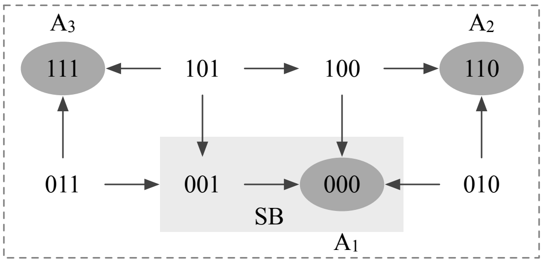


Figure 4.1: Transition system TS of Example 6. We omit selfloops for all the states except for state (101).

Example 6. *We use the Boolean network given in Example 1 to illustrate the computation of the weak and strong basins of an attractor. The transition system of the Boolean network is given in Figure 4.1. We use procedures* COMP_WEAK_BASIN *and* COMP_STRONG_BASIN *to compute the weak and strong basins of $A_1 = \{000\}$, denoted as $bas^W_{TS}(A_1)$ and $bas^S_{TS}(A_1)$, respectively. Intuitively, $bas^W_{TS}(A_1)$ is obtained by applying $pre^*_{TS}(A_1)$ until it reaches a fixed point, which is $bas^W_{TS}(A_1) = \{000, 001, 010, 011, 100, 101\}$. $bas^S_{TS}(A_1)$ is computed by recursively removing all the states in the weak basin of A_1, $bas^W_{TS}(A_1)$, that can reach states outside $bas^W_{TS}(A_1)$, until it reaches a fixed point, which is $bas^S_{TS}(A_1) = \{000, 001\}$.*

Note that Algorithm 1 is worst-case exponential in the size of the input (the description of BN). Indeed, since the strong basin of attraction of A might well be equal to all the states of the entire transition system TS which is exponential in the description of BN. Now, although an efficient algorithm for this problem is highly unlikely, it is possible that when the network has a certain well-behaved structure, one can do better than this global method. Most of the previous attempts at providing such an algorithm for such well-behaved networks either exploited exclusively the structure of the network or failed to provide the exact strong basin. Here we show that, when we take both the structure and the dynamics into account, we can have an algorithm developed later, which is much more efficient than the global method for certain networks.

4.3 A decomposition-based method

In this section, we introduce a method to compute the strong basin of an attractor A based on the decomposition of Boolean networks. The method is based on that of [MPQY19] for computing the attractors of asynchronous Boolean networks. The overall idea is as follows. The network is divided into blocks based on its maximal SCCs. The blocks are then sorted topologically resulting in a dependency graph of the blocks which is a DAG. The transition systems of the blocks are computed inductively in the sorted order and the attractor A is then projected to these blocks. The local strong basins for each of these projections are computed in the transition system of the particular block. These local strong basins are then combined to compute the global strong basin $bas^S_{TS}(A)$.

In the preliminaries (Section 2.2), we have introduced the SCC decomposition of a Boolean network, including several notions on blocks. We now give the key results of the construction of blocks, which will form the basis of the decomposition-based method for the strong basin computation.

Let us start with the case where the given Boolean network BN has two basic blocks B_1 and B_2. We shall later generalise the results to the case where BN has more than two basic blocks by inductive arguments.

Note that either one or both of the blocks B_1 and B_2 are elementary. If only one of the blocks is elementary, we shall without loss generality, assume that it is B_1. Let TS, TS_1 and TS_2 be the transition systems of BN, B_1 and B_2, respectively. If B_2 is non-elementary, we shall assume that TS_2 is the transition system of B_2 generated by the strong basin of attractor A_1 of TS_1.

The states of a transition system will be denoted by s with appropriate subscripts and/or superscripts. For any state $s \in TS$, we shall denote $s|_{B_1}$ by s_1 and $s|_{B_2}$ by s_2. Similarly, for a set of states T of TS, T_1 and T_2 will denote the set of projections of the states in T to B_1 and B_2 respectively.

Let $B_1^- = B_1 \setminus B_2$ and $B_2^- = B_2 \setminus B_2$. We shall denote any transition $s \longrightarrow s'$ in TS by $s \xrightarrow{B} s'$ if the variable whose value changes in the transition is in the set B.

Lemma 4. *For an elementary block B_i of* BN *and for every s_i, s'_i of TS_i, if there is a path from s_i to s'_i in TS_i, then there is a path from s to s' in TS such that $s|_{B_i} = s_i, s'|_{B_i} = s'_i$ and $s|_{B_j^-} = s'|_{B_j^-}$, $j \neq i$.*

Proof. Let B_i be elementary and suppose $s_i^0 \xrightarrow{B_i} s_i^1 \xrightarrow{B_i} \dots \xrightarrow{B_i} s_i^m$, where $s_i^0 = s_i$ and $s_i^m = s'_i$, be a path from s_i to s'_i in TS_i. Let $s|_{B_j^-} = s'|_{B_j^-} = s_j^-$. It is clear that $(s_i^0 \otimes s_j^-) \xrightarrow{B_i} (s_i^1 \otimes s_j^-) \xrightarrow{B_i} \dots \xrightarrow{B_i} (s_i^m \otimes s_j^-)$ is a path from s to s' in TS where $s = (s_i^0 \otimes s_j^-), s' = (s_i^m \otimes s_j^-)$ and s and s' have the required properties. Indeed, since B_i is elementary and values of the nodes in B_j^- are not modified along the path. □

Lemma 5. *For every s, s' of TS if there is a path from s to s' in TS then there is a path from s_i to s'_i in TS_i for every elementary block B_i.*

Proof. Suppose $\rho = s^0 \to s^1 \to \dots s^m$, where $s^0 = s$ and $s^m = s'$ be a path from s to s' in TS. Let B_i be an elementary block of BN. We inductively construct a path ρ_i from s_i to s'_i in TS_i using ρ. ρ_i^j, $0 \leq j < m$, will denote the prefix of ρ_i constructed in the jth step of the induction. Initially $\rho_i^0 = s_i^0$. Suppose ρ_i^j has been already constructed and consider the next transition $s^j \to s^{j+1}$ in ρ. If this transition is labeled with B_i then we let $\rho_i^{j+1} = \rho_i^j \xrightarrow{B_i} s_i^{j+1}$. Otherwise if this transition is labeled with $B_j^-, j \neq i$, then we let $\rho_i^{j+1} = \rho_i^j$. Since by induction hypothesis ρ_i^j is a path in TS_i and we add to this a transition from ρ only if a node of the elementary block B_i is modified in this transition, such a transition exists in TS_i. Hence, ρ_i^{j+1} is also a path in TS_i. Continuing in this manner, we shall have a path from s_i to s'_i in TS_i at the last step when $j + 1 = m$. □

Lemma 6. *Suppose B_1 and B_2 are both elementary blocks and $B_1 \cap B_2 = \emptyset$. Then for every $s, s' \in TS$, there is a path from s to s' in TS if and only if there is a path from s_i to s'_i in every TS_i.*

Proof. Follows directly from Lemma 4 and Lemma 5. □

Lemma 7. *Let B_1 and B_2 be two elementary blocks of* BN*, $B_1 \cap B_2 = \emptyset$. Then we have that A is an attractor of TS if and only if there are attractors A_1 and A_2 of TS_1 and TS_2 resp. such that $A = A_1 \otimes A_2$.*

Proof. Follows directly from Lemma 6. □

Lemma 8. *Let* BN *have two blocks B_1 and B_2 where B_2 is non-elementary, B_1 is elementary and is the parent of B_2. Then we have A is an attractor of TS if and only if A_1 is an attractor of TS_1 and A is also an attractor of TS_2 where TS_2 is realised by $bas_{TS}^{S}(A_1)$.*

Proof. Suppose A is an attractor of TS and for contradiction suppose A_1 is not an attractor of TS_1. Then either there exist $s, s' \in A$ such that there is no path from s_1 to s'_1 in TS_1. But that is not possible by Lemma 5. Or there exist $s_1 \in A_1$ and $s'_1 \notin A_1$ such that there is a transition from s_1 to s'_1. But then by Lemma 4, there is a transition from $s \in A$ to $s' \notin A$ in TS where $s|_{B_1} = s_1$ and $s'|_{B_2} = s'_2$. This contradicts the assumption that A is an attractor of A. Next suppose A is not an attractor of TS_2. Then there is a transition in TS_2 from $s \in A$ to $s' \notin A$. But we have, by the construction of TS_2 (Definition 11), that this is also a transition in TS which again contradicts the assumption that A is an attractor of TS.

For the converse direction, suppose for contradiction that A is an attractor of TS_2 and A_1 is an attractor of TS_1 but A is not an attractor of TS. We must then have that there is a transition in TS from $s \in A$ to $s' \notin A$. If this transition is labelled with B_1 then we must have, by Lemma 5, that there is a transition in TS_1 from s_1

to s'_1. But since $s'_1 \notin A_1$ this contradicts the assumption that A_1 is an attractor of TS_1. Next, suppose that this transition is labelled with B_2^-. We must then have that $s_1 = s'_1 \in A_1$. Hence, by the construction of TS_2 (Definition 11), it must be the case that $s' \in TS_2$ and this transition from s to s' is also present in TS_2. But this contradicts the assumption that A is an attractor of TS_2. $\square$

Now assume BN has k blocks that are topologically sorted as $\{B_1, B_2, \ldots, B_k\}$. For every i such that $1 \leq i \leq k$, $(\bigcup_{j \leq i} B_j)$ is an elementary block of BN and we denote its transition system by $\overline{\mathsf{TS}}_i$.

Theorem 2 (preservation of attractors). *Suppose for every attractor A of TS and for every $i : 1 \leq i < k$, if B_{i+1} is non-elementary then TS_{i+1} is realised by $bas^S_{TS}(\otimes_{j \in I} A_j)$, its basin w.r.t. the TS for $(\bigcup_{j \in I} B_j)$, where I is the set of indices of the basic blocks in $\mathsf{ac}(B_{i+1})^-$. We then have, for every $i : 1 \leq i < k$, A_{i+1} is an attractor of TS_{i+1}, $(\otimes_{j \in I} A_j \otimes A_{i+1})$ is an attractor of the TS for the elementary block $(\bigcup_{j \in I} B_j \cup B_{i+1})$, $(\otimes_{j=1}^{i+1} A_j)$ is an attractor of $\overline{\mathsf{TS}}_{i+1}$ and A is an attractor of TS_k.*

Proof. The proof is by induction on i. The base case is when $i = 2$ and BN has two blocks B_1 and B_2. If B_1 and B_2 are both elementary then the result follows from Lemma 7. If B_1 is elementary and is the parent of B_2 then the result follows from Lemma 8.

For the inductive case suppose the result holds for some i where $2 \leq i < k$. Now both $(\bigcup_{j \in I} B_j)$, where I is the set of indices of the basic blocks in $\mathsf{ac}(B_{i+1})^-$, and $(\bigcup_{j \leq i} B_j)$ are elementary. Now, if B_{i+1} is elementary then the result follows from Lemma 7. If B_{i+1} is non-elementary then $(\bigcup_{j \in I} B_j)$ is the parent of B_{i+1} and the result follows from Lemma 8. $\square$

Next, let us come back to the case where BN has two blocks B_1 and B_2.

Lemma 9. *Suppose $B_1 \cap B_2 = \emptyset$ and both B_1 and B_2 are elementary blocks of* BN. *Let A, A_1 and A_2 be attractors of TS, TS_1 and TS_2 respectively where $A = A_1 \otimes A_2$. Then $bas^S_{TS}(A) = bas^S_{TS}(A_1) \otimes bas^S_{TS}(A_2)$.*

Proof. Follows easily from Lemma 6. $\square$

Lemma 10. *Let A, A_1 and A_2 be the attractors of TS, TS_1 and TS_2 respectively where B_1 and B_1 are elementary and non-elementary blocks respectively of* BN *with B_1 being the parent of B_2 and TS_2 being realised by $bas^S_{TS}(A_1)$ and $A = A_2$. Then $bas^S_{TS}(A_1) \otimes bas^S_{TS}(A_2) = bas^S_{TS}(A_2) = bas^S_{TS}(A)$.*

Proof. Since TS_2 is realised by $bas^S_{TS}(A_1)$, by its construction (Definition 11) we have, for every state $s \in TS_2$, $s_1 \in bas^S_{TS}(A_1)$. Hence $bas^S_{TS}(A_1) \otimes bas^S_{TS}(A_2) = bas^S_{TS}(A_2)$.

We next show that $bas^S_{TS}(A_2) = bas^S_{TS}(A)$. Suppose $s \in bas^S_{TS}(A_2)$. To show that $s \in bas^S_{TS}(A)$, it is enough to show that:
(i) There is a path from s to some $s^A \in A$ in TS and
(ii) There is no path from s to $s^{A'} \in A'$ for some attractor $A' \neq A$ of TS.

(i) Since $s \in bas^S_{TS}(A_2)$, and $A_2 = A$, there is a path ρ from s to $s^A \in A$ in TS_2. It is easy to see from the construction of TS_2 (Definition 11) that ρ is also a path in TS from s to s^A.

(ii) Suppose for contradiction that there is a path ρ' in TS from s to $s^{A'} \in A'$ for some attractor $A' \neq A$ of TS. Since $A' \neq A$ we must have that either (a) $A_1 \neq A'_1$ or (b) $A_1 = A'_1$ but $A_2 \neq A'_2$.

(a) In this case, by Lemma 5, there must be a path from s_1 to $s^{A'_1} \in A'_1$ which is a contradiction to the fact that $s_1 \in bas^S_{TS}(A_1)$.

(b) We have by Theorem 2 that $A'_2 = A'$. Once again from the construction of TS_2 (Definition 11) it is easy to see that ρ' is also a path in TS_2 from s to $s^{A'} \in A'$. But this contradicts the fact that $s \in bas^S_{TS}(A_2)$.

For the converse direction suppose that $s \in bas^S_{TS}(A)$. To show that $s \in bas^S_{TS}(A_2)$, it is enough to show that:
(iii) There is a path from s to some $s^{A_2} \in A_2$ and
(iv) There is no path from s to $s^{A'_2} \in A'_2$ for some attractor $A'_2 \neq A_2$ of TS_2.

(iii) Since $s \in bas^S_{TS}(A)$, there is a path ρ in TS from s to some $s^A \in A$. By the fact that $A_2 = A$ and by the construction of TS_2 (Definition 11) it is clear that ρ is also a path in TS_2 from s to $s^A \in A_2$.

(iv) Suppose for contradiction that there is a path ρ' in TS_2 from s to $s^{A'_2} \in A'_2$ for some attractor $A'_2 \neq A_2$ of TS_2. By Theorem 2, A'_2 is equal to an attractor A' of TS and $A' \neq A$. It is then easy to see again from the construction of TS_2 (Definition 11) that ρ' is also a path in TS from s to $s^{A'} \in A'$. But this contradicts the assumption that $s \in bas^S_{TS}(A)$. □

Let us, for the final time, come back to the case where BN has $k > 2$ blocks and these blocks are topologically sorted as $\{B_1, B_2, \ldots, B_k\}$. Let i range over $\{1, 2, \ldots, k\}$. By the theorem on attractor preservation, Theorem 2, we have that $(\otimes_{j \leq i} A_j)$ is an attractor of $\overline{\mathsf{TS}}_i$.

Lemma 11. *Suppose for every attractor A of TS and for every $i : 1 \leq i < k$, if B_{i+1} is non-elementary then TS_{i+1} is realised by $bas^S_{TS}(\otimes_{j \in I} A_j)$, its basin with respect to the TS for $(\bigcup_{j \in I} B_j)$, where I is the set of indices of the basic blocks in $\mathsf{ac}(B_{i+1})^-$ [where $(\otimes_{j \in I} A_j)$, by Theorem 2, is an attractor of the TS for $(\bigcup_{j \in I} B_j)$]. Then for every i, $(\otimes_{j \leq i} bas^S_{TS}(A_j)) = bas^S_{TS}(\otimes_{j \leq i} A_i)$ where $bas^S_{TS}(\otimes_{j \leq i} A_j)$ is the basin of attraction of $(\otimes_{j \leq i} A_j)$ with respect to transition system $\overline{\mathsf{TS}}_i$ of $(\bigcup_{j \leq i} B_j)$.*

Proof. The proof is by induction on i. The base case is when $i = 2$. Then either B_1 and B_2 are both elementary and disjoint in which case the proof follows from Lemma 9. Or, B_1 is elementary and B_2 is non-elementary and B_1 is the parent block of B_2. In this case the proof follows from Lemma 10.

For the inductive case, suppose that the conclusion of the theorem holds for some $i : 2 \leq i < k$. Now, consider $(\otimes_{j \leq (i+1)} bas^S_{TS}(A_j))$. By the induction hypothesis, we have that $(\otimes_{j \leq i} bas^S_{TS}(A_j)) = bas^S_{TS}(\otimes_{j \leq i} A_j)$ where $(\otimes_{j \leq i} A_j)$ is an attractor of the transition system $\overline{\mathsf{TS}}_i$ of the elementary block $(\bigcup_{j \leq i} B_j)$ and $bas^S_{TS}(\otimes_{j \leq i} A_j)$ is its

Algorithm 2 Decomposition-based computation of the strong basin

```
1: procedure COMP_STRONG_BASIN_DECOMP($G_{\mathsf{BN}}$,$F$,$A$)
2:     $\mathcal{B}$ := FORM_BLOCK($G_{\mathsf{BN}}$);
3:     $\mathcal{B}$ := TOP_SORT($\mathcal{B}$);
4:     $k$ := size of $\mathcal{B}$; $\mathsf{SB} = \phi$; $\mathsf{SB}_i = \emptyset$;        // for all i
5:     for $i = 1$ to $k$ do
6:         $A_i$ :=DECOMPOSE($A, B_i$);    // decompose the target attractor into block $B_i$
7:     end for
8:     for $i := 1$ to $k$ do
9:         if $B_i$ is an elementary block then
10:            $TS_i$ := transition system of $B_i$;
11:            $\mathsf{SB}_i$ :=COMP_STRONG_BASIN($F|_{\overline{B_i}}, A_i$);
12:        else
13:            $TS_i$ := transition system of $B_i$ based on the basin of $(\otimes_{j<i}A_j)$ in $TS_{i-1}$;
14:            $\mathsf{SB}_i$ :=COMP_STRONG_BASIN($F|_{\overline{B_i}}, A_i$);
15:        end if
16:        $\mathsf{SB}$ =CROSS ($\mathsf{SB}, \mathsf{SB}_i$);
17:    end for
18:    return $\mathsf{SB}$
19: end procedure
```

basin. Now, either B_{i+1} is elementary in which case we use Lemma 9 or B_{i+1} is non-elementary and $(\bigcup_{j\in I} B_j)$ is its parent in which case we use Lemma 10.

In either case, we have $(\otimes_{j\leq(i+1)}bas^S_{TS}(A_j)) = bas^S_{TS}(\otimes_{j\leq(i+1)}A_j)$, where $bas^S_{TS}(\otimes_{j\leq(i+1)}A_j)$ is the strong basin of attraction of the attractor $(\otimes_{j\leq(i+1)}A_j)$ of $\overline{\mathsf{TS}}_{i+1}$. □

Theorem 3 (preservation of strong basins)**.** *Given the hypothesis and the notations of Lemma 11, we have* $(\otimes_{i\leq k}bas^S_{TS}(A_i)) = bas^S_{TS}(A)$ *where* $bas^S_{TS}(A)$ *is the strong basin of the attractor* $A = (A_1 \otimes A_2 \otimes \ldots \otimes A_k)$ *of TS.*

Proof. Follows directly by setting $i = k$ in Lemma 11. □

Equipped with the results in Theorems 2 and 3, we can describe our procedure for computing the strong basin of the attractor based on the decomposition of the network. Towards that, Theorem 3 tells us that in order to compute $bas^S_{TS}(A)$ it is sufficient to compute the local strong basin of the projection of A to each block B_i (which by Theorem 2 is an attractor of B_i) and finally merge these local strong basins using the cross operation.

Algorithm 2 implements this idea in pseudocode. It takes as inputs the dependency graph G_{BN}, the update functions F, and the attractor A and returns the strong basin of A. Line 2 decomposes G_{BN} into blocks $\mathcal{B}$ (resulting in k blocks) using the procedure FORM_BLOCK from [MPQY19] and line 3 topologically sorts the blocks by constructing the block graph $\mathcal{G}_{\mathcal{B}}$. Lines 5-7 decompose the attractor A into its projection to the blocks. Lines 8-17 then cycle through the blocks of $\mathcal{B}$ in topological order and for each block B_i: if B_i is elementary then it constructs the transition system TS_i independently or, if B_i is non-elementary it constructs TS_i realised by the strong basin of $(A_1 \otimes A_2 \otimes \ldots \otimes A_{i-1})$ which by Theorem 2 is an attractor of TS_{i-1},

Network	# nodes	# SCCs	Time (seconds)		Speedups
			Global	Decomposition	
yeast	10	3	0.003	0.002	1.5
myeloid	11	2	0.003	0.003	1.0
cardiac	15	13	0.021	0.007	3.0
ERBB	20	14	0.004	0.004	1.0
HSPC-MSC	26	5	0.009	0.006	1.5
tumour	32	13	0.128	0.059	2.2
hematopoiesis	33	11	0.480	0.168	2.9
PC12	33	19	0.018	0.008	2.3
bladder	35	10	0.056	0.031	1.8
psc-bFA	36	10	22.684	10.067	2.3
co-infection	52	7	43.899	19.435	2.3
MAPK	53	18	3.167	1.729	1.8
CREB	64	39	0.181	0.621	0.3
HGF	66	26	24.290	1.751	13.9
bortezomib	67	28	18.468	6.662	2.8
T-diff	68	34	0.349	0.251	1.4
HIV1	136	62	264.778	47.777	5.5
CD4+	188	86	429.975	116.619	3.7
pathway	321	302	*	32.674	*

Table 4.1: Computational time of the global and decomposition-based methods for the computation of strong basins. Symbol '*' means no results were returned.

the transition system for the elementary (non-basic) block $\overline{B}_{i-1}$. Thus at every iteration i of the for-loop the invariant that A_i is an attractor of TS_i is maintained. The procedure Comp_Strong_Basin (lines 11,14), described in Algorithm 1, computes the strong basin of A_i with respect to TS_i. Line 16 extends the global strong basin SB computed so far by crossing it with the local strong basin computed at each step. At the end of the for-loop, SB will thus be equal to the global strong basin (by Theorem 3). It then easily follows that

Proposition 1. *Algorithm 2 correctly computes the strong basin of the attractor A.*

4.4 Evaluation

In this chapter, we have developed the global method and the decomposition-based method for the computation of strong basins. Both methods compute the exact basins of a given attractor. In this section, we evaluate the efficiency of the two methods on a number of biological networks introduced in Section 2.4. All the experiments are performed on a high-performance computing (HPC) platform, which contains CPUs of Intel Xeon Gold 6132 @2.6 GHz.

Table 4.1 gives the evaluation results of the two methods for the computation of strong basins. The second and the third columns list the number of nodes and the number of SCCs (or components) contained in each network. The fourth and the fifth columns give the execution time for computing one of the basins. The last column shows the speedups gained by the decomposition-based method, computed with the formula $speedup = T_{global} / T_{decomp}$.

We can see that with the fixed point computation, both methods are quite efficient and scale well for large networks. Owing to the "divide and conquer" strategy, for most of the networks, the decomposition-based computation of strong basins achieves improvements in efficiency compared to the global method, which treats the entire network in one-go.

The performance of the two methods on the myeloid differentiation network and the ERBB receptor-regulated G1/S transition protein network are similar, probably due to that the sizes of the two networks are small. For the CREB network, the decomposition-based method underperforms the global method in terms of the efficiency, because 38 out of 39 SCCs of this network are single-node SCCs. Even though the sizes of the blocks are small, to traverse the blocks one by one can be more time-consuming than the global method. This problem can be solved by replacing the SCC decomposition with a better decomposition method, which reduces the number of blocks and balances the sizes of blocks. In Chapter 3, we have proposed a near-optimal decomposition to improve the efficiency of the decomposition-based attractor detection. However, the formation of the block dynamics for the methods of attractor detection and basin computation is different, therefore, the new decomposition proposed in Chapter 3 has to be refined before being integrated with the decomposition-based method for basin computation.

Another point worth mentioning is that for networks with more than 100 nodes, even if the network has a large number of SCCs, the decomposition-based method can still gain efficiency compared to the global method. For instance, for the pathway network with 321 nodes, symbol '*' in Table 4.1 indicates that the global method failed to return any results as the computation was terminated in the attractor detection phase by segmentation fault. The reason is that the state space of the network grows exponentially in the size of the network. With the decomposition-based method, this network is decomposed into 302 SCCs, out of which 298 are single-node SCCs. Even so, the decomposition-based method can finish the computation very efficiently. Thus, for large-scale networks, it is inevitable to use the decomposition-based method for the computation of the strong basins.

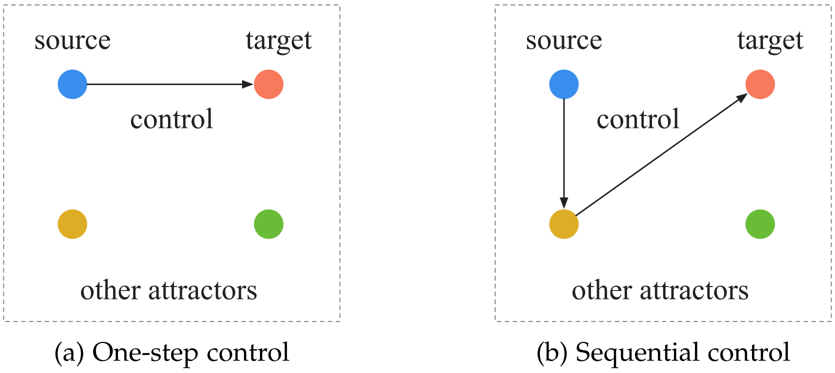
Chapter 5

Source-target Control

5.1 Introduction

Direct cell reprogramming amounts to being able to drive the dynamics of a Boolean network (from the source state) to the 'desirable' target attractor by controlling or reprogramming the nodes of the Boolean network. As discussed in Section 1.4, the application of existing approaches is restricted due to various reasons. Their limitations motivate us to develop new methods towards the control of Boolean networks. In this chapter, we study the problem of *source-target control* of asynchronous Boolean networks: to identify a subset C of nodes of a given BN, such that by perturbing the nodes in C, the dynamics of BN can be driven from the source state to the desired target attractor.

Due to the intrinsic diversity and complexity of biological systems, it is less likely to find one control method that can perfectly suit all cases. Thus, it is of great importance to explore different strategies to provide a number of cautiously selected candidates for further experimental validation. Based on the number of control steps, we can have *one-step control* and *sequential control*. As shown in Figure 5.1, one-step control applies all the perturbations simultaneously for one time; sequential control identifies a sequence of perturbations to be applied at different time points. Figure 5.1(b) illustrates a sequential control realised in two steps. In the first step, a set of perturbations are applied to the source state, which drives the network to an intermediate state. The intermediate state can be either a transient



(a) One-step control (b) Sequential control

Figure 5.1: Two source-target control strategies. The blue and red nodes are the source and target attractors, respectively.

state or a state in an attractor. Once the network reaches the intermediate state, the other set of perturbations are applied to guide the network towards the target attractor. By taking advantage of the natural dynamics of the network, sequential control can provide alternative control paths to one-step control, requiring considerably fewer perturbations [MHP17]. However, in order to apply the perturbations at the correct time, *general sequential control*, where any state can be intermediate states, requires *complete observability* of the network (i.e., the state of the network is known at any discrete time), which is rarely feasible in practice. To make the control more practical, we are particularly interested in the sequential control that only adopts attractors as intermediates, called *attractor-based sequential control*. Since the attractors can be observed experimentally, the attractor-based sequential control only requires *partial observability* of the network. In practice, biologists can determine how long it takes for the network to stabilise in an intermediate attractor, i.e. the timing to apply the next set of perturbations, based on empirical experiences. In that way, if the intermediate attractors are singleton attractors, partial observability is not required. But, if the intermediate attractors are cyclic attractors, partial observability of the network might still be required.

Rapid development of biomolecular techniques enables us to perturb expressions of nodes for different periods (instantaneously, temporarily or permanently) in both directions: from 'expressed' to 'not expressed' and/or from 'not expressed' to 'expressed' [GTZ$^+$18]. Thus, we can have *instantaneous control*, *temporary control* and *permanent control*. Instantaneous control applies perturbations instantaneously; temporary control applies perturbations for sufficient time and then releases them to retrieve the original dynamics; permanent control applies perturbations for all the following time steps. The application of control C reshapes BN to a new one, where the Boolean functions of the nodes in C are fixed to either '1' or '0'. Permanent control leads to a permanent shift of the dynamics, thus, it should be applied cautiously to reduce potential risks. By applying one-step control and attractor-based sequential control with instantaneous, temporary and permanent perturbations, we can have six different control strategies: *one-step instantaneous, temporary and permanent control (OI, OT and OP)* and *attractor-based instantaneous, temporary and permanent control (ASI, AST and ASP)*.

In this chapter, we develop efficient methods to solve the six source-target control problems (OI, OT, OP, ASI, AST and ASP). All the methods are based on the computation of weak and strong basins of attractors. Since biological experiments are usually laborious and very expensive, to reduce the experimental costs, we aim to compute the minimal subset of the nodes that can realise the control. With the minimality constraint, it is known that the problem of driving the Boolean network from a source attractor to a target attractor (the control problem) is computationally difficult [MHP16, MHP17]. To cope with the state-space explosion problem, the decomposition-based method for the computation of the strong basin of an attractor is used as the foundation for our control methods.

A key factor to make the control realistic is to employ physically admissible and experimentally feasible perturbations [CKM13]. We integrate our methods with constraints on perturbations, such that undesired perturbations are avoided during computation. For attractor-based sequential control, besides undesired perturbations, undesired attractors, such as the attractors corresponding to diseased states, should be avoided as intermediate states. The algorithms we will describe later provide options to avoid user-specified perturbations and/or attractors.

The rest of this chapter is organised as follows:

- In Section 5.2, we give some notions on Boolean networks under control and formally define the control problems of OI, OT, OP, ASI, AST and ASP.
- In Sections 5.3, 5.4 and 5.5, we introduce the methods for the minimal OI, OT and OP control based on the computation of basins.
- In Sections 5.6, 5.7 and 5.8, we extend the one-step control methods to solve the problems of ASI, AST and ASP control.
- We evaluate the performance of the six source-target control methods on a number of real-life biological networks in Section 5.9.

5.2 The source-target control problems

5.2.1 Boolean networks under control

In this section, we define several notions on Boolean networks under control, denoted $\mathsf{BN}|_{\mathsf{C}}$. We first give the formal definition of control C as follows.

Definition 15 (Control). *A control* C *is a tuple* $(\mathbb{0}, \mathbb{1})$, *where* $\mathbb{0}, \mathbb{1} \subseteq \{1, 2, \ldots, n\}$ *and* $\mathbb{0}$ *and* $\mathbb{1}$ *are mutually disjoint (possibly empty) sets of indices of nodes of a Boolean network* BN. *The size of the control* C *is defined as* $|\mathsf{C}| = |\mathbb{0}| + |\mathbb{1}|$. *The control* C *has an associated source state* $s \in S$. *The application of* C *to* s, *denoted as* $\mathsf{C}(s)$, *is defined as a state* $s' \in S$, *such that* $s'[i] = 0 = 1 - s[i]$ *for* $i \in \mathbb{0}$ *and* $s'[i] = 1 = 1 - s[i]$ *for* $i \in \mathbb{1}$. *State* s' *is called the intermediate state with respect to* C.

Intuitively, sets $\mathbb{0}$ and $\mathbb{1}$ represent the indices of variables of BN whose values are held fixed to 0 and 1 respectively under the control C. The union of the two sets, $\mathbb{0} \cup \mathbb{1}$, is the set of indices in which s and s' differ, out of which $\mathbb{0}$ and $\mathbb{1}$ are the indices of nodes which have a value of 0 and 1 in s', respectively. Let $\mathsf{C}' = (\mathbb{0}', \mathbb{1}')$ be a control of BN. We define $\mathsf{C} \setminus \mathsf{C}' = (\mathbb{0} \setminus \mathbb{0}', \mathbb{1} \setminus \mathbb{1}')$.

The application of a control C to $\mathsf{BN} = (X, F)$ has the effect of reducing the state space of BN to those which have the values of the variables in $\mathbb{0}$ and $\mathbb{1}$ set respectively to 0 and 1 and modifying the Boolean functions accordingly. This results in a new Boolean network derived from BN defined as follows.

Definition 16 (Boolean networks under control). *Let* $\mathsf{C} = (\mathbb{0}, \mathbb{1})$ *be a control and* $\mathsf{BN} = (X, F)$ *be a Boolean network. The Boolean network* BN *under control* C, *denoted* $\mathsf{BN}|_{\mathsf{C}}$, *is defined as a tuple* $\mathsf{BN}|_{\mathsf{C}} = (\hat{X}, \hat{F})$, *where* $\hat{X} = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n\}$ *and* $\hat{F} = \{\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_n\}$, *such that for all* $i \in [n]$:
(1) $\hat{x}_i = 0$ *if* $i \in \mathbb{0}$, $\hat{x}_i = 1$ *if* $i \in \mathbb{1}$, *and* $\hat{x}_i = x_i$ *otherwise;*
(2) $\hat{f}_i = 0$ *if* $i \in \mathbb{0}$, $\hat{f}_i = 1$ *if* $i \in \mathbb{1}$, *and* $\hat{f}_i = f_i$ *otherwise.*

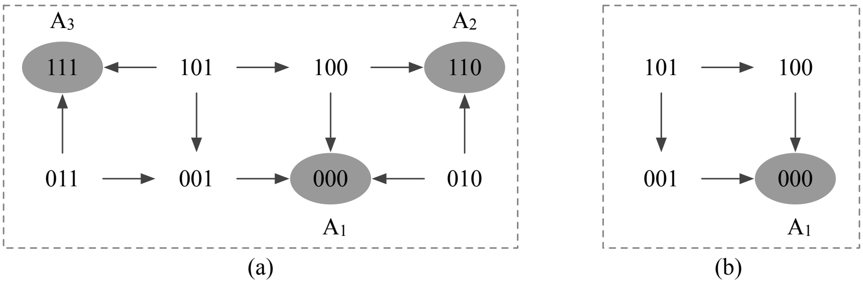


Figure 5.2: (a) Transition system TS and (b) Transition system under control $TS|_{\mathsf{C}}$ of Example 7, where $\mathsf{C} = \{x_2 = 0\}$.

The state space of $\mathsf{BN}|_{\mathsf{C}}$, denoted $S|_{\mathsf{C}}$, is derived by fixing the values of the variables in C to their respective values and is defined as $S|_{\mathsf{C}} = \{s \in S \mid s[i] = 1 \text{ if } i \in \mathbb{1} \text{ and } s[j] = 0 \text{ if } j \in \mathbb{0}\}$. It is obvious that $S|_{\mathsf{C}} \subseteq S$. For any subset S' of S, we let $S'|_{\mathsf{C}} = S' \cap S|_{\mathsf{C}}$.

The asynchronous dynamics of $\mathsf{BN}|_{\mathsf{C}} = (\hat{X}, \hat{F})$ on the state space of $S|_{\mathsf{C}}$, denoted $\xi_{\mathsf{BN}|_{\mathsf{C}}}$, is defined similarly to the asynchronous dynamics of BN (Definition 2).

Definition 17 (Asynchronous dynamics under control). *Suppose $s_0 \in S|_{\mathsf{C}}$ is a state of $\mathsf{BN}|_{\mathsf{C}}$. The asynchronous evolution of $\mathsf{BN}|_{\mathsf{C}}$ starting from s_0 is a function $\xi_{\mathsf{BN}|_{\mathsf{C}}} : \mathbb{N} \to \wp(S|_{\mathsf{C}})$ such that $\xi_{\mathsf{BN}|_{\mathsf{C}}}(0) = \{s_0\}$ and for every $j \geq 0$, if $s \in \xi_{\mathsf{BN}|_{\mathsf{C}}}(j)$ then $s' \in \xi_{\mathsf{BN}|_{\mathsf{C}}}(j+1)$ is a possible* next state *of s iff either $hd(s,s') = 1$ and $s'[i] = \hat{f}_i(s) = 1 - s[i]$ where $i = \arg(hd(s,s'))$ or $hd(s,s') = 0$ and there exists i such that $s'[i] = \hat{f}_i(s) = s[i]$.*

The dynamics of $\mathsf{BN}|_{\mathsf{C}}$ can also be defined as a transition system, which is defined similar to Definition 3, as follows:

Definition 18 (Transition system under control). *The transition system of $\mathsf{BN}|_{\mathsf{C}}$, denoted $TS_{\mathsf{BN}|_{\mathsf{C}}}$, is a tuple $(S|_{\mathsf{C}}, \to_{\mathsf{BN}|_{\mathsf{C}}})$ where the vertices are the set of states $S|_{\mathsf{C}}$ and for any two states s and s' there is a directed edge from s to s', denoted $s \to_{\mathsf{BN}|_{\mathsf{C}}} s'$ iff s' is a possible next state of s according to the asynchronous evolution function $\xi_{\mathsf{BN}|_{\mathsf{C}}}$ of $\mathsf{BN}|_{\mathsf{C}}$.*

The notions on TS, including attractors (Definition 5) and basins (Definition 6), can be extended to $TS|_{\mathsf{C}}$.

Example 7. *For the Boolean network BN defined in Example 1, given a control $\mathsf{C} = (\mathbb{0}, \mathbb{1})$, where $\mathbb{0} = \{2\}, \mathbb{1} = \emptyset$ (i.e. $\{x_2 = 0\}$), the application of C reshapes the transition system TS of BN from Figure 5.2(a) to the transition system under control $TS|_{\mathsf{C}}$ of $\mathsf{BN}|_{\mathsf{C}}$ in Figure 5.2 (b). We can see that the control results in a new transition system, where only a subset of states and transitions are preserved. Therefore, the attractors of TS and $TS|_{\mathsf{C}}$ might differ. For this example, only attractor A_1 is preserved in $TS|_{\mathsf{C}}$ as shown in Figure 5.2 (b).*

5.2.2 The control problems

Let BN be a given Boolean network, S be the set of states of BN and $\mathcal{A}$ be the set of attractors of BN. A control C is a *source-target control* from a source state $s \in S$

to a target attractor $A_t \in \mathcal{A}$, if after the application of C to s, BN eventually reaches A_t. Source-target control can be achieved by applying the control (as defined in the previous section) to the network for different periods of time. We have: (a) *instantaneous control* – the control is applied instantaneously; (b) *temporary control* – the control is applied for a finite (possibly zero) number of steps and then released; and (c) *permanent control* – the control is applied for all the following time steps, i.e., the parameters are changed for all the following steps. The process of the application and release of control is formally defined as follows.

Definition 19 (Application and release of control). *Let $s \in S$ be a state of* BN *and let* $\mathsf{C} = (\mathbb{0}, \mathbb{1})$ *be a control. The instantaneous application of* C *to s results in a state s' such that $s'[i] = 1$ for all $i \in \mathbb{1}$, $s'[j] = 0$ for all $j \in \mathbb{0}$ and $s'[k] = s[k]$ otherwise. We will often denote this as $s \overset{\mathsf{C}}{\sim} s'$ and denote s' as $\mathsf{C}(s)$. The application of* C *to s for $t \geq 1$ time steps results in a sequence $s_0, s_1, s_2, \ldots, s_t$, where $s_0 = \mathsf{C}(s)$ and for every $k \in [t], s_k \in \xi_{\mathsf{BN}|_{\mathsf{C}}}(s_{k-1})$. When $t \to \infty$, we shall call it a permanent application of* C *to s or a permanent control of s.*

Suppose BN *under control* C *is in state $s \in S|_{\mathsf{C}}$ and has been evolving according to $\xi_{\mathsf{BN}|_{\mathsf{C}}}$. The release of control at s is performed instantaneously and it restores the dynamics to ξ_{BN}. We often denote it as $s \overset{\mathsf{C}^{-1}}{\sim} s \rightarrow_{\mathsf{BN}} \ldots$*

Thus, suppose BN starts evolving from an initial state $s_0 \in S$ and after t_1 steps a control C is applied to it. Suppose the control lasts for t_2 steps and then it is released and then BN goes back to evolving according to its original dynamics. This will result in a sequence that can be represented as:

$$\underbrace{s_0 \rightarrow_{\mathsf{BN}} s_1 \rightarrow_{\mathsf{BN}} \cdots \rightarrow_{\mathsf{BN}} s_{t_1}}_{t_1 \text{ steps}} \overset{\mathsf{C}}{\sim} \underbrace{s'_0 \rightarrow_{\mathsf{BN}|_{\mathsf{C}}} s'_1 \rightarrow_{\mathsf{BN}|_{\mathsf{C}}} \cdots \rightarrow_{\mathsf{BN}|_{\mathsf{C}}} s'_{t_2}}_{t_2 \text{ steps under control } \mathsf{C}} \overset{\mathsf{C}^{-1}}{\sim} s''_0 \rightarrow_{\mathsf{BN}} s''_1 \rightarrow_{\mathsf{BN}} \cdots$$

Intuitively, on the application of control C for t_2 steps, the behaviour of BN is captured by $TS_{\mathsf{BN}|_{\mathsf{C}}}$ for t_2 time steps. After that, when C is released, the behaviour goes back to TS_{BN}. The release of C does not change the value of any variable, thus $s''_0 = s'_{t_2}$.

Given a source attractor A_s and a target attractor A_t of TS, to drive the network from A_s to A_t in one step is called *one-step source-target control*, formally defined as:

Definition 20 (One-step source-target control). *Compute a control* C, *such that the application of* C *to a state s, $s \in A_s$, drives the network from s to A_t.*

When the control C is the instantaneous, temporary or permanent control, we call it *one-step instantaneous, temporary or permanent control (OI, OT, or OP)*. Based on the definition of fairness (Definition 4), we formally define the one-step source-target control problems as follows.

Definition 21 (One-step source-target control problems).

1. **One-step instantaneous source-target control (OI):** *find a control* $\mathsf{C} = (\mathbb{0}, \mathbb{1})$ *such that the dynamics of* BN *always eventually reaches A_t on the instantaneous application of* C *to $s, s \in A_s$.*

2. **One-step temporary source-target control (OT):** *find a control* $\mathsf{C} = (\mathbb{0}, \mathbb{1})$ *such that there exists a* $t_0 \geq 0$ *such that for all* $t \geq t_0$*, the dynamics of* BN *always eventually reaches* A_t *on the application of* C *to* $s, s \in A_s$ *for* t *steps.*

3. **One-step permanent source-target control (OP):** *find a control* $\mathsf{C} = (\mathbb{0}, \mathbb{1})$ *such that the dynamics of* BN *always eventually reaches* A *on the permanent application of* C *to* $s, s \in A_s$*. (Here we assume implicitly that* A_t *is also an attractor of the transition system under control* $TS_{\mathsf{BN}|_{\mathsf{C}}}$*.)*

Given a fair path $\rho = s_0 \to s_1, \to \ldots$, we say that ρ *eventually* reaches A_t if there exists a $k \geq 0$ such that $s_k \in A_t$. To minimise the experimental costs, we are interested in the minimal solution $\mathsf{C}_{\min}^{A_s \to A_t}$, where $\mathsf{C}_{\min}^{A_s \to A_t}$ is the minimal such subset of $\{1, 2, \ldots, n\}$. OI can be considered as a special case of OT when $t \to 0$. Note that the constraint of minimality in the above definition makes the problems non-trivial. Otherwise, one can simply choose a control C such that $\mathsf{C}(s) \in A_t$ in each case. We first observe that the minimal OI control problem is computationally difficult (PSPACE-hard), the minimal temporary and permanent control problems are also computationally difficult (at least PSPACE-hard). Thus, efficient algorithms to solve these problems are highly unlikely. In this chapter, we shall develop algorithms to solve these problems based on the computation of the weak and strong basins of the target attractor A_t.

Besides the one-step source-target control, we can identify a sequence of perturbations to drive the network from A_s to A_t in a stepwise manner. We call it *sequential control*. We concentrate on the sequential control through other attractors, called *attractor-based sequential control*, defined below.

Definition 22 (Attractor-based sequential control)**.** *Find a sequence of attractors of TS, i.e.* $\Phi = \langle A_1, A_2, \ldots, A_m \rangle$*, where* $A_1 = A_s, A_m = A_t$*,* $A_i \neq A_j$ *for any* $i, j \in [1, m]$ *and* $2 \leq m \leq |\mathcal{A}|$*, such that after the application of a sequence of minimal one-step controls* $\langle \mathsf{C}_{\min}^{A_1 \to A_2}, \mathsf{C}_{\min}^{A_2 \to A_3}, \ldots, \mathsf{C}_{\min}^{A_{m-1} \to A_m} \rangle$*, the network always eventually reaches* A_m*, i.e.* A_t*. We call it an attractor-based sequential control path, denoted as*

$$\Omega : A_1 \xrightarrow{\mathsf{C}_{\min}^{A_1 \to A_2}} A_2 \xrightarrow{\mathsf{C}_{\min}^{A_2 \to A_3}} A_3 \xrightarrow{\cdots} \ldots \xrightarrow{\mathsf{C}_{\min}^{A_{m-1} \to A_m}} A_m$$

$(|\mathsf{C}_{\min}^{A_1 \to A_2}| + |\mathsf{C}_{\min}^{A_2 \to A_3}| + \ldots + |\mathsf{C}_{\min}^{A_{m-1} \to A_m}|)$ *is the total number of perturbations.*

Similarly, when the control $\mathsf{C}_{\min}^{A_i \to A_j}$, $A_i, A_j \in \Phi$ is the instantaneous, temporary or permanent control, we call it *attractor-based sequential instantaneous, temporary or permanent control (ASI, AST or ASP)*, respectively.

In the following sections, we shall develop several algorithms to solve the minimal OI, OT and OP control problems. Based on these algorithms, we further develop algorithms for ASI, AST and ASP control. Afterwards, we introduce how to modify these algorithms to integrate constraints on perturbations and intermediate attractors (for attractor-based sequential control).

5.3 One-step instantaneous control

In this section, we develop the method to solve the minimal OI control described in Definition 21. The following lemma will be crucial for the control methods we shall develop in this and the following sections.

Lemma 12. *Let $s \in S$ and $S' \subseteq S$. Every path $\rho \in P_\infty(s)$*

- *possibly eventually reaches S' if and only if $s \in bas^W_{TS}(S')$,*
- *always eventually reaches S' if and only if $s \in bas^S_{TS}(S')$.*

Indeed, if the Boolean network is in some state $s \in bas^S_{TS}(A_t)$ in some time step t, that is if $\xi(t) = s$ then by the definition of $bas^S_{TS}(A_t)$ it will eventually and surely reach a state $s' \in A_t$. That is, there exists a time step $t' > t$ such that $\xi(t') = s'$, $s' \in A_t$. In Section 2.1.1, we have defined $\arg(hd(S', S''))$ as the set of subsets of $\{1, 2, \ldots, n\}$ such that $I \in \arg(hd(S', S''))$ if and only if I is a set of indices of the variables that realise this Hamming distance. Hence given a source attractor A_s and a target attractor A_t, the union of a pair of $\mathbb{0}$ and $\mathbb{1}$ where $\mathsf{C}^{A_s \to A_t}_{\min} = (\mathbb{0}, \mathbb{1})$, can easily be seen to be an element in $\arg(hd(A_s, bas^S_{TS}(A_t)))$. That is

Theorem 4. *A control $\mathsf{C} = (\mathbb{0}, \mathbb{1})$ from A_s to A_t is a minimal OI control if and only if $\mathsf{C}(s) \in bas^S_{TS}(A_t)$ and $\mathsf{C} \in \arg(hd(A_s, bas^S_{TS}(A)))$, where s is the state in A_s that realises the minimal Hamming distance from A_s to $bas^S_{TS}(A_t)$.*

Proof. If $\mathsf{C}(s) \notin bas^S_{TS}(A_t)$ then either (a) $\mathsf{C}(s) \notin bas^W_{TS}(A_t)$ or (b) $\mathsf{C}(s) \in bas^W_{TS}(A_t)$. If (a) holds, then there is no path from $\mathsf{C}(s)$ to A_t and if (b) holds, then there is a path from $\mathsf{C}(s)$ to some other attractor $A \neq A_t$. In either case BN is not guaranteed to reach a state in A_t after the control C is applied to s. And, if $(\mathbb{0} \cup \mathbb{1}) \notin \arg(hd(A_s, bas^S_{TS}(A_t)))$, then C cannot be minimal (by definition of Hamming distance), and conversely. Since $\mathsf{C} \in \arg(hd(A_s, bas^S_{TS}(A)))$, the associated source state s of C is the state in A_s that realises the minimal Hamming distance from A_s to $bas^S_{TS}(A_t)$. □

Thus, solving the minimal OI control problem efficiently boils down to how efficiently we can compute the strong basin of the target attractor. In Chapter 4, we developed a global method, COMP_STRONG_BASIN, and a decomposition-method, COMP_STRONG_BASIN_DECOMP, for the computation of strong basins and proved that the decomposition method is generally more efficient than the global method.

In this section, we shall design an algorithm, Algorithm 3, for solving the minimal OI control problem based on the decomposition-based computation of the strong basin of the target attractor A_t. It first computes the strong basin SB of the target attractor A_t using procedure COMP_STRONG_BASIN_DECOMP and then computes a minimal OI control set $\mathsf{C}_{\min}$ using procedure COMP_CONTROL_SET. $\mathsf{C}_{\min}$ has an associated source state $s \in A_s$ and an intermediate state $s' = \mathsf{C}_{\min}(s)$, $s' \in \mathsf{SB}$. The pair of states s and s' realises the minimal Hamming distance between A_s and SB.

Our method for the minimal OI control, Algorithm 3, is generic, in that, we can plug into it any other algorithm for computing the strong basin of the target attractor and

Algorithm 3 Minimal OI control

```
procedure MIN_OI_CONTROL($F, A_s, A_t$)
    SB :=COMP_STRONG_BASIN_DECOMP($F, A_t$)
    ($\mathsf{C}_{\min}, s, s'$) :=COMP_CONTROL_SET($A_s$, SB) // $s \in A_s$ and $s' \in$ SB are the states that realise the minimal Hamming distance.
    return $\mathsf{C}_{\min}$
end procedure
procedure COMP_CONTROL_SET($A_s$, SB)
    $\mathsf{C} := (\mathbb{0}, \mathbb{1})$, where $\mathbb{0} := \emptyset$ and $\mathbb{1} := \emptyset$.
    $(s, s') :=$ MIN_HAMMING_DIST($A_s$, SB)    // compute the states that realise the minimal Hamming distance between $A_s$ and SB.
    $I := \arg(hd(s, s'))$    // compute indices of the nodes that have different values in $s$ and $s'$.
    for $i \in I$ do
        if $s'[i] = 0$ then
            Add $i$ to $\mathbb{0}$.
        else
            Add $i$ to $\mathbb{1}$.
        end if
    end for
    return $(\mathsf{C}, s, s')$
end procedure
```

it would still work. Its performance, however, directly depends on the performance of the particular algorithm used to compute this basin.

Example 8. *We use the Boolean network given in Example 1 to illustrate the computation of minimal OI control with Algorithm 3. The transition system of Example 1 can be found in Figure 5.2 (a) in Section 5.2.1. Let* $A_3 = \{111\}$ *and* $A_1 = \{000\}$ *be the source and target attractors, respectively. To compute the minimal OI control from* A_3 *to* A_1*, we first use the procedure* COMP_STRONG_BASIN_DECOMP *to compute the strong basin of* A_1*, which is* $bas^S_{TS}(A_1) = \{000, 001\}$*. The next step is to find the pair of states* (s, s')*, where* $s \in A_3$ *and* $s' \in bas^S_{TS}(A_1)$*, that realises the minimal Hamming distance between* A_3 *and* $bas^S_{TS}(A_1)$*. For this case,* $s = (111)$ *and* $s' = (001)$*. By comparing the values of the nodes in* s *and* s'*, we know that the minimal OI control* $\mathsf{C}_{\min} = (\mathbb{0}, \mathbb{1})$*, where* $\mathbb{0} = \{1, 2\}$ *and* $\mathbb{1} = \emptyset$*. That is, by instantaneously flipping the values of the nodes* $\{x_1, x_2\}$ *from* 1 *to* 0 *in state* s*, the network is guided to the state* s'*, which is in* $bas^S_{TS}(A_1)$ *and has the minimal Hamming distance to* A_3*. From* s'*, the network will evolve spontaneously towards the target attractor* A_1*.*

5.4 One-step temporary control

In this section, we study the problem of the minimal OT control defined in Definition 21. According to Lemma 12, given a source state s and a target attractor A_t of TS of BN, if after the application of a control C to s, the resulting state $\mathsf{C}(s)$ lies in the strong basin of A_t with respect to the transition system under control, $TS|_{\mathsf{C}}$, then the dynamics will always eventually reach A_t. One needs to be careful though as the attractors and their structure in $TS|_{\mathsf{C}}$ might be different from TS. However, note that the application of C to TS does not create any additional edges except for self loops.

Lemma 13. *Let C be a control. If $s \rightarrow_{\mathsf{BN}|_{\mathsf{C}}} s'$ is in $TS|_{\mathsf{C}}$ then $s \neq s'$ implies $s \rightarrow_{\mathsf{BN}} s'$ is in TS.*

Proof. Suppose $\mathsf{C} = (\mathbb{0}, \mathbb{1})$ and $s \rightarrow_{\mathsf{BN}|_{\mathsf{C}}} s'$. Then since $s \neq s'$, by the definition of $TS|_{\mathsf{C}}$, $s, s' \in S|_{\mathsf{C}}$ and there exists $i \in [1, n]$ such that $s'[i] \neq s[i]$. Therefore $i \notin (\mathbb{0} \cup \mathbb{1})$ and $s' = f_i(s)$. Now since $s, s' \in S$, we have $s \rightarrow_{\mathsf{BN}} s'$ is a valid transition in TS. □

Next, we prove the following proposition with the help of Lemma 13.

Proposition 2. *Let A be an attractor of TS and C be any control.*

1. *For any state $s \in bas^S_{TS}(A)$, $reach_{TS}(s) \subseteq bas^S_{TS}(A)$.*
2. *For any state $s \in bas^S_{TS}(A)|_{\mathsf{C}}$, $reach_{TS|_{\mathsf{C}}}(s) \subseteq bas^S_{TS}(A)|_{\mathsf{C}}$.*

Proof. Assume for any state $s \in bas^S_{TS}(A)$, $reach_{TS}(s) \nsubseteq bas^S_{TS}(A)$. Then there must exist one path from s to a state $s', s' \notin bas^S_{TS}(A)$, i.e. $\rho = s \rightarrow s_1 \rightarrow \ldots \rightarrow s'$. Since $s' \notin bas^S_{TS}(A)$, there exists one path from s' to some other attractor $A', A' \neq A$, i.e. $\rho' = s' \rightarrow s'_1 \rightarrow \ldots \rightarrow A'$. Thus, there is one path from s to another attractor A', which contradicts with the definition of strong basin (Definition 6). Therefore, for any state $s \in bas^S_{TS}(A)$, $reach_{TS}(s) \subseteq bas^S_{TS}(A)$.

According to Lemma 13, the application of control C does not create any additional edges except for self loops. Thus, for any state $s \in (bas^S_{TS}(A)|_{\mathsf{C}})$, $reach_{TS|_{\mathsf{C}}}(s) \subseteq (bas^S_{TS}(A) \cap S_{\mathsf{C}})$, namely $reach_{TS|_{\mathsf{C}}}(s) \subseteq (bas^S_{TS}(A)|_{\mathsf{C}})$. □

Using Lemma 12 and Proposition 2, we can prove the following theorem.

Theorem 5. *Let $s \in S$ be a source state in A_s and let A_t be the target attractor of TS. A control C is a temporary control from s to A_t if and only if $bas^S_{TS}(A_t)|_{\mathsf{C}} \neq \emptyset$ and $\mathsf{C}(s) \in bas^S_{TS|_{\mathsf{C}}}(bas^S_{TS}(A_t)|_{\mathsf{C}})$.*

Proof. $bas^S_{TS}(A)|_{\mathsf{C}} = \emptyset$ implies $bas^S_{TS}(A) \cap S|_{\mathsf{C}} = \emptyset$ and such a control C is not well-defined. Hence, we assume that $bas^S_{TS}(A)|_{\mathsf{C}} \neq \emptyset$

Let s_0 denote $\mathsf{C}(s)$. If $s_0 \in bas^S_{TS|_{\mathsf{C}}}(bas^S_{TS}(A)|_{\mathsf{C}})$ then, by the definition of strong basin, for every fair path $\rho = s_0 \rightarrow_{\mathsf{BN}|_{\mathsf{C}}} s_1 \rightarrow_{\mathsf{BN}|_{\mathsf{C}}} \ldots$ there exists t such that $s_t \in bas^S_{TS}(A)|_{\mathsf{C}}$. By Proposition 2, $reach_{TS|_{\mathsf{C}}}(s_t) \subseteq bas^S_{TS}(A)|_{\mathsf{C}}$. Hence no matter for how many more steps t' C is held, $s_{t'} \in bas^S_{TS}(A)|_{\mathsf{C}}$. Thus when C is released after $(t + t')$ time steps, $s_{(t+t')} \in bas^S_{TS}(A)$ and by the definition of strong basin, the dynamics always eventually reaches A. Hence C is a temporary control.

Conversely, if $s_0 \notin bas^S_{TS|_{\mathsf{C}}}(bas^S_{TS}(A)|_{\mathsf{C}})$ then there is a path ρ in $TS|_{\mathsf{C}}$ from s_0 to some $s' \notin (bas^S_{TS}(A)|_{\mathsf{C}})$. Since $bas^S_{TS}(A)|_{\mathsf{C}} = bas^S_{TS}(A) \cap S|_{\mathsf{C}}$, and $s' \in S|_{\mathsf{C}}$ we must have $s' \notin bas^S_{TS}(A)$. Hence by the definition of strong basin, $reach_{TS}(s') \subseteq bas^S_{TS}(A)$ and so the dynamics of BN may not always eventually reach A from s'. So, C cannot be a valid temporary control. □

Algorithm 4 Minimal OT control

```
procedure MIN_OT_CONTROL($F, A_s, A_t$)    // $A_s$: the source; $A_t$: the target
    WB :=COMP_WEAK_BASIN($A_t, F$)    // the weak basin of $A_t$ in TS
    SB :=COMP_STRONG_BASIN_DECOMP($A_t, F$)    // the strong basin of $A_t$ in TS
    $\mathsf{C}_{\min}$ :=COMP_MIN_OT_CONTROL($F, A_s$, WB, SB)
    return $\mathsf{C}_{\min}$
end procedure
procedure COMP_MIN_OT_CONTROL($F, A_s$, WB, SB)
    isMin := false, $\mathsf{C}_{\min} = \emptyset$
    while isMin = false and WB $\neq \emptyset$ do
        ($\mathsf{C}, s, s'$) :=COMP_CONTROL_SET($A_s$, WB)    // $s \in A_s$ and $s' \in$ WB are the states that realise the shortest Hamming distance.
        isMin :=VERIFY_TEMPORARY_CONTROL($\mathsf{C}, F, s'$, SB)
        if isMin then
            $\mathsf{C}_{\min} := \mathsf{C}$
        else
            WB := WB $\setminus s'$
        end if
    end while
    return $\mathsf{C}_{\min}$
end procedure
procedure VERIFY_TEMPORARY_CONTROL($\mathsf{C}, F, s'$, SB)
    if $s' \in$ SB then
        isMin := true    // instantaneous perturbation
    else
        $F|_{\mathsf{C}}$ :=COMP_FN_CONTROL($F, \mathsf{C}$)    // $F$ in $\mathsf{BN}|_{\mathsf{C}}$ (see Algorithm 5)
        $\mathsf{SB}|_{\mathsf{C}}$ :=COMP_STATE_CONTROL(SB, C)    // SB in $TS|_{\mathsf{C}}$ (see Algorithm 5)
        if $\mathsf{SB}|_{\mathsf{C}} \neq \emptyset$ then
            $bas^{S}_{TS|_{\mathsf{C}}}(\mathsf{SB}|_{\mathsf{C}})$ :=COMP_STRONG_BASIN_DECOMP($\mathsf{SB}|_{\mathsf{C}}, F|_{\mathsf{C}}$)
            if $s' \in bas^{S}_{TS|_{\mathsf{C}}}(\mathsf{SB}|_{\mathsf{C}})$ then
                isMin := true
            end if
        end if
    end if
    return isMin
end procedure
```

Theorem 5 forms the basis for our algorithm for computing the minimal OT control C, given a source state s and a target attractor A_t. Intuitively, on the application of C, we want the dynamics to move to a state $\mathsf{C}(s)$ which is in the strong basin (w.r.t the restricted transition system $TS|_{\mathsf{C}}$) of the strong basin (w.r.t the original transition system TS) of the target attractor A_t, restricted to states in $S|_{\mathsf{C}}$, in $TS|_{\mathsf{C}}$. Then if we hold the control C for long enough, the dynamics will eventually reach the strong basin of A_t in TS. By Proposition 2, we know that once in the strong basin, the dynamics cannot escape it. This means that finally when the control C is released, the dynamics is in the strong basin of A_t and hence will eventually reach A_t which is the target.

To ensure that we indeed compute a temporary control C that is minimal, we

Algorithm 5 Helper functions

```
procedure COMP_FN_CONTROL($F, \mathsf{C}$) // $\mathsf{C} = \{\mathbb{0}, \mathbb{1}\}$
    $F|_{\mathsf{C}} := F$
    for $i \in \mathbb{1}$ do
        $F|_{\mathsf{C}}[i] := 1$
    end for
    for $i \in \mathbb{0}$ do
        $F|_{\mathsf{C}}[i] := 0$
    end for
    return $F|_{\mathsf{C}}$
end procedure

procedure COMP_STATE_CONTROL($S, \mathsf{C}$)
    $S|_{\mathsf{C}} := S$
    for $s \in S$ do
        for $i \in \mathbb{1}$ do
            if $s[i] \neq 1$ then
                $S|_{\mathsf{C}} := S|_{\mathsf{C}} \setminus s$
            end if
        end for
        for $j \in \mathbb{0}$ do
            if $s[j] \neq 0$ then
                $S|_{\mathsf{C}} := S|_{\mathsf{C}} \setminus s$
            end if
        end for
    end for
    return $S|_{\mathsf{C}}$
end procedure
```

proceed as follows. We start with the state $s' \in bas^W_{TS}(A_t)$ that has the minimal Hamming distance to A_s and $s \in A_s$ is the state that realises this Hamming distance. Let $\mathsf{C}^{A_s \to s'}$ be the control with respect to s'. We check if $s' \in bas^S_{TS|_{\mathsf{C}^{A_s \to s'}}}(bas^S_{TS}(A_t)|_{\mathsf{C}^{A_s \to s'}})$. If so, the control $\mathsf{C}^{A_s \to s'}$ is a minimal OT control and we are done. Otherwise, we remove s' from $bas^W_{TS}(A_t)$ and select the next state s'' from $(bas^W_{TS}(A_t) \setminus \{s'\})$ having the minimal Hamming distance to A_s. We repeat the same procedure this time with the control $\mathsf{C}^{A_s \to s''}$ associated with s''. We iterate till we find a state $s^* \in bas^W_{TS}(A_t)$ with the control $C^{A_s \to s^*}$, such that $s^* \in bas^S_{TS|_{\mathsf{C}^{A_s \to s*}}}(bas^S_{TS}(A_t)|_{\mathsf{C}^{A_s \to s*}})$. It is enough to explore only the states in $bas^W_{TS}(A_t)$ and it will eventually find the required control. Algorithm 4 describes this procedure in pseudocode.

Remark. Notice that the temporary control C should be released after being applied for sufficient time. This is because in the transition system under a temporary control $TS|_{\mathsf{C}}$, the target attractor A_t may be absent. Thus, it is necessary to retrieve the original transition system TS by releasing the control C. Regarding the amount of time for the application of control C, it has to be determined based on the specific system and the detailed perturbations. We believe it is more practical for biologists to determine when to release the control case by case based on experimental settings, rather than interpreting a period of discrete time in real life. Moreover, as long as the control is released in finite steps, holding it longer will not affect its effectiveness.

5.5 One-step permanent control

We now develop an algorithm to solve the minimal OP control problem. The following proposition will be useful.

Proposition 3. *Let* C *be a control and let* A *be an attractor of* TS *such that* A *is also an attractor of* $TS|_{\mathsf{C}}$. *For any* $s \in S$, *if* $s \in bas^{W}_{TS|_{\mathsf{C}}}(A)$ *then* $s \in bas^{W}_{TS}(A)$.

Proof. By the definition of the weak basin, if $s \in bas^{W}_{TS|_{\mathsf{C}}}(A)$, there exists a path ρ from s to A in $TS|_{\mathsf{C}}$. Suppose $\rho = s_0 \rightarrow_{\mathsf{BN}|_{\mathsf{C}}} s_1 \rightarrow_{\mathsf{BN}|_{\mathsf{C}}} \cdots \rightarrow_{\mathsf{BN}|_{\mathsf{C}}} s_k$ where $s_0 = s$ and $s_k \in A$. Further, suppose that ρ is a shortest such path from s to A, that is, $s_i \neq s_j$ for any $i \neq j, 0 \leq i, j \leq k$ and $s_j \notin A$ for any $j < k$. By Lemma 13, for every s_i, $0 \leq i < k$, $s_i \rightarrow_{\mathsf{BN}} s_{i+1}$ is a valid transition in TS and hence ρ is a path in TS. Thus $s \in bas^{W}_{TS}(A)$. □

The converse of Proposition 3 may not hold as shown by the following example.

Example 9. *In Figure 5.2, state* (010) *is in the weak basin of attractor* A_1 *in the original transition system (see Figure 5.2 (a)), whereas state* (010) *is absent in the transition system under control* C *(see Figure 5.2 (b)). That is,* $s \in bas^{W}_{TS}(A)$ *but* $s \notin bas^{W}_{TS|_{\mathsf{C}}}(A)$.

The intuition for the permanent control algorithm that we shall develop in this section is as follows. Suppose $s \in A_s$ is an initial state and A_t is the target attractor of TS that we want the dynamics of BN to always eventually reach. The following is a straightforward corollary of Theorem 4.

Corollary 1. *A control* C *is a permanent control from* s *to* A_t *iff* A_t *is an attractor of* $TS|_{\mathsf{C}}$ *and* $\mathsf{C}(s) \in bas^{S}_{TS|_{\mathsf{C}}}(A_t)$.

Thus, we want to find a control C such that the condition $\mathsf{C}(s) \in bas^{S}_{TS|_{\mathsf{C}}}(A_t)$ is satisfied. Now, since we want the control C to be minimal, we proceed as follows. We start with a state $s' \in bas^{W}_{TS}(A_t)$ that has the minimal Hamming distance with A_s. Let us denote its associated control as $\mathsf{C}^{A_s \rightarrow s'}$. We first check if A_t is an attractor of $TS|_{\mathsf{C}^{A_s \rightarrow s'}}$ since otherwise, $\mathsf{C}^{A_s \rightarrow s'}$ cannot be a permanent control (by definition). If A_t is indeed an attractor of $TS|_{\mathsf{C}^{A_s \rightarrow s'}}$, we check if $s' \in bas^{S}_{TS|_{\mathsf{C}^{A_s \rightarrow s'}}}(A_t)$. If so, $\mathsf{C}^{A_s \rightarrow s'}$ is a minimal OP control. Otherwise, we remove s' from $bas^{W}_{TS}(A_t)$ and select the next state s'' from $(bas^{W}_{TS}(A_t) \setminus \{s'\})$ having the minimal Hamming distance with A_s. We repeat the same procedure this time with s''. We iterate the above procedures till we find a minimal OP control.

The procedure described above, in the worst case, explores all possible states in $bas^{W}_{TS}(A_t)$. By Proposition 3, we know that for any control C, $bas^{W}_{TS|_{\mathsf{C}}}(A_t) \subseteq bas^{W}_{TS}(A_t)$. Thus, it is enough to explore only the states in $bas^{W}_{TS}(A_t)$ and it will eventually find the control satisfying the constraints. Algorithm 6 describes this procedure in pseudocode.

Example 10. *To continue with Example 8, we compute the minimal OT and OP control from* A_3 *to* A_1 *with Algorithm 4 and Algorithm 6. Procedures described in Algorithm 4 are used to compute the minimal OT control. It first computes the weak basin and the strong basin of* A_1, $\mathsf{WB} = \{000, 001, 010, 011, 100, 101\}$ *and* $\mathsf{SB} = \{000, 001\}$. *In Figure 5.3(a), states* $s_1 = (011)$ *and* $s_2 = (101)$ *in* WB *have the minimal Hamming distance to* A_3 *and their corresponding control sets are* $\mathsf{C}_1 = (\mathbb{0}_1, \mathbb{1}_1)$, *where* $\mathbb{0}_1 = \{1\}$ *and* $\mathbb{1}_1 = \emptyset$, *and* $\mathsf{C}_2 = (\mathbb{0}_2, \mathbb{1}_2)$, *where* $\mathbb{0}_2 = \{2\}$ *and* $\mathbb{1}_2 = \emptyset$. *Figures 5.3(a) and (b) describe the*

Algorithm 6 Minimal OP control

```
procedure MIN_OP_CONTROL($F, A_s, A_t$)                  // $A_s$: the source; $A_t$: the target
    WB :=COMP_WEAK_BASIN($A_t, F$)                       // the weak basin of $A_t$ in TS
    $C_{min}$ :=Comp_Min_OP_Control($F, A_s$, WB, SB)
    return $C_{min}$
end procedure
procedure COMP_MIN_OP_CONTROL($F, A_s$, WB, SB)
    isMin := false, $C_{min} = \emptyset$
    while isMin = false and WB $\neq \emptyset$ do
        ($C, s, s'$) :=COMP_CONTROL_SET($A_s$, WB)
        isMin :=VERIFY_PERMANENT_CONTROL($C, F, s', A_t$)
        if isMin then
            $C_{min}$ := C
        else
            WB := WB $\setminus s'$
        end if
    end while
    return $C_{min}$
end procedure
procedure VERIFY_PERMANENT_CONTROL($C, F, s', A_t$)
    if $s'|_C = A_t|_C$ then                             // $A_t$ is preserved in $TS|_C$
        $F|_C$ :=COMP_FN_CONTROL($F$, C)                 // $F$ in $BN|_C$ (see Algorithm 5)
        $SB|_C$ :=COMP_STRONG_BASIN_DECOMP($A_t, F|_C$)  // the strong basin of $A_t$ in $TS|_C$
        if $s' \in SB|_C$ then
            isMin := true
        end if
    end if
    return isMin
end procedure
```

application and release of C_1 *and* C_2*, respectively. In both* $TS|_{C_1}$ *and* TS_{C_2}*, only attractor* A_1 *is preserved and all the states belong to the strong basin of* A_1 *in the transition system under control. In* $TS|_{C_1}$*, the network will eventually and surely reach* (001)*, a state in* SB*, from which the control can be released and spontaneous evolution will guide the network towards* A_1*. In* $TS|_{C_2}$*, the network will either follow the path* $\rho_1 = (101) \rightarrow (001) \rightarrow (000)$ *or* $\rho_2 = (101) \rightarrow (100) \rightarrow (000)$*. If the network takes path* ρ_1*,* C_2 *can be released when the network reaches state* (001)*. If the network takes path* ρ_2*,* C_2 *can only be released when the network settles down to* (000)*. Applying the temporary control for longer time will not affect the inevitable reachability of the target attractor.*

Procedures described in Algorithm 6 are used to compute the minimal OP control. The process is similar to the minimal OT control. For this case, the controls C_1 *and* C_2 *are also the minimal OP control from* A_3 *to* A_1*.*

Constraints on perturbations

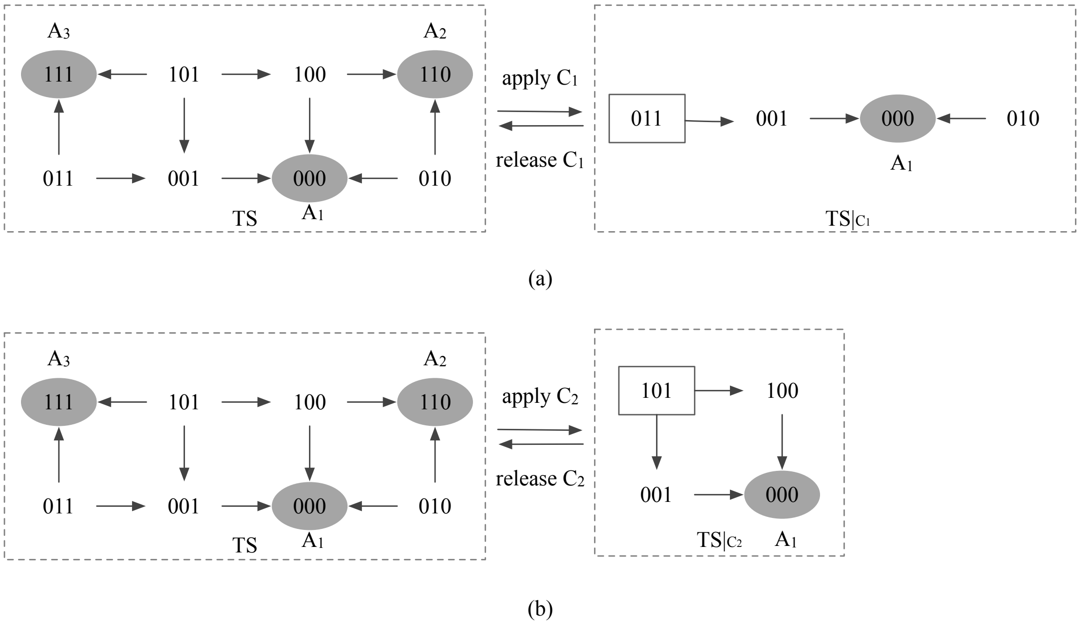


Figure 5.3: Minimal OT and OP control of BN of Example 10. (a) and (b) describe the control effects of C_1 and C_2, respectively.

To translate the predictions based on mathematical models to biological experiments may encounter many difficulties. For instance, essential genes for cell survival [ZL08] should not be perturbed in biological experiments. Some genes may be very difficult or expensive to perturb with the latest techniques. To accelerate this translation process, we bridge the gap between theoretical computation and biological experiments by taking practical constraints into consideration. Based on the specific system and reprogramming tasks, we encode practical requirements as preconditions and avoid undesired or unrealistic perturbations during the computations.

We consider three kinds of nodes:

- $\mathsf{R}_{\mathbb{0}}$: nodes that cannot be perturbed from 0 to 1;
- $\mathsf{R}_{\mathbb{1}}$: nodes that cannot be perturbed from 1 to 0;
- R: nodes that cannot be perturbed in either direction.

The basic idea is that we exclude the intermediate states that will involve undesired perturbations from the search space, which is the strong basin of the target attractor for instantaneous control and the weak basin of the target attractor for temporary and permanent control.

Let Ξ denote the search space, either the strong basin or the weak basin depending on the control method we use. The values of nodes in R always stay the same as the initial state s. The set of available intermediate states $\Xi' \subseteq \Xi$ can be computed using the cross operation as $\Xi' = \Xi \otimes s|_{\mathsf{R}}$. Intuitively, if node $x \in \mathsf{R}$ has a value of 0 in the source state s, we restrict the search space to the set of states, in which x has a value of 0. Otherwise, if x is 1 in the intermediate state, the value of x needs to be flipped from 0 to 1, which violates the constraint. For node $x \in \mathsf{R}_{\mathbb{0}}$, it is not allowed to perturb the value of x from 0 to 1. However, it is possible that x has a value of 1 in the initial state s. The control of such nodes can happen only from 1 to 0, which does not violate the restriction. Hence, given a set $\mathsf{R}_{\mathbb{0}}$, we compute the subset $\mathsf{R}'_{\mathbb{0}}$ of $\mathsf{R}_{\mathbb{0}}$, such that $\mathsf{R}'_{\mathbb{0}} = \{x \in \mathsf{R}_{\mathbb{0}} |\ s|_x = 0 \text{ for } x \in \mathsf{R}_{\mathbb{0}}\}$ Similarly, we update $\mathsf{R}_{\mathbb{1}}$ to $\mathsf{R}'_{\mathbb{1}} = \{x \in \mathsf{R}_{\mathbb{1}} |\ s|_x = 1 \text{ for } x \in \mathsf{R}_{\mathbb{1}}\}$. Then, we restrict the remaining search space Ξ' using the cross operation as $\Xi'' = \Xi' \otimes s|_{\mathsf{R}_{\mathbb{0}}} \otimes s|_{\mathsf{R}_{\mathbb{1}}}$ and explore Ξ'' for solutions.

5.6 Attractor-based sequential instantaneous control

In Sections 5.3, 5.4 and 5.5, we have developed algorithms for the minimal OI, OT and OP control. Based on these algorithms, in this and the following two sections, we shall develop algorithms for computing ASI, AST and ASP control paths with at most k perturbations.

Given a Boolean network $\mathsf{BN} = (X, F)$, let $\mathcal{A}$ be the set of attractors of TS, A_s and A_t be the source and target attractors. We let $\Phi = \langle A_1, A_2, \ldots, A_m \rangle$, where $A_1 = A_s, A_m = A_t$, $A_i \neq A_j$ for any $i, j \in [1, m]$ and $2 \leq m \leq |\mathcal{A}|$, denote the sequence of the identified attractors. As defined in Definition 22, a sequential control path Ω is made up of a sequence of one-step control sets: $\langle \mathsf{C}_{\min}^{A_1 \to A_2}, \mathsf{C}_{\min}^{A_2 \to A_3}, \ldots, \mathsf{C}_{\min}^{A_{m-1} \to A_m} \rangle$. Theorem 4 shows that to compute the minimal OI control for each step, let us say from A_i to A_j, $i, j \in [1, m]$, we first compute the strong basin of A_j, $bas^S_{TS}(A_j)$, and then find the state $s_j \in bas^S_{TS}(A_j)$ that has the minimal Hamming distance to A_i. Suppose the state s_i is the state in A_i that realises this minimal Hamming distance. By comparing the values of the nodes in s_i and s_j, we are able to compute the minimal OI control $\mathsf{C}_{\min}^{A_i \to A_j}$.

Algorithm 7 describes a procedure, called ASI_Control, for computing ASI control paths with at most k perturbations. This algorithm is based on the decomposition-based computation of strong basins, Comp_Strong_Basin_Decomp. The procedure ASI_control takes as inputs the Boolean functions F, a threshold k of the number of perturbations, the source attractor A_s, the target attractor A_t, and the set of attractors $\mathcal{A}$ of TS.

Algorithm 7 contains two parts. The first part includes lines 2-13. It computes the strong basin SB_{A_t} of the target attractor A_t (line 3). For each attractor A, ($A \in \mathcal{A}$ and $A \neq A_t$), it generates a dictionary L_A to save all the valid ASI control paths from A to A_t. It then computes the minimal OI control from A to A_t, denoted $\mathsf{C}_{A \to A_t}$. Let $\#p$ represent the number of perturbations required by $\mathsf{C}_{A \to A_t}$. $\mathsf{C}_{A \to A_t}$ is considered valid and saved to L_A if either (1) A is the source attractor A_s and the number of perturbations $\#p$ is not greater than k; or (2) A is not A_s and $\#p$ is less or equal to $(k-1)$. If A is an intermediate attractor, namely $A \neq A_s$, $\mathsf{C}_{A_s \to A}$ would require at least one perturbation. Therefore, the size of $\mathsf{C}_{A \to A_t}$ should not exceed $(k-1)$. A is saved to I as an intermediate attractor if $A \neq A_s$ and $\#p \leq k-1$.

The second part includes lines 14-36. It extends the control paths computed in the previous part by recursively taking every intermediate attractor as a new target

Algorithm 7 Attractor-based sequential instantaneous control

```
 1: procedure ASI_CONTROL($F, k, A_s, A_t, \mathcal{A}$)
 2:     Initialise a list $I := \emptyset$ to store possible intermediate attractors.
 3:     $SB_{A_t}$ :=COMP_STRONG_BASIN_DECOMP($F, A_t$)          // strong basin of the target
 4:     Initialise a dictionary to store paths $\mathcal{L} := \{L_{A_1}, L_{A_2}, \ldots, L_{A_m}\},\ A_i \in \mathcal{A}$.
 5:     for $A \in (\mathcal{A} \setminus A_t)$ do          //find attractors that have shorter paths to $A_t$
 6:         $\mathsf{C}_{A \to A_t}$ :=COMP_CONTROL_SET($A, \mathsf{SB}_{A_t}$)
 7:         if ($A = A_s$ and $|\mathsf{C}_{A \to A_t}| \leq k$) or ($A \neq A_s$ and $|\mathsf{C}_{A \to A_t}| \leq k - 1$) then
 8:             $\Phi_{A \to A_t}$.add($A_t$)
 9:             $\Omega_{A \to A_t}$.add($\mathsf{C}_{A \to A_t}$)
10:             Add the path $(\Phi_{A \to A_t}, \Omega_{A \to A_t})$ to $L_A$
11:             Add $A$ to $I$ as a candidate intermediate if $A \neq A_s$.
12:         end if
13:     end for
14:     while $I \neq \emptyset$ do
15:         Initialise a new list $I' := \emptyset$
16:         for $A'_t \in I$ do                                        // new target
17:             $SB_{A'_t}$ :=COMP_STRONG_BASIN_DECOMP($F, A'_t$)
18:             for $A'_s \in (\mathcal{A} \setminus (A'_t \cup A_t))$ do                        // new source
19:                 $\mathsf{C}_{A'_s \to A'_t}$ :=COMP_CONTROL_SET($A'_s, \mathsf{SB}_{A'_t}$)
20:                 for $(\Phi_{A'_t \to A_t}, \Omega_{A'_t \to A_t}) \in L_{A'_t}$ do
21:                     $\Phi_{A'_s \to A_t} := \Phi_{A'_t \to A_t}$; Insert $A'_t$ to the beginning of $\Phi_{A'_s \to A_t}$.
22:                     if $A'_s \notin \Phi_{A'_t \to A_t}$ then
23:                         $h :=$ the total number of perturbations required by $\Omega_{A'_t \to A_t}$.
24:                         $h := h + |\mathsf{C}_{A'_s \to A'_t}|$
25:                         if ($A'_s = A_s$ and $h \leq k$) or ($A'_s \neq A_s$ and $h \leq k - 1$) then
26:                             $\Omega_{A'_s \to A_t} := \Omega_{A'_t \to A_t}$
27:                             Insert $\mathsf{C}_{A'_s \to A'_t}$ to the beginning of $\Omega_{A'_s \to A_t}$.
28:                             Add the extended path $(\Phi_{A'_s \to A_t}, \Omega_{A'_t \to A_t})$ to $L_{A'_s}$.
29:                             Add $A'_s$ to $I'$ as a candidate intermediate if $A'_s \neq A_s$.
30:                         end if
31:                     end if
32:                 end for
33:             end for
34:         end for
35:         $I := I'$
36:     end while
37:     Return $L_{A_s}$
38: end procedure
```

and computing the minimal control from other attractors to the new target attractor. Specifically, for each new target attractor $A'_t \in I$, it computes the minimal OI control $\mathsf{C}_{A'_s \to A'_t}$ from A'_s in $(\mathcal{A} \setminus (A'_t \cup A_t))$ to A'_t (line 19). Then, for every sequential path from A'_t to A_t, for instance $(\Phi_{A'_t \to A_t}, \Omega_{A'_t \to A_t})$, it verifies whether A'_s can be appended to the beginning of $\Phi_{A'_t \to A_t}$ to form a new path from A'_s to A'_t based on the following conditions: (1) A'_s is not an intermediate in path $A'_t \to \ldots \to A_t$; and (2) the total number of perturbations of the new path $\Phi_{A'_s \to A_t}$ should not exceed k (or $k - 1$) if $A'_s = A_s$ (or $A'_s \neq A_s$). When both conditions are satisfied, we save the new path to $L_{A'_s}$ (line 28) and add A'_s to I' as a candidate intermediate if $A'_s \neq A_s$

Algorithm 8 Attractor-based sequential temporary control

```
 1: procedure AST_Control($F, k, A_s, A_t, \mathcal{A}$)
 2:     Initialise a list $I := \emptyset$ to store possible intermediate attractors.
 3:     $WB_{A_t}$ :=Comp_Weak_Basin($F, A_t$)                    // weak basin of the target
 4:     $SB_{A_t}$ :=Comp_Strong_Basin_Decomp($F, A_t$)           // strong basin of the target
 5:     Initialise a dictionary to store paths $\mathcal{L} := \{L_{A_1}, L_{A_2}, \ldots, L_{A_m}\},\ A_i \in \mathcal{A}$.
 6:     for $A \in (\mathcal{A} \setminus A_t)$ do          //find attractors that have shorter paths to $A_t$
 7:         $\mathsf{C}_{A\to A_t}$ :=Comp_Min_OT_Control($A, \mathsf{WB}_{A_t}, \mathsf{SB}_{A_t}$)
 8:         if ($A = A_s$ and $|\mathsf{C}_{A\to A_t}| \leq k$) or ($A \neq A_s$ and $|\mathsf{C}_{A\to A_t}| \leq k-1$) then
 9:             $\Phi_{A\to A_t}$.add($A_t$)
10:             $\Omega_{A\to A_t}$.add($\mathsf{C}_{A\to A_t}$)
11:             Add the path $(\Phi_{A\to A_t}, \Omega_{A\to A_t})$ to $L_A$
12:             Add $A$ to $I$ as a candidate intermediate if $A \neq A_s$.
13:         end if
14:     end for
15:     while $I \neq \emptyset$ do
16:         Initialise a new list $I' := \emptyset$
17:         for $A'_t \in I$ do                                              // new target
18:             $WB_{A'_t}$ :=Comp_Weak_basin($F, A'_t$)
19:             $SB_{A'_t}$ :=Comp_Strong_basin_Decomp($F, A'_t$)
20:             for $A'_s \in (\mathcal{A} \setminus (A'_t \cup A_t))$ do                       // new source
21:                 $\mathsf{C}_{A'_s\to A'_t}$ :=Comp_Min_OT_Control($A, \mathsf{WB}_{A_t}, \mathsf{SB}_{A_t}$)
22:                 for $(\Phi_{A'_t\to A_t}, \Omega_{A'_t\to A_t}) \in L_{A'_t}$ do
23:                     $\Phi_{A'_s\to A_t} := \Phi_{A'_t\to A_t}$; Insert $A'_t$ to the beginning of $\Phi_{A'_s\to A_t}$.
24:                     if $A'_s \notin \Phi_{A'_t\to A_t}$ then
25:                         $h :=$ the total number of perturbations required by $\Omega_{A'_t\to A_t}$.
26:                         $h := h + |\mathsf{C}_{A'_s\to A'_t}|$
27:                         if ($A'_s = A_s$ and $h \leq k$) or ($A'_s \neq A_s$ and $h \leq k-1$) then
28:                             $\Omega_{A'_s\to A_t} := \Omega_{A'_t\to A_t}$
29:                             Insert $\mathsf{C}_{A'_s\to A'_t}$ to the beginning of $\Omega_{A'_s\to A_t}$.
30:                             Add the extended path $(\Phi_{A'_s\to A_t}, \Omega_{A'_t\to A_t})$ to $L_{A'_s}$.
31:                             Add $A'_s$ to $I'$ as a candidate intermediate if $A'_s \neq A_s$.
32:                         end if
33:                     end if
34:                 end for
35:             end for
36:         end for
37:         $I := I'$
38:     end while
39:     Return $L_{A_s}$
40: end procedure
```

(line 29). After going through all the intermediate attractors in I (lines 16-34), it updates the set of intermediate attractors I and repeats steps at lines 14-36 until I is empty. In the end, all the ASI control paths from A_s to A_t are saved in L_{A_s}.

5.7 Attractor-based sequential temporary control

Given a Boolean network $\mathsf{BN} = (X, F)$, let $\mathcal{A}$ be the set of attractors of TS, A_s and A_t be the source and target attractors, respectively. Algorithm 8 describes a procedure, called AST_Control, for computing AST control paths within k perturbations. Procedure AST_Control is based on previously developed algorithms for the computation of weak basin and strong basin, denoted as Comp_Weak_Basin and Comp_Strong_Basin_Decomp, and the algorithm for the minimal OT control, namely Comp_Min_OT_Control.

The overall procedures of AST_Control are very similar to ASI_Control. It also traverses attractors in $\mathcal{A}$ to find possible intermediate attractors and then tries to extend the sequential paths. Here we only highlight the differences between AST_Control and ASI_Control as follows. Because at each step, temporary control searches the weak basin of the (intermediate) target attractor A_t (A'_t) rather than the strong basin for solutions, procedure AST_Control computes the weak basin of A_t (A'_t) besides the strong basin (lines 3 and 18). Given the strong basin and the weak basin of A_t (A'_t), it uses the procedure Comp_Min_OT_Control in Algorithm 4 to compute the minimal OT control from A_s to A_t (or from A'_s to A'_t). Let $\Phi = \langle A_1, A_2, \ldots, A_m \rangle$, where $A_1 = A_s, A_m = A_t$, $A_i \neq A_j$ for any $i, j \in [1, m]$ and $2 \leq m \leq |\mathcal{A}|$, denote the sequence of the attractors of an AST control path. To compute the minimal OT control for each step, let us say from A_i to A_j, procedure Comp_Min_OT_Control explores the weak basin $bas^W_{TS}(A_j)$ of A_j starting from the state s' that has the minimal Hamming distance to the temporal source attractor A_i. If the corresponding control $\mathsf{C}_{A_i \to A_j}$ results in a transition system $TS|_{\mathsf{C}_{A_i \to A_j}}$ such that s' is in the strong basin of the remaining strong basin of A_j, denoted $bas^S_{TS|_{\mathsf{C}_{A_i \to A_j}}}(bas^S_{TS}(A_t \cap S|_{\mathsf{C}_{A_i \to A_j}}))$, $\mathsf{C}_{A_i \to A_j}$ is a minimal OT control from A_i to A_j. Otherwise, we move on to the next candidate. Eventually, we will find a minimal OT control. Because the temporary control C leads to a temporary change of the dynamics, it is possible that the temporal target attractor A_j is not preserved in $TS|_{\mathsf{C}_{A_i \to A_j}}$. As long as some state in $bas^S_{TS}(A_j)$ is preserved in $TS|_{\mathsf{C}_{A_i \to A_j}}$, namely $(bas^S_{TS}(A_j) \cap S|_{\mathsf{C}_{A_i \to A_j}}) \neq \emptyset$, and s' is in the strong basin of the remaining strong basin in $TS|_{\mathsf{C}(bas^S_{TS}(A_j) \cap S|_{\mathsf{C}_{A_i \to A_j}})}$, s' will surely evolve to $(bas^S_{TS}(A_j) \cap S|_{\mathsf{C}_{A_i \to A_j}})$ when applying the control for sufficient time. Afterwards, the control can be released and the network will eventually evolve to A_t spontaneously. The others steps for the construction of sequential steps remain the same as procedure ASI_Control.

5.8 Attractor-based sequential permanent control

In this section, we shall develop an algorithm to solve the ASP control problem. We have developed an algorithm to compute the minimal OP control [SPP19b], denoted as OP_Control, based on Corollary 1. The basic idea is that we start from the state s' in the weak basin of the target attractor that have the minimal Hamming distance to the source attractor, and verify if (1) the the corresponding control preserves the

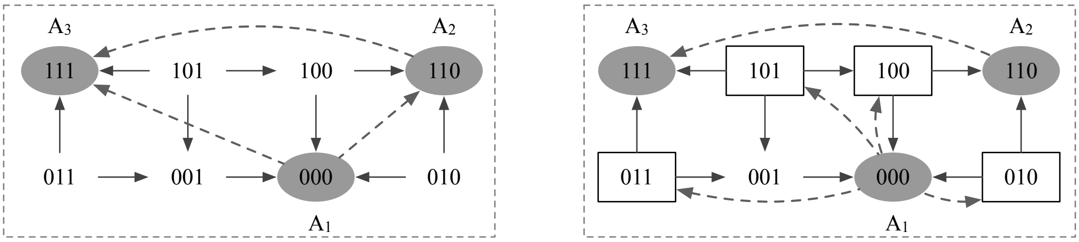
Algorithm 9 Attractor-based sequential permanent control

```
1: procedure PERM_CONTROL_VALIDATION($\mathsf{C}_{A'_s \to A'_t}, A'_t, \Phi_{A'_t \to A_t}, \Omega_{A'_t \to A_t}$)
2:     $A_1 := \Phi[0]$                          // the first intermediate $A_1$ in $\Phi_{A'_t \to A_t}$
3:     $\mathsf{C}_{A'_t \to A_1} := \Omega[0]$                 // the first control set $\mathsf{C}_{A'_t \to A_1}$ in $\Omega_{A'_t \to A_t}$
4:     $\Phi' := \Phi_{A'_t \to A_t}$.pop(), $\Omega' := \Omega_{A'_t \to A_t}$.pop()      //delete the first element
5:     $\mathsf{C}'' := \mathsf{C}_{A'_s \to A'_t} \setminus \mathsf{C}_{A'_t \to A_1}$
6:     $isValid := True$
7:     if $A'_t|_{\mathsf{C}''} = A_1|_{\mathsf{C}''}$ and $\Phi' \neq \emptyset$ then
8:         $isValid$ :=PERM_CONTROL_VALIDATION($\mathsf{C}'', A'_t, \Phi', \Omega'$)
9:     else if $A'_t|_{\mathsf{C}''} \neq A_1|_{\mathsf{C}''}$ then
10:        $isValid := False$
11:    end if
        return $isValid$
12: end procedure
```

target attractor and (2) the intermediate state s' is in the strong basin of the target attractor in the transition system under control. If so, we are done; otherwise, we exclude the state s' from the weak basin of the target attractor and repeat the above steps until we find a valid permanent control.

The algorithm for ASP control explores the same way as Algorithm 8 to construct sequential paths, but it is more involved. It can be achieved by modifying procedure AST_CONTROL in Algorithm 8 as follows. First, at lines 7 and 21, we simply replace the procedure COMP_MIN_OT_CONTROL with the procedure COMP_MIN_OP_CONTROL. Second, when extending the sequential paths, besides the conditions at line 27, we add the procedure PERM_CONTROL_VALIDATION to verify whether the control $\mathsf{C}_{A'_s \to A'_t}$ can be inserted to the beginning of $\Omega_{A'_t \to A_t}$. Because for each control step of AST, the temporary perturbations are released at one time point to retrieve the original transition system and let the network evolve spontaneously to the the intermediate/target attractor. But ASP control adopts permanent control that will be maintained for all the following time steps. Therefore, when extending a permanent control C to the beginning of a sequential path, it has to be verified whether the application of C will affect the reachability of the following control steps. To avoid duplication, here we only give the explanations of the procedure PERM_CONTROL_VALIDATION in Algorithm 9. The purpose of this procedure is to verify whether the control $\mathsf{C}_{A'_s \to A'_t}$ can be added to the beginning of $\Phi_{A'_t \to A_t}$ to form a new path $\Phi_{A'_s \to A_t}$. The verification is carried out recursively. Let us assume $\Phi_{A'_t \to A_t} = \langle A_1, A_2, \ldots, A_t \rangle$. The first intermediate attractor is A_1 and the control from A'_t to A_1 is $\mathsf{C}_{A'_t \to A_1}$. Since $\mathsf{C}_{A'_s \to A'_t}$ and $\mathsf{C}_{A'_t \to A_1}$ may require to perturb the same node in the opposite way, we compute $\mathsf{C}'' = \mathsf{C}_{A'_s \to A'_t} \setminus \mathsf{C}_{A'_t \to A_1}$. If the projections of A'_t and A_1 to C'' are the same, A_1 is preserved under the permanent control C'' and we proceed to the remaining control steps (lines 7-8); otherwise, $\mathsf{C}_{A'_s \to A'_t}$ is not valid (lines 9-10).

Example 11. *For the Boolean network given in Example 1, we compute ASI, AST and ASP control paths from A_1 and A_3. The thresholds of the number of perturbations for ASI, AST and ASP are set as the number of perturbations required by the minimal OI, OT and OP, respectively. Figure 5.4 (a) shows the ASI control paths. The minimal OI*



(a) (b)

Figure 5.4: Sequential control of the Boolean network of Example 11. (a) shows the ASI control paths and (b) shows the AST/ASP control paths.

control drives the network from (000) *directly to* (111) *and requires three perturbations. ASI control identifies a sequential path* $(000) \xrightarrow{\{x_1=1,x_2=1\}} (110) \xrightarrow{\{x_3=1\}} (111)$*, which also needs three perturbations in total. The results of AST and ASP are the same as shown in Figure 5.4* (b)*. The minimal OT and OP control requires two perturbations by stirring the network from* (000) *to the intermediate state* (101) *or* (011)*. AST and ASP identify two sequential control paths:* $(000) \xrightarrow[\{x_2=1\}]{\{x_1=1\}} (110) \xrightarrow{\{x_3=1\}} (111)$.

Constraints on intermediate attractors for sequential control

Given a Boolean network, some of its attractors are not suitable to play the role of intermediates for sequential control. For instance, mature red blood cells lack a cell nucleus in order to carry maximal oxygen to the body. Such cell cannot be reprogrammed to other cells and thus are not qualified for intermediate attractors.

Our algorithms for attractor-based sequential control can be modified to exclude such undesired attractors as intermediate attractors. Let $\mathcal{U}$ denote the set of undesired attractors. The implementation of restrictions on intermediate attractors is quite straightforward. The input $\mathcal{A}$ of algorithms for ASI, AST and ASP stores the candidate intermediate attractors. Thus, we simply exclude attractors in $\mathcal{U}$ from $\mathcal{A}$ and keep the other procedures unchanged. In this way, undesired attractors will be skipped during the computation.

5.9 Evaluation

In this section, we apply the six source-target control methods, including OI, OT, OP, ASI, AST and ASP, to a number of real-life biological networks to demonstrate their efficiency and efficacy. All the methods are implemented in our software CABEAN [SP20a] based on the model checker MCMAS [LQR17] to encode Boolean networks into the efficient data structure BDDs. All the experiments are performed on a high-performance computing (HPC) platform, which contains CPUs of Intel Xeon Gold 6132 @2.6 GHz.

The comparison of the six control methods focuses on the number of perturbations, the number of solutions and the computational time. We will analyse in depth the control results of the cardiac gene regulatory network, the myeloid differentiation network, and the tumour network, and give an overview of the results of some other

	FHF	SHF	Attractor
BMP2	1	0	1
canWnt	0	1	0
Dkk1	0	0	0
Fgf8	0	1	0
Foxc12	0	1	0
GATAs	1	1	0
Isl1	0	1	0
Mesp1	0	0	0
Nkx25	1	1	0
Tbx1	0	1	0
Tbx5	1	0	0
exogenBMP2I	1	1	1
exogenBMP2II	1	1	1
exogenCanWntI	0	1	0
exogenCanWntII	0	1	0

Table 5.1: Attractors of the cardiac network.

	FHF → SHF	SHF → FHF
OI	{exogenCanWntI=1}	{BMP2=1 canWnt=0 exogenCanWntI=0 exogenCanWntII=0 }

OT	{exogenCanWntI=1}	{Tbx5=1 exogenCanWntI=0}
		{BMP2=1 exogenCanWntI=0}
		{Mesp1=1 exogenCanWntI=0}

OP	{exogenCanWntI=1}	{Tbx5=1 exogenCanWntI=0}
		{BMP2=1 exogenCanWntI=0}

Table 5.2: One-step control of the cardiac network.

networks. We do not compare our methods with other methods in the literature, because as discussed in Section 1.4, most of the methods proposed in the literature are not dealing with the problem of source-target control of asynchronous Boolean networks. The most relevant works by Mandon *et al.* [MHP16, MHP17] cannot deal with networks with more than around 20 nodes.

5.9.1 Control of the cardiac gene regulatory network

The cardiac gene regulatory network is constructed for the early cardiac gene regulatory network of the mouse, including the core genes required for the cardiac development and the first heart field (FHF) and the second heart field determination [HGZ$^+$12]. This network has 15 nodes, out of which exogenBMP2I and exogenCanWntI are input nodes. We refer the structure of the network to the original work [HGZ$^+$12]. Under the initial condition given in [HGZ$^+$12], where the input node exogenBMP2I is set to 1, this network contains three attractors listed in Table 5.1. Two of the attractors correspond to FHF and SHF.

	FHF → SHF	SHF → FHF
OI	0.001	0.006
OT	0.003	0.067
OP	0.002	0.046

Table 5.3: Computational time for one-step control of the cardiac network. The units of time are seconds.

We compute different control paths for the conversion between FHF and SHF. For the conversion either from FHF to SHF or from SHF to FHF, there does not exist any sequential control path (ASI, AST or ASP). The results of one-step control, including OI, OT and OP, are given in Table 5.2. The results show that it is necessary to control exogenCanWntI for the conversion between FHF and SHF. This is quite straightforward, since exogenCanWntI is a non-initialised input node and it can take a value of either 0 or 1. Moreover, exogenCanWntI has a value of 1 in SHF and has a value of 0 in FHF as shown in Table 5.1. Thus, exogenCanWntI is essential for the transformation.

To reprogram FHF to SHF, it is sufficient to control exogenCanWntI with instantaneous, temporary or permanent perturbations. But more perturbations are required to reprogram SHF to FHF. The minimal OI control requires four perturbations, while the minimal OT and OP control require only two perturbations. The minimal OT and OP control find two common solutions, {BMP2=1 exogenCanWntI=0} and {BMP2=1 exogenCanWntI=0}. That is, the temporary or permanent control of these two solutions will both guide the network from SHF to FHF. For such cases, temporary control is preferable to avoid unknown consequences that may be caused by the permanent change of the dynamics. The OT control {Mesp1=1 exogenCanWntI=0} cannot be applied with permanent perturbations, because attractor FHF vanishes in its resulting transition system, which makes it impossible to reach FHF. The computational time given in Table 5.3 shows that our methods for one-step control are all very efficient.

5.9.2 Control of the myeloid differentiation network

Hematopoiesis is the process through which mature blood cells are manufactured. The myeloid differentiation network is constructed to model the differentiation process of common myeloid progenitors (CMPs) into four types of mature blood cells, including megakaryocytes, erythrocytes, granulocytes and monocytes [KMST11]. This network consists of 11 transcription factors as shown in Figure 5.5. Their Boolean functions can be found in the original work [KMST11].

With the attractor detection method [MPQY19], we identify six singleton attractors as shown in Table 5.4. It has been validated that expressions of four attractors correspond to microarray expression profiles of megakaryocytes, erythrocytes, granulocytes and monocytes [KMST11]. Attractor A_1 is an all-zero attractor, where all the nodes have a value of 0. Attractor A_2 with the activation of PU1, cJun and EgrNab might be caused by pathological alterations [KMST11].

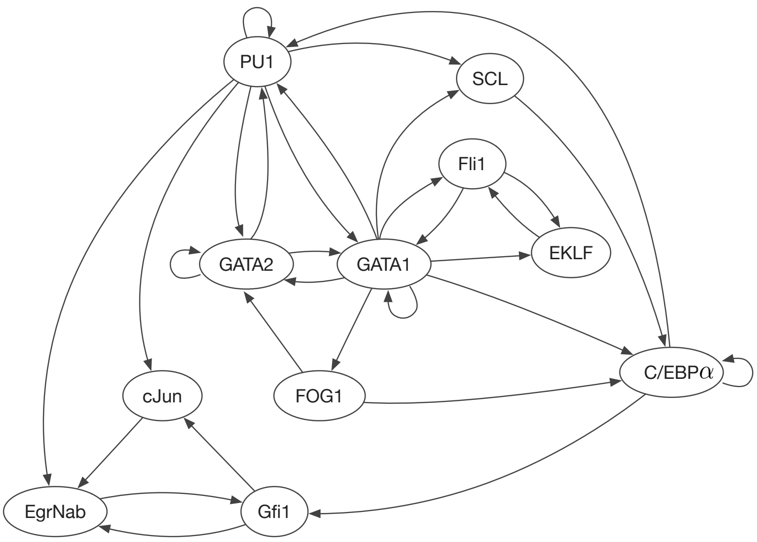


Figure 5.5: The structure of the myeloid differentiation network.

nodes	A_1	A_2	granulocytes	monocytes	megakaryocytes	erythrocytes
GATA2	0	0	0	0	0	0
GATA1	0	0	0	0	1	1
FOG1	0	0	0	0	1	1
EKLF	0	0	0	0	0	1
Fli1	0	0	0	0	1	0
SCL	0	0	0	0	1	1
CEBPα	0	0	1	1	0	0
PU1	0	1	1	1	0	0
cJun	0	1	0	1	0	0
EgrNab	0	1	0	1	0	0
Gfi1	0	0	1	0	0	0

Table 5.4: Attractors of the myeloid differentiation network.

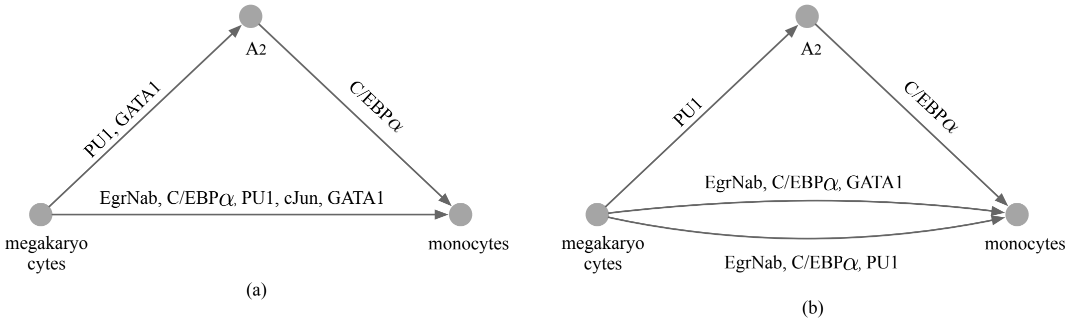


Figure 5.6: Source-target control from megakaryocytes to monocytes.

Conversion from megakaryocytes to monocytes

We compute the control for the conversion from megakaryocytes to monocytes. The control paths identified by OI and ASI are illustrated in Figure 5.6 (a) and the control paths of OT, OP, AST and ASP are shown in Figure 5.6 (b). In particular, the results of OT and OP and the results of AST and ASP are identical. We can

	Control without constraints	Control with constraints

OI	{GATA1=0 C/EBPα=1 PU1=1 Gfi1=1}
	{GATA1=0 Fli1=0 C/EBPα=1 Gfi1=1}
		$\emptyset$

OT	{C/EBPα=1 PU1=1 Gfi1=1}
	{GATA1=0 C/EBPα=1 Gfi1=1}
		{C/EBPα=1 PU1=1 Gfi1=1}

OP	{C/EBPα=1 PU1=1 Gfi1=1}
	{GATA1=0 C/EBPα=1 Gfi1=1}
		{C/EBPα=1 PU1=1 Gfi1=1}

Table 5.5: One-step control from megakaryocytes to granulocytes with and without constraints on the perturbations.

GATA2	GATA1	FOG1	EKLF	Fli1	SCL	C/EBPα	PU1	cJun	EgrNab	Gfi1
0	0	*	*	0	*	1	*	*	0	1
0	0	*	*	1	*	1	1	*	0	1

Table 5.6: Strong basin of the attractor granulocytes. Symbol '*' means the node can take a value of both 0 and 1.

see that the minimal OI control requires the activation of EgrNab, C/EBPα, PU1, cJun and the inhibition of GATA1, while OT or OP can achieve the goal by either (1) the activation of EgrNab, C/EBPα and PU1; or (2) the activation of EgrNab and C/EBPα, together with the inhibition of GATA1. All the sequential paths need two steps, where A_2 is adopted as the intermediate attractor. For the first step, ASI activates PU1 and inhibits GATA1, while AST and ASP only need to activate PU1. When the network converges to A_2, the three methods require to activate C/EBPα. After that, the network will evolve spontaneously to the target attractor monocytes.

We can easily observe that OT and OP require fewer perturbations than OI. Compared to one-step control, sequential control has the potential to identify novel control paths with even fewer perturbations. In particular, Figure 5.6 shows that AST and ASP are able to identify a path with only two perturbations, while ASI requires at least three perturbations.

The efficacy of the identified sequential temporary/permanent path is confirmed by the predictions in [KMST11]. We spot the critical role of PU1 and C/EBPα during the transdifferentiation from megakaryocytes to monocytes. According to the expression profiles, both PU1 and C/EBPα are not expressed in MegE lineage (megakaryocytes and erythrocytes), while they are expressed in GM lineage (granulocytes and monocytes). In this network, no regulator can activate C/EBPα and PU1 is primarily activated by C/EBPα. Therefore, C/EBPα has to be altered externally to reprogram MegE lineage to GM lineage. More perturbations are necessary to accurately reach the monocytes lineage. Sustained activation of PU1 and the absence of C/EBPα guide the network to A_2, the expression of which differs with monocytes only in C/EBPα [KMST11].

Conversion from megakaryocytes to granulocytes

We take the conversion from megakaryocytes to granulocytes to illustrate the effects of integrating practical constraints on the perturbations and intermediate attractors.

AST control

```
PATH 1 - #perturbations: 3
Sequence of the attractors: megakaryocytes -> granulocytes
  STEP 1
    Control set 1: C/EBPA=1 PU1=1 Gfi1=1
    Control set 2: GATA1=0 C/EBPA=1 Gfi1=1

PATH 2 - #perturbations: 3
Sequence of the attractors: megakaryocytes -> A1 -> granulocytes
  STEP 1
    Control set 1: GATA1=0
  STEP 2
    Control set 1: C/EBPA=1 Gfi1=1

PATH 3 - #perturbations: 3
Sequence of the attractors: megakaryocytes -> A2 -> granulocytes
  STEP 1
    Control set 1: PU1=1
  STEP 2
    Control set 1: C/EBPA=1 Gfi1=1
    Control set 2: EgrNab=0 C/EBPA=1
    Control set 3: C/EBPA=1 cJun=0
    Control set 4: C/EBPA=1 PU1=0

PATH 4 - #perturbations: 3
Sequence of the attractors: megakaryocytes -> A2 -> monocytes ->
    granulocytes
  STEP 1
    Control set 1: PU1=1
  STEP 2
    Control set 1: C/EBPA=1
  STEP 3
    Control set 1: Gfi1=1
    Control set 2: EgrNab=0
    Control set 3: cJun=0
    Control set 4: PU1=0
```

Figure 5.7: AST control from megakaryocytes to granulocytes.

Constraints on perturbations. Let us assume node GATA1 should not be perturbed in either way (from 1 to 0 or from 0 to 1). Table 5.5 shows the results of one-step control with and without constraints on the perturbations. Without any constraints, OI, OT and OP all find solutions that require to perturb GATA1 (see the second column of Table 5.5). By setting GATA1 as an undesired node for perturbation, OT and OP find a minimal control satisfying the constraint. However, there does not exist any OI control without perturbing GATA1. The major principle of OI control is that the control should drive the network from megakaryocytes to the strong basin of granulocytes. Table 5.4 shows that GATA1 has a value of 1 in megakaryocytes, while GATA1 is 0 in any state in the strong basin of granulocytes as shown in Table 5.6. Hence, it is inevitable to flip the expression of GATA1 for OI control. When GATA1 is not allowed for intervention, OI control returns an empty set.

AST control with constraints on intermediate attractors

```
PATH 1 - #perturbations: 3
Sequence of the attractors: megakaryocytes -> granulocytes
  STEP 1
    Control set 1: Gfi1=1 C/EBPA=1 PU1=1
    Control set 2: Gfi1=1 GATA1=0 C/EBPA=1

PATH 2 - #perturbations: 3
Sequence of the attractors: megakaryocytes -> A2 -> granulocytes
  STEP 1
    Control set 1: PU1=1
  STEP 2
    Control set 1: Gfi1=1 C/EBPA=1
    Control set 2: EgrNab=0 C/EBPA=1
    Control set 3: cJun=0 C/EBPA=1
    Control set 4: C/EBPA=1 PU1=0

PATH 3 - #perturbations: 3
Sequence of the attractors: megakaryocytes -> A2 -> monocytes ->
    granulocytes
  STEP 1
    Control set 1: PU1=1
  STEP 2
    Control set 1: C/EBPA=1
  STEP 3
    Control set 1: Gfi1=1
    Control set 2: EgrNab=0
    Control set 3: cJun=0
    Control set 4: PU1=0
```

Figure 5.8: AST control from megakaryocytes to granulocytes with constraints on intermediate attractors.

Constraints on intermediate attractors. Note that attractor A_1 does not have a biological interpretation and mature erythrocytes in mammals do not have cell nucleus, therefore, we set these two attractors as undesired intermediate attractors for sequential control. We take AST control as the representative and show the results of AST control with and without constraints on the intermediate attractors in Figures 5.7 and 5.8. In the two figures, 'PATH 1' is the minimal OT control, which determines the value of the threshold k. Without any restrictions, there exist several AST control paths, corresponding to the following sequences of attractors: (1) megakaryocytes → A1 → granulocytes; (2) megakaryocytes → A2 → granulocytes and (3) megakaryocytes → A2 → monocytes → granulocytes. When A_1 and erythrocytes are set as undesired intermediates, AST control identify several paths corresponding to the following sequences of attractors: (1) megakaryocytes → A2 → granulocytes and (2) megakaryocytes → A2 → monocytes → granulocytes.

	HS	Apop1	Apop2	Apop3	Apop4	EMT1	EMT2	M1	M2
Metastasis	0	0	0	0	0	0	0	1	1
Migration	0	0	0	0	0	0	0	1	1
Invasion	0	0	0	0	0	0	0	1	1
EMT	0	0	0	0	0	1	1	1	1
Apoptosis	0	1	1	1	1	0	0	0	0
CellCycleArrest	0	1	1	1	1	1	1	1	1
ECMicroenv	0	0	0	1	1	0	0	1	1
DNAdamage	0	1	1	1	1	1	0	1	0
GF	0	0	0	0	0	1	1	1	1
TGFbeta	0	0	0	1	1	0	0	1	1
p21	0	1	1	1	1	0	0	0	0
CDH1	1	1	1	1	1	0	0	0	0
CDH2	0	0	0	0	0	1	1	1	1
VIM	0	0	0	0	0	1	1	1	1
TWIST1	0	0	0	0	0	1	1	1	1
SNAI1	0	0	0	0	0	1	1	1	1
SNAI2	0	0	0	0	0	1	1	1	1
ZEB1	0	0	0	0	0	1	1	1	1
ZEB2	0	0	0	0	0	1	1	1	1
AKT1	0	0	0	0	0	0	0	0	0
DKK1	0	0	0	0	0	0	0	1	1
CTNNB1	0	0	0	0	0	0	0	0	0
NICD	0	0	0	0	0	0	0	1	1
p63	0	0	1	0	1	0	0	0	0
p53	0	1	0	1	0	0	0	0	0
p73	0	0	1	0	1	0	0	0	0
miR200	0	1	1	1	1	0	0	0	0
miR203	0	1	0	1	0	0	0	0	0
miR34	0	0	0	0	0	0	0	0	0
AKT2	0	0	0	0	0	1	1	1	1
ERK	0	0	0	0	0	1	1	1	1
SMAD	0	0	0	0	0	0	0	1	1

Table 5.7: Attractors of the tumour network.

5.9.3 Control of the tumour network

The tumour network is constructed to study the role of individual mutations or their combinations that have influences on the development of metastasis [CMR+15]. This network has 32 nodes and 9 singleton attractors. The expressions of the attractors are given in Table 5.7. Among the attractors, the homeostatic state (HS) corresponds to the state where metastasis is inhibited by Cdh1 activity. Four of the attractors, Apop1, Apop2, Apop3 and Apop4, where DNAdamage is on and the growth factor (GF) is off, correspond to apoptosis states. Two of the attractors, EMT1 and EMT2, represent the EMT phenotype (the epithelial-mesenchymal transition). The other two attractors, M1 and M2, represent metastasis in the presence of GF.

	Control	Time (seconds)
OI	{ECMicroenv=1 DNAdamage=1 ZEB1=1 SNAI2=1 NICD=1}	0.076

OT	{ECMicroenv=1 DNAdamage=1 AKT1=1}
	{ECMicroenv=1 DNAdamage=1 AKT2=1}
		1.113

OP	{ECMicroenv=1 DNAdamage=1 AKT2=1}	1.342

Table 5.8: One-step control from HS to M1.

ASI		AST/ASP	
#p	Sequence of attractors	#p	Sequence of attractors
2	HS → M2 → M1	2	HS → M2 → M1
4	HS → M2 → EMT2 → M1	3	HS → EMT2 → M1
4	HS → M2 → EMT1 → M1	3	HS → EMT1 → M1
4	HS → M2 → EMT2 → EMT1 → M1	3	HS → EMT2 → EMT1 → M1
5	HS → EMT2 → M1	3	HS → EMT2 → M2 → M1
5	HS → EMT2 → EMT1 → M1		
5	HS → EMT2 → M2 → M1		
5	HS → Apop1 → M2 → M1		
5	HS → Apop2 → M2 → M1		
5	HS → Apop3 → M2 → M1		
5	HS → Apop4 → M2 → M1		
5	HS → Apop1 → Apop3 → M2 → M1		
5	HS → Apop2 → Apop4 → M2 → M1		

Table 5.9: Sequential control from HS to M1. '#p' represents the size of the control sets, namely the number of perturbations.

ASI		AST/ASP	
#p	Sequence of attractors	#p	Sequence of attractors
2	HS → M2 → M1	2	HS → M2 → M1
4	HS → M2 → EMT2 → M1	3	HS → EMT2 → M1
4	HS → M2 → EMT1 → M1	3	HS → EMT1 → M1
4	HS → M2 → EMT2 → EMT1 → M1	3	HS → EMT2 → EMT1 → M1
5	HS → EMT2 → M1	3	HS → EMT2 → M2 → M1
5	HS → EMT2 → EMT1 → M1		
5	HS → EMT2 → M2 → M1		

Table 5.10: Sequential control from HS to M1 with Apop1, Apop2, Apop3, and Apop4 as undesired attractors. '#p' represents the size of the control sets, namely the number of perturbations.

We compute the key interventions required for the conversion from HS to M1 with the six control methods. The results of one-step control, including detailed control sets and the computational time, are given in Table 5.8. OI requires at least five perturbations. OT and OP reduce the number of perturbations from five to three. Even though OT and OP require the same number of perturbations, OT finds two solutions while OP finds only one. In terms of the computational time, OT and OP took longer time than OI, because OT and OP may take several verification iterations before finding a valid solution.

The attractor-based sequential control methods use the minimal number of perturbations required by their corresponding one-step version as the threshold k and identify all the sequential paths within the threshold. ASI identifies 13 sequential paths, while AST and ASP find 23 identical control paths. For simplicity, we give the sequences of the attractors and the number of total perturbations rather than detailed control sets of the sequential control paths in Table 5.9. Note that there may exist different paths that return the same sequence of attractors. For instance, there are seven AST paths that adopt EMT2 as the intermediate states (HS $\rightarrow$ EMT2 $\rightarrow$ M1). These paths perturb different nodes to reprogram HS to EMT2.

In the results of ASI control, four attractors corresponding to apoptosis phenotypes are chosen to be the intermediate attractors in some paths. Although some studies showed that apoptotic cells can be rescued from the apoptosis process [GLSG01], it is recommended to bypass such states. Thus, we set the four apoptotic attractors (Apop1, Apop2, Apop3, and Apop4) as undesired attractors and recompute ASI, AST and ASP control. The results are given in Table 5.10. We can see that this condition does not affect the results of AST and ASP control as none of the AST/ASP control paths employs apoptotic attractors as intermediate attractors. For ASI control, this feature eliminates many unrealistic control paths.

5.9.4 Control of other biological networks

We apply our source-target control methods (OI, OT, OP, ASI, AST and ASP) to the other biological networks introduced in Section 2.4. The number of nodes, the number of edges and the number of attractors of each network have been given in Table 3.2 in Section 3.5. Since sequential control identifies the control paths with at most k perturbations, they may find solutions with different number of perturbations. For the purpose of illustration, we choose one pair of source and target attractors for each network and only summarise the minimal number of perturbations required by different control methods in Table 5.11.

Efficacy. Columns OI, OT and OP of Table 5.11 give the minimal number of perturbations required by the three one-step control methods. The results show that OI usually requires more perturbations than the other control methods. Even so, the number of perturbations required by OI is still relatively small compared to the size of the network. OT and OP can greatly reduce the number of perturbations compared to OI by extending the period of time for control. For all the cases listed in Table 5.11, OT and OP can achieve the goal with at most four perturbations. Moreover, OT has the potential to further reduce the number of perturbations compared to OP. For the bortezomib network and the Th Cell differentiation network (T-diff), OT requires three perturbations while OP needs at least four perturbations. The reason is that temporary control will be released at some time point, which empowers the control that might cause the absence of the target attractor during the control. This is not allowed for OP as the perturbations are applied for all the following time steps and thus the target attractor must be preserved in the transition system under control.

Network	The minimal number of perturbations					
	OI	OT	OP	ASI	AST	ASP
yeast	4	4	4	4	2	3
ERBB	9	3	3	8	3	3
HSPC-MSC	2	2	2	2	2	2
hematopoiesis	4	3	3	4	2	2
PC12	11	2	2	2	2	2
bladder	5	1	1	4	1	1
psc-bFA	6	3	3	5	2	3
co-infection	1	1	1	1	1	1
MAPK	14	2	2	11	2	2
CREB	3	3	3	3	3	3
HGF	14	2	2	3	2	2
bortezomib	21	3	4	14	3	4
T-diff	9	3	4	8	3	3
HIV1	3	3	3	3	3	3
CD4+	4	3	3	4	3	3
pathway	2	2	2	2	2	2

Table 5.11: The minimal number of perturbations required by the source-target control of the biological networks.

Network	Time (seconds)					
	OI	OT	OP	ASI	AST	ASP
yeast	0.002	0.015	0.014	0.072	0.825	0.896
ERBB	0.003	0.241	0.307	0.042	0.278	0.333
HSPC-MSC	0.071	0.075	0.079	0.086	0.098	0.109
hematopoiesis	0.031	0.238	0.446	0.467	28.902	25.595
PC12	0.01	0.466	0.403	0.235	1.787	2.173
bladder	0.033	0.064	0.065	0.296	0.192	0.295
psc-bFA	14.003	58.114	18.107	58.404	533.829	38.251
co-infection	0.320	0.509	0.552	0.333	2089.050	22970.800
MAPK	1.895	3.093	4.603	11.202	13.371	66.182
CREB	4.045	4.765	4.853	11.186	13.415	49.373
HGF	1.583	16.883	11.304	115.658	49.675	60.667
bortezomib	5.111	15.721	70.39	9.628	267.802	214.171
T-diff	2.759	8.245	10.117	18.877	224.646	271.329
HIV1	170.901	207.191	200.61	511.77	585.524	2547.94
CD4+	190.928	280.465	373.806	1079.74	1615.1	3582.46
pathway	335.129	2589.37	2590.74	394.117	2932.93	3524.83

Table 5.12: Computational time for the source-target control of the biological networks.

Columns ASI, AST and ASP of Table 5.11 give the minimal number of perturbations required by the three attractor-based sequential control methods. In a stepwise manner, sequential control can not only reduce the number of perturbations compared to one-step control, but also enrich the 'solution pool'. Fewer perturbations generally lead to lower experimental costs and a higher success rate for wet-lab experiments. Diverse reliable solutions provide biologists more options for experimental validation.

It is worth noting that for the CREB network, its attractors are purely induced by the input nodes. For such network, suppose it has m non-initialised input nodes, the number of attractors equals 2^m and the projection of the attractors to the input nodes includes all combination of binary strings of m bits. For any pair of source and target attractors, the input nodes that have different values in the source and the target are the essential control nodes.

Efficiency. Table 5.12 summarises the computational time for the six methods. All the control methods are based on the computation of the basins, thus, their efficiency strongly depends on the efficiency of the method for basin computation. The temporary and permanent controls (OT, OP, AST and ASP) have similar performance in terms of efficiency. In general, they are less efficient than the instantaneous control (OI and ASI) as it may take several iterations to find a valid control (see Algorithms 4 and 6). Sequential control takes longer time than one-step control as expected. Despite that, all the methods are still very efficient. For instance, for the $CD4^+$ T-cell network with 188 nodes, the computational time is 190.928, 280.465, 373.806, 1079.74, 1615.1 and 3582.46 seconds for OI, OT, OP, ASI, AST and ASP, respectively.

5.10 Conclusion

In this chapter, we have developed six methods for the source-target control of asynchronous Boolean networks, including OI, OT, OP, ASI, AST and ASP. These methods investigate various strategies to solve the source-target control problem. OI, OT and OP apply the control in one step, while ASI, AST and ASP drive the network from the source state to the target attractor through intermediate attractors (observable biological phenotypes) by applying a sequence of perturbations. Such a sequential strategy can provide novel reprogramming paths and may require fewer perturbations than one-step control.

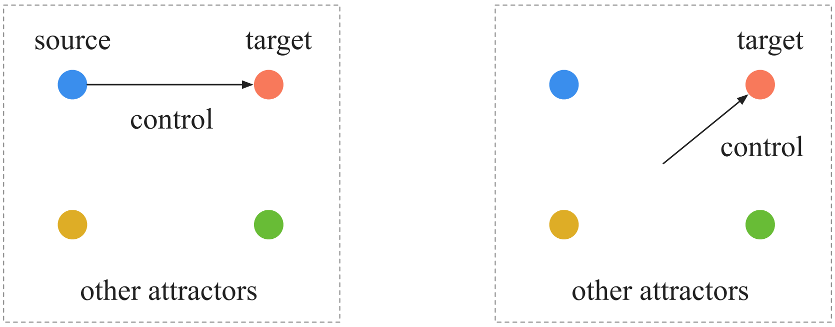
Temporary control and permanent control (OT, OP, AST, ASP) alter the dynamics of networks either for sufficient time or permanently, thus fewer perturbations are required to achieve the goal compared to instantaneous control (OI and ASI). Permanent control has a permanent influence on the dynamics of the system and thus is more invasive than instantaneous and temporary control. Hence, temporary control, like OT and AST, is more likely to serve better solutions than instantaneous and permanent control.

Chapter 6

Target Control

6.1 Introduction

In Chapter 5, we proposed six different control strategies for the source-target control of Boolean networks to drive the dynamics of a Boolean network from the source attractor to the target attractor. However, cells in tissues and in culture normally exist as a population of cells, corresponding to different states [dSB14]. There is a need of *target control* to compute a subset of nodes, whose perturbation can drive the network from any initial state to the desired target attractor. Figures 6.1 (a) and (b) illustrate the processes of source-target control and target control, respectively. The main difference lies in the source state: the source is a given attractor for source-target control, while the source can be any state in the state space for target control.



(a) Source-target control

(b) Target control

Figure 6.1: (a) Source-target control and (b) target control of Boolean networks. The red nodes represent the target attractor.

In this chapter, we study target control of asynchronous Boolean networks with instantaneous, temporary and permanent perturbations (ITC, TTC and PTC). We aim to find a control $\mathsf{C} = (\mathbb{0}, \mathbb{1})$, such that the instantaneous, temporary or permanent application of C – setting the value of a node, whose index is in $\mathbb{0}$ (or $\mathbb{1}$), to 0 (or 1) – can drive the network from any initial state s in the state space S to the target attractor A_t. Since the network can take any state $s \in S$ as the initial state, there exist a set of possible intermediate states with respect to C and they form a subset S' of S, called *schema*. As explained in Section 5.3, instantaneous control should drive the system to states in the strong basin of the target attractor. Thus, we partition the strong basin of the target attractor into a set of disjoint schemata. The support variables of each schema form an instantaneous target control. For temporary and permanent control, according to Sections 5.4 and 5.5, we know that all the intermediate states should fall into the weak basin of the target attractor. Therefore, we partition the weak basin of the target attractor into a set of mutually disjoint schemata. Each schema results in a candidate temporary or permanent target control, which will be minimised and verified.

Clinical applications are highly time-sensitive, controlling more nodes may shorten the period of time for generating sufficient desired cells for therapeutic application [GD19]. Hence, we integrate our method with a threshold ζ on the number of perturbations. By increasing ζ, we can obtain solutions with at most ζ perturbations. It is worth noting that more perturbations may cause a significant increase in experimental costs, hence, the threshold ζ should be considered individually based on specific experimental settings.

In Section 1.4, we have discussed existing works for the control of Boolean networks. Among these works, the stable motif-based control [ZnA15] also deals with the temporary target control problem. We apply our target control methods on a number of real-life biological networks and compare their performance with the stable motif-based control.

The rest of the chapter is organised as follows:

- In Section 6.2, we give the formal definition of the problems we are going to study in this chapter, viz. ITC, TTC and PTC.
- In Sections 6.3, 6.4 and 6.5, we introduce our methods for ITC, TTC and PTC based on the notion of schema, respectively.
- In Section 6.6, we apply our methods to a variety of biological networks and compare their performance with the stable motif-based control.

6.2 The target control problems

We have studied the one-step and sequential source-target control of Boolean networks in Chapter 5, to identify control paths that can drive the dynamics of the network from the source attractor to the target attractor. When the source is not given, to identify a subset of nodes, the control of which can stir the dynamics from any state $s \in S$ to the target attractor A_t, is called *target control* of Boolean networks. Because the source can be any state $s \in S$ rather than a given attractor, the definition of control C for target control is slightly different with the one for source-target control defined in Definition 15. A target control C is also a tuple $(\mathbb{0}, \mathbb{1})$, where $\mathbb{0}, \mathbb{1} \subseteq \{1, 2, \ldots, n\}$ and $\mathbb{0}$ and $\mathbb{1}$ are mutually disjoint (possibly empty) sets of indices of nodes of a Boolean network BN. Given a state $s \in S$, the application of C to s, denoted as $\mathsf{C}(s)$, is defined as a state $s' \in S$, such that $s'[i] = 0$ for $i \in \mathbb{0}$ and $s'[i] = 1$ for $i \in \mathbb{1}$. The target control can be lifted to a subset of states $S' \subseteq S$. Given a target control $\mathsf{C} = (\mathbb{0}, \mathbb{1})$, $\mathsf{C}(S') = S''$, where $S'' = \{s'' \in S | s'' = \mathsf{C}(s'), s' \in S'\}$. Set

S'' includes all the intermediate states with respect to C. The control C for source-target control and target control differs in the values of the control nodes in s and s'. For source-target control, a node x in C has different values in s and s'. That is, the application of a source-target control C always flips the values of the nodes in C, therefore, it is also called *toggle control*. Target control neglects the values of the nodes in s, the application of a target control C overexpresses the nodes in $\mathbb{1}$ and inhibits the nodes in $\mathbb{0}$, therefore, we also call it *fixed control*.

For target control, when perturbations are applied instantaneously, temporarily or permanently, we call them *instantaneous target control (ITC)*, *temporary target control (TTC)* or *permanent target control (PTC)*, respectively. Let BN be a given Boolean network, S be the set of states of BN and A_t be the target attractor of BN. We formally define the three target control problems as follows.

Definition 23 (Target control)**.**

1. **Instantaneous target control (ITC)**: *find a control* $\mathsf{C} = (\mathbb{0}, \mathbb{1})$ *such that the dynamics of* BN *always eventually reaches* A_t *on the instantaneous application of* C *to any initial state* $s, s \in S$.
2. **Temporary target control (TTC):** *find a control* $\mathsf{C} = (\mathbb{0}, \mathbb{1})$ *such that there exists a* $t_0 \geq 0$ *such that for all* $t \geq t_0$, *the dynamics of* BN *always eventually reaches* A_t *on the application of* C *to any initial state* $s, s \in S$ *for* t *steps.*
3. **Permanent target control (PTC):** *find a control* $\mathsf{C} = (\mathbb{0}, \mathbb{1})$ *such that the dynamics of* BN *always eventually reaches* A_t *on the permanent application of* C *to any initial state* $s, s \in S$. *(We assume implicitly that* A_t *is also an attractor of the transition system under control* $TS|_{\mathsf{C}}$*).*

We define the concept of *schema*, which is crucial for the development of the target control methods. Given a control $\mathsf{C} = (\mathbb{0}, \mathbb{1})$, the possible intermediate states with respect to C, denoted $S' = \mathsf{C}(S)$, form a schema, defined as follows.

Definition 24 (Schema)**.** *A subset* S' *of* S *is a schema if there exists a triple* $M = (\mathbb{0}, \mathbb{1}, \mathbb{D})$, *where* $\mathbb{0} \cup \mathbb{1} \cup \mathbb{D} = \{1, 2, \ldots, n\}$, $\mathbb{0}, \mathbb{1}$ *and* $\mathbb{D}$ *are mutually disjoint (possibly empty) sets of indices of nodes of* BN, *such that* $S'|_{\mathbb{0}} = \{0\}^{|\mathbb{0}|}$, $S'|_{\mathbb{1}} = \{1\}^{|\mathbb{1}|}$ *and* $S'|_{\mathbb{D}} = \{0, 1\}^{|\mathbb{D}|}$. $\mathbb{0}, \mathbb{1}$ *and* $\mathbb{D}$ *are called off-set, on-set and don't-care-set of* S', *respectively. The elements in* $\mathbb{0} \cup \mathbb{1}$ *are called indices of support variables of* S'.

Intuitively, for node $x_i, i \in \mathbb{0}$, it has a value of 0 in any state $s \in S'$; for node $x_i, i \in \mathbb{1}$, it has a value of 1 in any state $s \in S'$. The projection of S' to the don't-care-set $\mathbb{D}$ contains all combinations of binary strings of $|\mathbb{D}|$ bits. Thus, any schema S' is of size $2^{|\mathbb{D}|}$. Since the total number of nodes $n = |\mathbb{0}| + |\mathbb{1}| + |\mathbb{D}|$ is fixed, a larger schema implies more elements in $\mathbb{D}$ and fewer elements in $\mathbb{0} \cup \mathbb{1}$.

Example 12. *For attractor* A_1 *of* BN *given in Example 1, its strong basin,* $bas^S_{TS}(A_1) = \{000, 001\}$*, forms a schema. Let us denote the values of the nodes in off-set, on-set and don't-care-set as '0', '1' and '$*$', respectively.* $bas^S_{TS}(A_1)$ *can be represented as '00$*$'. The weak basin,* $bas^W_{TS}(A_1) = \{000, 001, 010, 011, 101, 100\}$, *can be partitioned into two schemata* $\{000, 001, 010, 011\}$ *and* $\{101, 100\}$, *represented as '0 $*$ $*$' and '10$*$', respectively.*

6.3 Instantaneous target control

According to Theorem 4, an instantaneous control C will surely guide the dynamics of BN from an initial state s to the target attractor A_t if the intermediate state $s' = \mathsf{C}(s)$ is in the strong basin of A_t in TS. Thus, when the initial state can be any state $s \in S$, to guarantee the inevitable reachability of the target attractor A_t on the instantaneous application of C to any $s \in S$, all possible intermediate states $S' = \mathsf{C}(S)$ must fall in the strong basin of the target attractor A_t. That is

Corollary 2. *A control* $\mathsf{C} = (\mathbb{0}, \mathbb{1})$ *is an instantaneous target control from any initial state* $s \in S$ *to a target attractor* A_t *iff* $\mathsf{C}(S) \subseteq bas^{S}_{TS}(A_t)$.

Instantaneous control is only applied instantaneously, thus, its impact on the transition system is transient. If the instantaneous control does not drive the dynamics directly to $bas^{S}_{TS}(A_t)$ but to any state $s' \in (S \setminus bas^{S}_{TS}(A_t))$, from s', there exist paths to other attractor $A, A \neq A_t$ based on the definition of strong basin. This does not ensure the inevitable reachability of the target attractor. Therefore, for an ITC C, its intermediate states $S' = \mathsf{C}(S)$ must form a subset of $bas^{S}_{TS}(A_t)$. For any possible intermediate state $s' \in S'$, $s'[i] = 0$ for $i \in \mathbb{0}$ and $s'[i] = 1$ for $i \in \mathbb{1}$, which indicate that $S'|_{\mathbb{0}} = \{0\}^{|\mathbb{0}|}$ and $S'|_{\mathbb{1}} = \{1\}^{|\mathbb{1}|}$. Let $\mathbb{D}$ denote the indices of the nodes that are not in $\mathbb{0}$ or $\mathbb{1}$. Then, $S'|_{\mathbb{D}} = \{0, 1\}^{|\mathbb{D}|}$ because the values of the nodes in the initial states $S|_{\mathbb{D}}$ stay unchanged. In another word

Observation 4. *If the initial state can be any state* $s \in S$, *for any control* $\mathsf{C} = (\mathbb{0}, \mathbb{1})$, *the set of intermediate states* $S' = \mathsf{C}(S)$ *forms a schema.*

The notion of schema sheds light on the computation of ITC. Each schema W_i of the strong basin of the target attractor, $bas^{S}_{TS}(A_t)$, returns a candidate target control $\mathsf{C}_i = (\mathbb{0}_i, \mathbb{1}_i)$, where $\mathbb{0}_i$ and $\mathbb{1}_i$ are the off-set and on-set of W_i. The size of control $|\mathsf{C}_i|$ equals $(n - \log_2 |W_i|)$, therefore, a larger schema results in a smaller control set. Thus, we can partition the strong basin of the target attractor, $bas^{S}_{TS}(A_t)$, into a set of mutually disjoint schemata $\mathcal{W} = \{W_1, W_2, \ldots, W_m\}$, such that $W_1 \cup W_2 \cup \ldots \cup W_m = bas^{S}_{TS}(A_t)$. Each $W_i \in \mathcal{W}$ is one of the largest schemata in $bas^{S}_{TS}(A_t) \setminus (W_1 \cup \ldots \cup W_{i-1})$ and the indices of its support variables in $\mathbb{0}_i$ and $\mathbb{1}_i$ form a candidate ITC $\mathsf{C}_i = (\mathbb{0}_i, \mathbb{1}_i)$. In C_i, the specified input nodes can be removed because input nodes do not have any predecessors and the values of the specified input nodes are fixed. For large networks, there may exist many valid control sets. To restrict the number and the size of solutions, we set a threshold ζ on the number of perturbations, keep ζ updated with the minimal size of the computed control sets, and only save the control sets with at most ζ perturbations. Algorithm 10 realises the above idea in pseudocode.

In this way, the computation of ITC turns into the computation of the strong basin of the target attractor and the computation of schemata. The computation of strong basins has been solved using procedure COMP_STRONG_BASIN_DECOMP in Algorithm 2. The computation of schemata is based on BDDs, a symbolic representation of large state space. The size of a BDD is determined by both the set of states being represented and the chosen ordering of the variables. In BDDs, a schema is

Algorithm 10 Instantaneous target control

```
procedure INSTANTANEOUS_TARGET_CONTROL($\mathsf{BN}, A_t$)
    initialise $\mathcal{L} := \emptyset$ to store computed control sets.
    $I, I^s, I^{ns}$ :=COMP_INPUT_NODES($G$)    // compute input nodes $I$, specified input nodes $I^s$ and non-specified input nodes $I^{ns}$.
    $\mathsf{SB}$ :=COMP_STRONG_BASIN_DECOMP($F, A_t$)    //strong basin of $A_t$ in $TS$
    $\mathcal{W}$ :=COMP_SCHEMATA($\mathsf{SB}$), $m := |\mathcal{W}|$
    $\zeta := n$    an initial threshold on the number of perturbations.
    for $i = 1 : m$ do    // traverse the set of schemata
        $\mathsf{C}_i$ :=COMP_SUPPORT_VARIABLES($W_i$)    // $\mathsf{C}_i := (\mathbb{0}_i, \mathbb{1}_i)$
        $\mathsf{C}_i := (\mathbb{0}_i \setminus I^s, \mathbb{1}_i \setminus I^s)$    // remove specified input nodes
        if $|\mathsf{C}_i| \leq \zeta$ then
            save $\mathsf{C}_i$ to $\mathcal{L}$
            $\zeta := min(|\mathsf{C}_i|, \zeta)$
        end if
    end for
    return $\mathcal{L}$
end procedure
```

represented as a *cube* and each state is the smallest cube, also called a *minterm*. To compute the largest schema S_i of S is equivalent to the computation of the largest cube of S. The partitioning of the strong basin into schemata is then transformed into a cube cover problem in BDDs. A different variable ordering may lead to a different partitioning. Given a fixed ordering, the partitioning remains the same. Although finding the best variable ordering is NP-hard, there exist efficient heuristics to find the optimal ordering. Here we compute a partitioning under one variable ordering as provided by the CUDD package [Som15] and we call this procedure COMP_SCHEMATA.

6.4 Temporary target control

In this section, we develop a method for TTC. First, we introduce the following corollary, which can be derived from Theorem 5.

Corollary 3. *A control* $\mathsf{C} = (\mathbb{0}, \mathbb{1})$ *is a temporary target control to a target attractor* A_t *from any initial state* $s \in S$ *iff* $bas^S_{TS}(A_t) \cap S|_{\mathsf{C}} \neq \emptyset$ *and* $\mathsf{C}(S) \subseteq bas^S_{TS|_{\mathsf{C}}}(bas^S_{TS}(A_t) \cap S|_{\mathsf{C}})$.

Here we give an intuitive explanation of Corollary 3. We know that the application of a control C results in a new Boolean network $\mathsf{BN}|_{\mathsf{C}}$ and the state space is restricted to $S|_{\mathsf{C}}$. To guarantee the inevitable reachability of A_t, by the time we release the control, the network has to reach a state s in the strong basin of A_t w.r.t. the original transition system TS, i.e. $bas^S_{TS}(A_t)$, from which there only exist paths to A_t. This requires the remaining strong basin in $S|_{\mathsf{C}}$, i.e. $(bas^S_{TS}(A_t) \cap S|_{\mathsf{C}})$, is a non-empty set; otherwise, it is not guaranteed to reach A_t. Furthermore, the condition $\mathsf{C}(S) \subseteq bas^S_{TS|_{\mathsf{C}}}(bas^S_{TS}(A_t) \cap S|_{\mathsf{C}})$ ensures any possible intermediate state $s' \in \mathsf{C}(S)$ is in the strong basin of the remaining strong basin $(bas^S_{TS}(A_t) \cap S|_{\mathsf{C}})$ in the transition system under control $TS|_{\mathsf{C}}$, so that the network will always evolve to the remaining

Algorithm 11 Temporary target control

```
procedure TEMPORARY_TARGET_CONTROL($\mathsf{BN}, A_t$)
    initialise $\mathcal{L} := \emptyset$ and $\Omega := \emptyset$ to store valid temporary control sets and the checked control sets, respectively.
    $I, I^{ns}$ :=COMP_INPUT_NODES($G$)    //compute input nodes I and non-specified input nodes $I^{ns}$.
    $\mathsf{SB}$ :=COMP_STRONG_BASIN_DECOMP($F, A_t$)    //strong basin of $A_t$ in TS
    $\mathsf{WB}$ :=COMP_WEAK_BASIN($F, A_t$)    //weak basin of $A_t$ in TS
    $\mathcal{W}$ :=COMP_SCHEMATA($\mathsf{WB}$), $m := |\mathcal{W}|$
    generate a vector $\Theta$ of length $m$ and set all the elements to false    // $\Theta[i]$ indicates if $W_i$ can be skipped or not.
    $\zeta := n$ // set an initial threshold on the number of perturbations. n is the size of the network.
    for $i = 1 : m$ do    // traverse the set of schemata
        if $\Theta[i] = true$, then continue
        $\mathsf{C}_i$ :=COMP_SUPPORT_VARIABLES($W_i$)    // $\mathsf{C}_i := (\mathbb{0}_i, \mathbb{1}_i)$
        $\mathsf{C}_i^e := (\mathbb{0}_i \cap I^{ns}, \mathbb{1}_i \cap I^{ns})$, $\mathsf{C}_i^r := (\mathbb{0}_i \setminus I, \mathbb{1}_i \setminus I)$    //essential control nodes and non-input nodes in $\mathsf{C}_i$
        $k := 0$, $isValid := false$
        while $isValid = false$ and $k \leq \min(\zeta - |\mathsf{C}_i^e|, |\mathsf{C}_i^r|)$ do
            $\mathsf{C}_i^{sub}$ :=COMP_SUBSETS($\mathsf{C}_i^r, k$)    //compute subsets of $\mathsf{C}_i^r$ of size k.
            for $\mathsf{C}_j^{sub} \in \mathsf{C}_i^{sub}$ do
                $\mathsf{C}_i^j := \mathsf{C}_j^{sub} \cup \mathsf{C}_i^e$, $\Phi := \mathsf{C}_i^j(S)$    // $\Phi$ represents the set of intermediate states w.r.t. $\mathsf{C}_i^j$.
                if $\mathsf{C}_i^j \notin \Omega$ then    // $\mathsf{C}_i$ has not been checked.
                    $isValid$ :=VERIFY_TTC($F, \mathsf{C}_i^j, \mathsf{SB}, \Phi$)
                    add $\mathsf{C}_i^j$ to $\Omega$.
                    if $isValid = true$ then
                        add $\mathsf{C}_i^j$ to $\mathcal{L}$, $\zeta := \min(\zeta, |\mathsf{C}_i^j|)$
                        $\Theta[z] := true$ if $W_z \subseteq \Phi$ for $z \in [i+1, m]$ // if a schema $W_z$ is a subset of $\Phi$, it will be skipped.
                    end if
                end if
            end for
            if $isValid = false$, then $k := k + 1$
        end while
    end for
    return $\mathcal{L}$
end procedure
```

strong basin. Once the network reaches the remaining strong basin, the control can be released and the network will evolve spontaneously towards the target attractor A_t. Based on the definition of the weak basin, it is sufficient to search the weak basin $bas_{TS}^{W}(A_t)$ for TTC.

A noteworthy point is that temporary control needs to be released once the network reaches a state in $(bas_{TS}^{S}(A_t) \cap S|_{\mathsf{C}})$. On one hand, although Corollary 3 guarantees that partial of the strong basin of A_t in TS is preserved in $TS|_{\mathsf{C}}$, it does not guarantee the presence of A_t in $TS|_{\mathsf{C}}$. In that case, the control C has to be released at one point to recover the original TS, which at the same time retrieves A_t. On the other hand, in clinic, it is preferable to eliminate human interventions to avoid unforeseen consequences. Concerning the timing to release the control, since it is hard to interpret theoretical time steps in diverse biological experiments, it would be more feasible for biologists to estimate the timing based on empirical knowledge and specific experimental settings.

Similar to ITC, the computation of TTC is also based on the concept of schema. Each schema W_i of the weak basin $bas^W_{TS}(A_t)$ gives a candidate TTC, $\mathsf{C}_i = (\mathbb{0}_i, \mathbb{1}_i)$, for further optimisation and validation. A larger schema results in a smaller control set. To explore the entire weak basin $bas^W_{TS}(A_t)$, we partition it into a set of mutually disjoint schemata $\mathcal{W} = \{W_1, W_2, \ldots, W_m\}$, $W_1 \cup W_2 \cup \ldots \cup W_m = bas^W_{TS}(A_t)$. Each $W_i, i \in m$ is one of the largest schemata in $bas^W_{TS}(A_t) \setminus (W_1 \cup \ldots \cup W_{i-1})$. For W_i, the indices of its support variables in $\mathbb{0}_i$ and $\mathbb{1}_i$ form a candidate control $\mathsf{C}_i = (\mathbb{0}_i, \mathbb{1}_i)$. Each candidate control C_i is primarily optimised based on the properties of input nodes. Because input nodes do not have any predecessors, it is reasonable to assume that specified input nodes I^s are redundant control nodes, while non-specified input nodes I^{ns} are essential for control. For the remaining non-input nodes in C_i, denoted C^r_i, we verify its subsets of size k based on Corollary 3 from $k = 0$ with an increment of 1, until we find a valid solution.

Algorithm 11 implements the above idea in pseudocode. It takes as inputs the Boolean network $\mathsf{BN} = (X, F)$ and the target attractor A_t. It first initialises two vectors $\mathcal{L}$ and Ω to store valid controls and the checked controls, respectively. (We use Ω to avoid duplicate control validations.) Then, it computes input nodes I and the non-specified input nodes I^{ns}, $I^{ns} \subseteq I$ (line 3). The weak basin WB and the strong basin SB of A_t of TS are computed using procedures COMP_WEAK_BASIN and COMP_STRONG_BASIN_DECOMP developed in Chapter 4 (lines 4-5). The weak basin WB is then partitioned into m mutually disjoint schemata with procedure COMP_SCHEMATA. Realisation of this procedure relies on the function to compute the largest cube provided by the CUDD package [Som15]. For each schema W_i, the indices of its support variables computed by procedure COMP_SUPPORT_VARIABLES form a candidate control C_i (line 11). The essential control nodes C^e_i of C_i consist of the non-specified input nodes and the non-input nodes in C_i constitute a set C^r_i for further optimisation (line 12). We search the subsets of C^r_i starting from size $k = 0$ with an increment of 1 and verify whether the union of a subset C^{sub}_j of C^r_i and the essential nodes C^e_i, namely $\mathsf{C}^j_i = \mathsf{C}^{sub}_j \cup \mathsf{C}^e_i$, is a valid temporary target control using procedure VERIFY_TTC in Algorithm 12. If C^j_i is valid, save it to $\mathcal{L}$. When all the subsets are traversed or a valid control has been found, we proceed to the next schema W_{i+1}. In the end, all the verified temporary target controls are returned.

The most time-consuming part of our method lies in the verification process. As shown in Algorithm 12, for each candidate control C, we need to reconstruct the associated transition relations $F|_{\mathsf{C}}$ and compute the strong basin of the remaining strong basin in $TS|_{\mathsf{C}}$, i.e. $bas^S_{TS|_{\mathsf{C}}}(SB|_{\mathsf{C}})$ (lines 6-8 of Algorithm 12). Even though we have developed an efficient method for the strong basin computation, the computational time of Algorithm 12 still increases when the network size grows. To improve

Algorithm 12 Verification of temporary target control

```
procedure VERIFY_TTC($F, \mathsf{C}, \mathsf{SB}, \Phi$)
    $isValid := false$
    if $\Phi \subseteq \mathsf{SB}$ then
        $isValid = true$
    else
        $\mathsf{SB}|_{\mathsf{C}}$ :=COMP_STATE_CONTROL($\mathsf{C}, \mathsf{SB}$) //compute the remaining strong basin w.r.t. $\mathsf{C}$ in $TS|_{\mathsf{C}}$
        $F|_{\mathsf{C}}$ :=COMP_FN_CONTROL($\mathsf{C}, F$)
        $bas^{S}_{TS|_{\mathsf{C}}}(SB|_{\mathsf{C}})$ :=COMP_STRONG_BASIN_DECOMP($F|_{\mathsf{C}}, \mathsf{SB}|_{\mathsf{C}}$)
        if $\Phi \subseteq bas^{S}_{TS|_{\mathsf{C}}}(SB|_{\mathsf{C}})$ then
            $isValid = true$
        end if
    end if
    return $isValid$
end procedure
```

Algorithm 13 Verification of permanent target control

```
procedure VERIFY_PTC($F, \mathsf{C}, A_t, \Phi$)
    $isValid := false$
    if $\Phi|_{\mathsf{C}} = A_t|_{\mathsf{C}}$ then
        $F|_{\mathsf{C}}$ :=COMP_FN_CONTROL($\mathsf{C}, F$)
        $bas^{S}_{TS|_{\mathsf{C}}}(A_t)$ :=COMP_STRONG_BASIN_DECOMP($F|_{\mathsf{C}}, A_t$)
        if $\Phi \subseteq bas^{S}_{TS|_{\mathsf{C}}}(A_t)$ then
            $isValid = true$
        end if
    end if
    return $isValid$
end procedure
```

the efficiency, we propose two heuristics: (1) skip a schema W_z (line 10 and 23 of Algorithm 11) if it is a subset of intermediate states Φ of a pre-validated control C_i^j (line 23 of Algorithm 11); and (2) set a threshold ζ on the number of perturbations, keep ζ updated with the smallest size of valid TTCs C_i^j (line 22 of Algorithm 11) and only compute control sets with at most ζ perturbations.

6.5 Permanent target control

In this section, we develop a method to solve the problem of PTC. We first introduce the following corollary derived from Theorem 4.

Corollary 4. *A control $\mathsf{C} = (\mathbb{0}, \mathbb{1})$ is a permanent target control from any initial state $s \in S$ to a target attractor A_t iff A_t is an attractor of $TS|_{\mathsf{C}}$ and $\mathsf{C}(S) \subseteq bas^{S}_{TS|_{\mathsf{C}}}(A_t)$.*

Different from temporary control, permanent control is applied for all the following time steps. Thus, a permanent control should preserve the target attractor. To guarantee the inevitable reachability of the target attractor, all possible intermediate

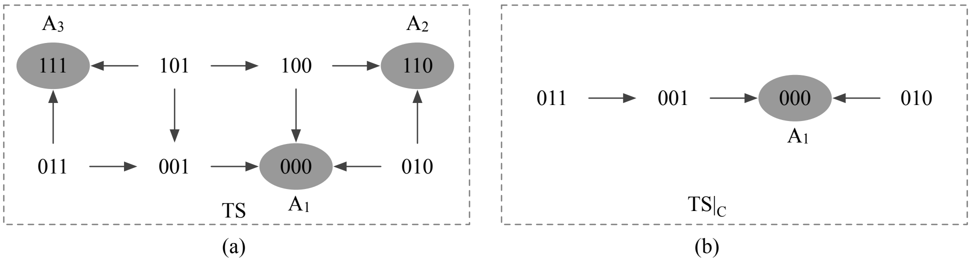


Figure 6.2: (a) The original transition system and (b) the transition system under control $\mathsf{C} = \{x_1 = 0\}$.

states should fall in the strong basin of the target attractor A_t in the transition system under control, $TS|_{\mathsf{C}}$.

The algorithm for PTC can be derived from Algorithm 11 by replacing procedure Verify_TTC with procedure Verify_PTC in Algorithm 13. To avoid duplication, we only explain procedure Verify_PTC here. This procedure is designed based on Corollary 4. Line 3 verifies whether the target attractor is preserved or not. If the target is preserved, we compute the transition relations under control $F|_{\mathsf{C}}$ and compute the strong basin of A_t in $TS|_{\mathsf{C}}$ (lines 4-5). C is a PTC if the set of intermediate states is a subset of $bas^S_{TS|_{\mathsf{C}}}(A_t)$ (lines 6-7).

Example 13. *In Example 12, we showed that the strong basin of A_1 of* BN *in Example 1, namely $bas^S_{TS}(A_1)$, can be represented as '$00*$'. It is easy to obtain the ITC for A_1, which is $\{x_1 = 0, x_2 = 0\}$. The simultaneous inhibition of x_1 and x_2 can drive the network from any state to* (000) *or* (001)*, such that the network will eventually reach* (000)*.*

The weak basin of A_1, $bas^W_{TS}(A_1)$, can be divided into two schemata, represented as '$0**$' and '$10*$'. '$0**$' contains more states than '$10*$', which implies that '$0**$' can potentially give a smaller TTC or PTC. Algorithms for TTC and PTC verify subsets of the control derived from '$0**$' and '$10*$'. Based on Corollary 3 and Corollary 4, $\{x_1 = 0\}$ is both a TTC and a PTC for A_1. By fixing x_1 to 0, the transition system changes from Figure 6.2 (a) to (b). The network is driven to a state in $TS|_{\mathsf{C}}$ (see Figure 6.2 (b)) and will eventually stable in A_1.

6.6 Evaluation

As discussed in the introduction of this chapter, our methods ITC, TTC and PTC and the stable motif-based control (SMC) [ZnA15] focus on target control of asynchronous Boolean networks. In particular, both TTC and SMC employ temporary perturbations. We apply our methods on the biological networks introduced in Section 2.4 and compare their performance with SMC. We discuss the results of the cardiac gene regulatory network, the myeloid differentiation network and the tumour network in Sections 6.6.1, 6.6.2 and 6.6.3 and give an overview of the results of the other networks in Section 6.6.4. Our target control methods, ITC, TTC and PTC, are implemented in our software CABEAN [SP20a]. SMC[1] is implemented in

---

[1] SMC is publicly available at https://github.com/jgtz/StableMotifs.

	SHF	FHF
ITC	{exogenCanWntI=1}	{canWnt=0 Foxc12=0 Tbx1=0 GATAs=0 Tbx5=1 exogenCanWntI=0 exogenCanWntII=0 }

TTC	{exogenCanWntI=1}	{GATAs=1 exogenCanWntI=0 }
		{Tbx5=1 exogenCanWntI=0}
		{Nkx25=1 exogenCanWntI=0}
		{Mesp1=1 exogenCanWntI=0}
		{Tbx1=1 exogenCanWntI=0}
		{Foxc12=1 exogenCanWntI=0}

PTC	{exogenCanWntI=1}	{GATAs=1 exogenCanWntI=0}
		{Nkx25=1 exogenCanWntI=0}
		{Tbx5=1 exogenCanWntI=0}

SMC	{exogenCanWntI=1}	{GATAs=1 exogenCanWntI=0 }
		{Tbx5=1 exogenCanWntI=0}
		{Nkx25=1 exogenCanWntI=0}

Table 6.1: Target control of the cardiac network.

Java. All the experiments are performed on a high-performance computing (HPC) platform, which contains CPUs of Intel Xeon Gold 6132 @2.6 GHz.

6.6.1 Control of the cardiac gene regulatory network

In Section 5.9.1, we analysed the source-target control of the cardiac gene regulatory network. We know that this network consists of three attractors, two of which correspond to FHF and SHF, when the input node, exogenBMP2I, is set to 1. Here we apply ITC, TTC, PTC and SMC to identify interventions that can lead the network to FHF and SHF. The results of the four control methods are given in Table 6.1.

With instantaneous, temporary or permanent perturbations, it is guaranteed to reach SHF by the control of the non-initialised input node, exogenCanWntI. To reach FHF, ITC requires seven perturbations, while TTC realises the goal by the control of two nodes, including the input node exogenCanWntI together with one of the nodes in {GATAs, Tbx5, Nkx25, Mesp1, Tbx1, Foxc12}. The results of PTC and SMC are subsets of TTC, which demonstrates the ability of TTC in identifying more novel solutions. With temporary and permanent perturbations, the number of perturbations required to reach FHF is reduced from seven to two, which can greatly reduce experimental costs and improve the operability of the experiments.

The total execution time of ITC, TTC, PTC and SMC for computing target control for the three attractors are 0.035, 0.128, 0.148 and 4.540 seconds, respectively. For this network, our methods are more efficient than SMC.

	granulocytes	monocytes	megakaryocytes	erythrocytes
ITC	6	7	4	4
TTC	3	3	2	2
PTC	3	3	2	2
SMC	4	4	2	2

Table 6.2: The number of perturbations required by ITC, TTC, PTC and SMC of the myeloid differentiation network.

	granulocytes	monocytes
ITC	{GATA2=0 GATA1=0 Fli1=0 EgrNab=0 C/EBPα=1 Gfi1=1}	{GATA2=0 GATA1=0 EgrNab=1 C/EBPα=1 PU1=1 cJun=1 Gfi1=0}

TTC	{C/EBPα=1 PU1=1 cJun=0}
	{EgrNab=0 C/EBPα=1 PU1=1}
	{C/EBPα=1 PU1=1 Gfi1=1}
		{EgrNab=1 C/EBPα=1 PU1=1}
		{C/EBPα=1 PU1=1 Gfi1=0}

PTC	{C/EBPα=1 PU1=1 cJun=0}
	{EgrNab=0 C/EBPα=1 PU1=1}
	{C/EBPα=1 PU1=1 Gfi1=1}
		{EgrNab=1 C/EBPα=1 PU1=1}
		{C/EBPα=1 PU1=1 Gfi1=0}

SMC	{GATA2=0 GATA1=0 C/EBPα=1 EgrNab=0}
	{GATA2=0 GATA1=0 C/EBPα=1 Gfi1=1}
	{GATA2=0 PU1=1 C/EBPα=1 EgrNab=0}
	{GATA2=0 PU1=1 C/EBPα=1 Gfi1=1}
		{GATA2=0 GATA1=0 C/EBPα=1 EgrNab=1}
		{GATA2=0 GATA1=0 C/EBPα=1 Gfi1=0}
		{GATA2=0 PU1=1 EgrNab=1 C/EBPα=1}
		{GATA2=0 PU1=1 Gfi1=0 C/EBPα=1}

Table 6.3: The control sets computed by ITC, TTC, PTC and SMC for granulocytes and monocytes of the myeloid differentiation network.

	megakaryocytes	erythrocytes
ITC	{GATA2=0 GATA1=0 EgrNab=1 C/EBPα=1 PU1=1 cJun=1 Gfi1=0}	{GATA1=1 EKLF=1 Fli1=0 PU1=0}

TTC	{GATA2=1 EKLF=0}
	{GATA1=1 EKLF=0}
	{GATA2=1 Fli1=1}
	{GATA1=1 Fli1=1}
	{Fli1=1 PU1=0 }
		{GATA2=1 EKLF=1}
		{GATA1=1 EKLF=1}
		{GATA2=1 Fli1=0}
		{GATA1=1 Fli1=0}

PTC	{GATA1=1 EKLF=0 }
	{GATA1=1 Fli1=1}
	{Fli1=1 PU1=0 }
		{GATA1=1 EKLF=1 }
		{GATA1=1 Fli1=0 }

SMC	{GATA1=1 EKLF=0}
	{GATA1=1 Fli1=1}
		{GATA1=1 EKLF=1}
		{GATA1=1 Fli1=0}

Table 6.4: The control sets computed by ITC, TTC, PTC and SMC for megakaryocytes and erythrocytes of the myeloid differentiation network.

6.6.2 Control of the myeloid differentiation network

In Section 5.9.2, we showed that four of the attractors of the myeloid differentiation network correspond to granulocytes, monocytes, megakaryocytes and erythrocytes. We apply ITC, TTC, PTC and SMC to identify interventions to reach the four cell types.

Table 6.2 gives the number of perturbations required by the four methods. The control with instantaneous perturbations (ITC) requires more perturbations than the control with temporary perturbations (TTC and SMC) and permanent perturbations (PTC) as expected. For granulocytes and monocytes, TTC and PTC find smaller control sets than SMC. For megakaryocytes and erythrocytes, TTC, PTC and SMC require the same number of perturbations.

Table 6.3 and Table 6.4 summarise the control sets identified by the four methods. For each attractor, ITC only finds one control set with more perturbations than the other methods. Although SMC finds more control sets than TTC for granulocytes and monocytes as shown in Table 6.3, SMC requires four perturbations while TTC and PTC need only three perturbations. Since our methods TTC and PTC only compute the results within the threshold, they may identify more solutions if we increase the threshold to four. For megakaryocytes and erythrocytes in Table 6.4, TTC, PTC and SMC require the same number of perturbations, but TTC provides more control sets than PTC and SMC. For this network, the results of PTC are either identical to TTC or just a subset of the solutions identified by TTC. Potentially, TTC is able to find smaller control sets than PTC, because it does not need to preserve the target attractor during the control. We will demonstrate this point in Section 6.6.4.

The total execution time of ITC, TTC, PTC and SMC for computing target control of this network are 0.003, 0.026, 0.03 and 8.178 seconds, respectively. We can see that our methods outperform SMC in efficiency.

6.6.3 Control of the tumour network

In Section 5.9.3, we discussed the source-target control of the tumour network. This network has nine singleton attractors corresponding to different cell fates. In this section, we apply ITC, TTC, PTC and SMC to compute the target control of this network. However, SMC was stuck in the identification of stable motifs and failed to finish the computation within twelve hours, because the number of cycles and/or SCCs in the expanded network of the tumour network is computationally intractable.

Table 6.5 summarises the number of perturbations and the computational time of ITC, TTC and PTC for each attractor. Similar to the results of the cardiac and myeloid networks, TTC and PTC can reduce the size of control sets to a great extent compared to ITC. With at most four perturbations, TTC and PTC can reprogram a cell from any initial state to the desired cell phenotype. For each attractor, the number of perturbations required by TTC and PTC is identical. The total execution time of ITC, TTC and PTC are 0.906, 20.368 and 17.580 seconds, respectively.

Attractor	The minimal number of perturbations			Time (seconds)		
	ITC	TTC	PTC	ITC	TTC	PTC
HS	11	3	3	0.043	0.741	0.883
Apop1	7	3	3	0.343	2.616	2.353
Apop2	15	4	4	0.041	2.463	2.869
Apop3	7	3	3	0.116	4.131	3.029
Apop4	15	4	4	0.053	7.831	5.008
EMT1	11	3	3	0.055	1.015	0.936
EMT2	10	3	3	0.042	0.386	0.591
M1	11	3	3	0.079	1.704	1.799
M2	2	2	2	0.1	0.12	0.11

Table 6.5: Target control of the tumour network.

Network	The minimal number of perturbations			
	ITC	TTC	PTC	SMC
yeast	10	5	5	5
ERBB	10	2	2	2
HSPC-MSC	2	2	2	2
hematopoiesis	5	3	3	∗
PC12	12	3	3	3
bladder	14	2	2	4
psc-bFA	11	1	2	∗
co-infection	19	5	5	7
MAPK	24	4	4	5
CREB	3	3	3	∗
HGF	22	4	4	∗
bortezomib	3	1	1	∗
T-diff	20	4	4	4
HIV1	3	3	3	∗
CD4+	7	3	3	3
pathway	2	2	2	2

Table 6.6: The minimal number of perturbations required by ITC, TTC, PTC and SMC for several biological networks.

6.6.4 Control of other biological networks

We apply the four target control methods, ITC, TTC, PTC and SMC, to the other networks introduced in Section 2.4, and evaluate their performance.

Efficacy. Table 6.6 summarises the minimal number of perturbations required by the four methods for one of the attractors of the networks. It is easy to observe that ITC requires more perturbations than TTC, PTC and SMC due to its instantaneous effect. ITC usually needs to control 10 to 20 nodes, whereas TTC, PTC and SMC can achieve the inevitable reachability of the target attractor with at most 7 perturbations. Moreover, it is hard to realise the simultaneous and instantaneous perturbation of a number of nodes, which makes the ITC less practical in applications. Thus, TTC, PTC and SMC, which employ temporary or permanent perturbations, are preferable than ITC. For the bladder cancer network and the MAPK network, TTC and PTC identify smaller control sets than SMC. Compared to PTC, TTC has the ability to further reduce the number of perturbations as demonstrated by the model of mouse embryonic stem cells (PSC-bFA) – the number of perturbations required by TTC and PTC are 1 and 2, respectively.

Both TTC and SMC solve the target control problem with temporary perturbations. To further compare these two methods, Figure 6.3 shows the number of solutions identified by the two methods. The x-axis lists the names of the networks and the y-axis denotes the number of control sets. Blue bars and grey bars represent the control sets that only appear in the results of TTC and SMC (TTC$\setminus$ (TTC $\cap$ SMC), SMC$\setminus$ (TTC $\cap$ SMC)), respectively. Green bars represent the intersections of the two methods. Equations above the bars ($|C| = k$) denote the number of nodes contained in the control set, i.e. the number of required perturbations.

Since neither of the methods guarantees the minimal control, they may find control sets of different sizes for one attractor. For comparison, we only consider the smallest control sets. In Figure 6.3, there is no grey bar because the solutions identified by SMC are either also found by TTC and thus belong to (TTC $\cap$ SMC), or require more perturbations than TTC. For the cell cycle network of fission yeast (yeast), the ERBB receptor-regulated G1/S transition protein network (ERBB), the HSPC-MSC network (HSPC-MSC) and the model of signalling pathways (pathway), there are only green bars, which means that the results of TTC and SMC are identical. For the bladder cancer network (bladder), the co-infection network (co-infection) and the MAPK network (MAPK), we can only see blue bars because TTC finds smaller control sets than SMC. SMC failed to finish the computation for several networks within twelve hours, including the network of hematopoietic cell specification (hematopoiesis), the model of mouse embryonic stem cells (PSC-bFA), the CREB network, the model for HGF-induced keratinocyte migration (HGF), the model of bortezomib responses (bortezomib) and the HIV-1 network networks (HIV1). For the PC12 cell differentiation network (PC12), the Th-cell differentiation network (T-diff) and the CD4+ T-cell network (CD4+), although TTC and SMC require the same number of perturbations, our method TTC has the capability to provide more unique solutions, which may give more flexibility for practical applications.

Efficiency. Table 6.7 summarises the computational time for computing the target control for all the attractors of the networks rather than the selected target attractor. The reason is that SMC computes the control for all the attractors in one-go by generating the stable motif diagram, in which different sequences of stable motifs lead to different attractors. SMC does not support the computation of target control for only one attractor. Hence, for ITC, TTC and PTC, we also take the total computational time for all the attractors of the networks for comparison.

We can see that ITC is the most efficient one, however, it requires more perturbations. TTC and PTC are more efficient than SMC for most of the cases. The efficiency of our methods are influenced by many factors, such as the network size, the density, the number of attractors and the number of required perturbations. For the co-infection network and the model of bortezomib responses, TTC and PTC are

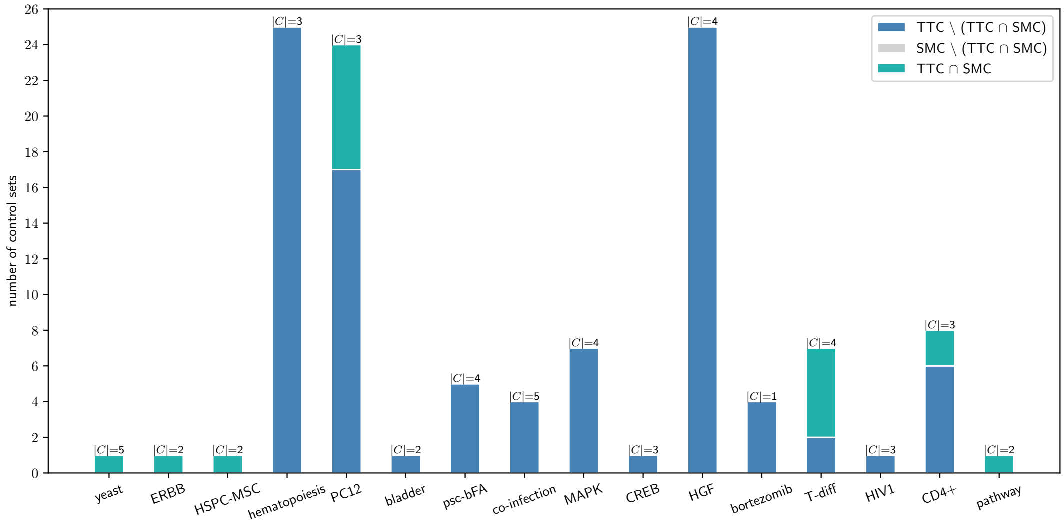


Figure 6.3: Comparison of TTC and SMC on the number of solutions.

Network	Time (seconds)			
	ITC	TTC	PTC	SMC
yeast	0.028	0.987	0.933	10.837
ERBB	0.055	0.117	0.163	6.400
HSPC-MSC	0.097	0.101	0.109	11.393
hematopoiesis	0.374	139.859	72.793	$*$
PC12	0.149	17.653	22.189	234.513
bladder	0.302	2.426	7.997	36.277
psc-bFA	36.77	3732.78	9296.740	$*$
co-infection	6294.29	$*$	$*$	15097.511
MAPK	4.608	22.218	45.504	395.014
CREB	7.962	8.277	8.693	$*$
HGF	19.925	1437.29	201.363	$*$
bortezomib	15.605	$*$	$*$	$*$
T-diff	21.581	29738.5	$*$	353.473
HIV1	302.8	323.666	379.127	$*$
CD4+	549.878	1982.45	21358.400	27.836
pathway	445.251	4435.59	10038.600	42.180

Table 6.7: Computational time of ITC, TTC, PTC and SMC for several biological networks. Symbol '*' means that the method failed to finish the computation within twelve hours.

able to identify target control efficiently for some of the attractors, but failed for the other attractors. One conjecture is that the target control of those attractors require many perturbations, such that it takes considerable time to verify the subsets of the schemata.

SMC failed to finish the computation for several networks within twelve hours, including the hematopoiesis, PSC-bFA, CREB, HGF, bortezomib and HIV1 networks. For the hematopoiesis network and the bortezomib network, SMC failed in the identification of stable motifs, which has been pointed out to be the most time-consuming part of SMC [ZnA15]. The reason could be that the number of cycles and/or SCCs in its expanded network is computationally intractable. For the HGF-induced keratinocyte migration network, SMC was blocked in the optimisation of stable motifs due to that this network has 19 stable motifs and most of the stable motifs contain more than 16 nodes. SMC failed to construct the expanded network representation for the CREB, PSC-BFA and HIV-1 networks because some of their Boolean functions depend on many parent nodes ($k \geq 10$). Detailed discussion on the complexity of SMC can be found in [ZnA15].

6.7 Conclusion

In this chapter, we have proposed three methods for the target control of asynchronous Boolean networks with instantaneous, temporary and permanent perturbations. We compared their performance with SMC [ZnA15] on various real-life biological networks. The results showed that ITC requires more perturbations than TTC, PTC and SMC as it uses instantaneous perturbations. Both TTC and SMC solve the target control problem with temporary perturbations and potentially they may require fewer perturbations than PTC. Moreover, compared to SMC, our method TTC has the potential to identify more solutions with fewer perturbations.

Regarding the computational time, our methods are quite efficient and scale well for large networks. SMC explores both structures and Boolean functions of Boolean networks, and is potentially more scalable for large networks as demonstrated by the CD4+ T-cell network and the pathway network in Table 6.7. In contrast, our methods are essentially based on the dynamics of the networks, and they will suffer the state space explosion problem for networks of several hundreds of nodes. We believe that our methods and SMC complement each other well.

Chapter 7

CABEAN: a Software for the Control of Asynchronous Boolean Networks

7.1 Introduction

Cell reprogramming is equivalent to the control problem in the framework of Boolean networks: finding a subset of nodes of the network, the perturbation of which can drive the dynamics of the network (from a source state) to the desired attractor. We have given a comprehensive discussion of the important control methods for complex networks in Section 1.4. Here we introduce several software tools for the analysis of logical models. ACTONETLIB [BD18] is a Mathematica library designed to compute driver nodes for Boolean control networks based on abductive reasoning. The caspo toolbox [VSRGS17] is a python package providing a workflow for the study of a family of logical networks of three-valued semantics under the synchronous updating scheme. It provides functions for learning and classification of the networks, design of experiments, and identification of intervention targets for reprogramming. CANA [CGWR18] is a Python package to study redundancy and control of synchronous Boolean networks of biochemical dynamics, however, it is not directly applicable to asynchronous Boolean networks. BoolNet [MHK10] is a powerful R package, which integrates methods for reconstruction, generalisation, and attractor identification for synchronous, asynchronous, and probabilistic Boolean networks. PyBoolNet [KSS17] is a python package for manipulating Boolean networks, such as generation, visualisation, and attractor detection. However, neither BoolNet nor PyBoolNet supports the identification of intervention targets for modulating the dynamics. The algorithm Kali [PG18] is proposed to predict intervention targets which can reduce the reachability of attractors associated with pathological phenotypes for both synchronous and asynchronous Boolean networks. It traverses the state space of a Boolean network by performing $\max_s$ random walks of $\max_k$ steps to find all the attractors while estimating their basins. Thus, Kali is limited to an estimation rather than an accurate computation of the attractors and their basins.

The main course of this thesis concentrates on the scalable control of asynchronous Boolean networks. So far, we have developed several methods for the source-target control (OI, OT, OP, ASI, AST, and ASP) and target control (ITC, TTC and PTC) of Boolean networks. All these methods are based on the computation of strong basins of attractors (Algorithm 2), which explores both the structural and dynamical properties of asynchronous Boolean networks [PSPM18, PSPM19]. An instantaneous control drives the network dynamics from the source state to a state in the strong basin of the target attractor, from which there only exist paths to the target attractor. Both temporary control and permanent control will hold the control for a different amount of time, thus they can make use of the spontaneous evolutions of the network dynamics by moving into the weak basin of the target attractor, from which there exist paths to the target attractor and may also exist paths to other attractors. To guarantee the inevitable reachability of the target attractor, a temporary control drives the network dynamics to a state in the strong basin of the target attractor at the end of control, while a permanent control stirs the network from the source state to a state in the strong basin of the target attractor in the resulting transition system under control.

In this chapter, we present our novel software, CABEAN[1] , integrating all the control methods introduced in this thesis for the control of asynchronous Boolean networks, including OI, OT, OP, ASI, AST, ASP, ITC, TTC and PTC. This chapter is organised as follows: we introduce the general features of CABEAN in Section 7.2, and then illustrate the basic usages of CABEAN with a toy example in Section 7.3. Detailed instructions on how to analyse a Boolean network with CABEAN can be found at the website of the software: https://satoss.uni.lu/software/CABEAN/.

7.2 General features

	Source-target Control		Target Control
	Minimal One-step Control	Attractor-based Sequential Control	
Instantaneous	✓	✓	✓
Temporary	✓	✓	✓
Permanent	✓	✓	✓

Table 7.1: Control methods integrated in CABEAN.

CABEAN is designed for the analysis of asynchronous Boolean networks. It implements the decomposition-based attractor detection method [MPQY19], which identifies all the exact attractors of a given Boolean network. After the attractor detection, users can specify the source attractor (for source-target control) and the target attractor to compute effective interventions for the conversion. As shown in Table 7.1, currently CABEAN implements several control methods that can modulate the dynamics in different ways, including the methods for the six source-target control strategies and three target control strategies. The minimal OI, OT, and OP

---

[1] CABEAN is freely available at https://satoss.uni.lu/software/CABEAN/.

compute the exact and minimal control sets, while ASI, AST, and ASP compute all the sequential control paths (including the shortest paths) with at most k perturbations. The thresholds for ASI, AST and ASP are set as the number of perturbations required by the minimal OI, OT and OP control methods, respectively. Although the three target control methods (ITC, TTC and PTC) do not ensure the minimality of the control results, the control sets they identified are quite small as demonstrated in Section 6.6. All the methods guarantee the inevitable reachability of the target attractor. Due to the high diversity of biological systems, there does not exist any criteria for selecting the 'best' control method. It is recommended to compute the results with all the provided control methods and select suitable candidates for wet-lab validation based on specific experimental settings.

Practical constraints have to be taken into consideration to make the results more feasible. Moreover, actual practicalities need to be investigated case by case. Some genes are difficult to perturb. For instance, GATA1 and GATA2 belong to a family of transcription factors that have different functionalities but have similar structural properties, thus, it can be difficult to distinguish these two by the 'perturbation tools'. Some genes are essential for cell survival, such as AKT1 and CTNNB1, which have critical responsibilities in maintaining normal functionalities of cells, thus, it should be prohibited to perturb such genes to 'not expressed'. In terms of perturbing a node from 'not expressed' to 'expressed', there are a number of tools that can achieve a successful overexpression of a previously non-expressed node. However, due to the abstraction of Boolean networks, some nodes of the network represent the entire pathways (e.g. canWNT, EgrNab) or a family of genes (e.g. GATAs, SMAD). For such cases, it is necessary to look into details to figure out the specific species to perturb, instead of a set of molecular players. For attractor-based sequential control, besides the constraints on the undesired perturbations, we should also consider the status of the intermediate attractors. As an abstraction of the real systems, in Boolean networks, some attractors correspond to diseased states. It is reasonable to assume that the cell death or severe dysfunctions should be avoided. To make the results realistic, CABEAN provides functions to encode practical constraints on perturbations and intermediate attractors as prerequisite conditions for source-target control methods, such that these undesired perturbations or attractors are avoided during the computation.

CABEAN is implemented in C and C++ based on the efficient model checker MCMAS [LQR17] to encode Boolean networks into BDDs. BDDs were introduced by Bryant [Bry85] to represent Boolean functions. Thanks to its advantage of memory efficiency, BDDs have been widely used in many model checking algorithms to alleviate the state space explosion problem. Most of the realisation of our control methods are based on efficient BDD operations. All the above mentioned factors contribute to a high efficiency of CABEAN.

7.3 Case study

In this section, we show the basic usages of CABEAN, including the syntax of model files, the syntax of specification files for encoding constraints, as well as the command options. We use the three-node Boolean network given in Example 1 for illustration.

Model files

CABEAN supports the BoolNet and ISPL (Interpreted Systems Programming Language) format of the software MCMAS [LQR17]. We recommend the BioLQM toolkits[2] for the conversion of other formats, such as SBML-qual, Petri net, GINsim, to the BoolNet format.

ISPL file

```
Agent M
        Vars:
                x1: boolean;
                x2: boolean;
                x3: boolean;
        end Vars
        Actions = {none};
        Protocol:
                Other: {none};
        end Protocol
        Evolution:
                x1=true if  x2=true;
                x1=false if  x2=false;
                x2=true if  x1=true;
                x2=false if  x1=false;
                x3=true if  x2&x3=true;
                x3=false if  x2&x3=false;
        end Evolution
end Agent
InitStates
        M.x1=true or M.x1=false;
end InitStates
```

Bnet file

```
targets, factors
 x1, x2
 x2, x1
 x3, x2&x3
```

The model file of the example that follows the ISPL format is given above and saved as 'toy.ispl'. It contains two sections: section 'Agent M' and section 'InitStates'. Section 'Agent M' defines the Boolean variables and the Boolean functions of the network; section 'InitStates' specifies the initial state. Section 'Agent M' includes four parts: 'Vars', 'Actions', 'Protocol', and 'Evolution'. The 'Vars' part defines the Boolean variables and the order of the variables is consistent with the order of the nodes in each state in the output. The 'Evolution' part specifies the Boolean functions for each variable. The functions are defined using parenthesis and three

---

[2] BioLQM is available at http://colomoto.org/biolqm/.

logical operations, including logical and '&', logical or '|', and logical not '∼'. The 'Actions' and 'Protocol' parts are defined as 'none'.

BoolNet describes a Boolean network or probabilistic Boolean network in a standardized text file format [MHK10]. Here we only explain the syntax of BoolNet for Boolean networks. The model file of the example under BoolNet format, saved as 'toy.bnet', is also given at the previous page. The first line is a header: 'targets, factors'. The name and the Boolean function of each node are given at each line, separated by comma. The header implies the format: 'targets' is the name of the node and 'factors' represents the Boolean function of the node. In the Boolean functions, the logical operators and, or and not are represented as '&', '|' and '!', respectively. Users can initialise the value of an input node to either '1' or '0'.

Specification files
CABEAN allows users to encode three kinds of undesired perturbations:

- R_0: nodes that cannot be perturbed from 'not expressed' to 'expressed';
- R_1: nodes that cannot be perturbed from 'expressed' to 'not expressed';
- R: nodes that cannot be perturbed in any direction.

In the specification file for defining undesired perturbations, we list the nodes of each kind of perturbations and separate them by comma. Let us assume $R_0 = \{x_1, x_3\}$, $R_1 = \emptyset$, and $R = \{x_2\}$. The specification file is written as follows.

Specification file for undesired perturbations

```
R0: x1,x3
R1:
R: x2
```

In the specification file for undesired attractors, the indices of the attractors are separated by comma. Given a network with six attractors sorted in lexicographic order, suppose the third and the fourth attractors are undesired attractors, the specification file is given below.

Specification file for undesired attractors

```
3,4
```

Attractor detection
Prior to the computation of control, attractors of the example ('toy.ispl') are computed with the following command line:

Command for attractor detection

```
./cabean -compositional 2 toy.ispl
```

Option ’-compositional 2’ implies that the decomposition-based method [MPQY19] is used for attractor detection. CABEAN computes all the exact attractors of the network and prints them in lexicographic order. The output is given below.

Output of attractor detection

```
Command line: ./cabean -compositional 2 toy.ispl
====== find attractor #1 : 1 states ======
: 4 nodes 1 leaves 1 minterms
0-0-0-  1

====== find attractor #2 : 1 states ======
: 4 nodes 1 leaves 1 minterms
1-1-0-  1

====== find attractor #3 : 1 states ======
: 4 nodes 1 leaves 1 minterms
1-1-1-  1

number of attractors = 3
time for attractor detection=0.001 seconds
```

This network has three attractors. In each state, the sequence of the nodes in the expression is consistent with the sequence of nodes listed in section ‘Vars’ in the model file, which is ‘x1’, ‘x2’ and ‘x3’ for this case. The transition system of the network under the asynchronous updating scheme is given in Figure 2.1(c). We can see that attractors #1, #2 and #3 identified by our methods correspond to A_1, A_2 and A_3 in Figure 2.1(c). From any initial state, the network will eventually settle down to one of the attractors.

One-step source-target control
We take the minimal OT control from attractor $A_1 = \{000\}$ to attractor $A_3 = \{111\}$ as an example to show how to compute the control with and without constraints on perturbations. The minimal OT control without any constraints is computed with the following command line:

Command for the minimal OT control

```
./cabean -compositional 2 -control OT -sin 1 -tin 3 toy.ispl
```

Option ’-compositional 2’ indicates that the decomposition-based methods [MPQY19, PSPM18] are used for attractor detection and the computation of strong basins. Option ‘-control <control>’ specifies the control method to apply. Options ‘-sin <index of the source> -tin <index of the target>’ set the indices of the source and target attractors, which are 1 and 3, respectively. CABEAN first identifies all the exact attractors of the network and then compute the control for the specified source and target attractors. Once this is clear, we omit the attractors in the output and only give the results of the control sets as shown below. CABEAN identifies two minimal

OT controls: {x2=1 x3=1} and {x1=1 x3=1}.

Output of the minimal OT control

```
====== ONE-STEP TEMPORARY SOURCE-TARGET CONTROL (DECOMP) ======
source - 1 target - 3
PATH 1 - #perturbations: 2
        Control set: x2=1 x3=1
PATH 2 - #perturbations: 2
        Control set: x1=1 x3=1
execution time for control = 0.001 seconds
```

Assume node x2 cannot be perturbed, the specification file for encoding this restriction, saved as ‘undesiredPert.txt’, is given below:

Specification file for undesired perturbations

```
R0:
R1:
R: x2
```

We add option ′-rmPert <file name>′ to the command line as follows to compute the minimal OT control from A_1 to A_3 without perturbing x_2.

Command for the minimal OT control with constraints on perturbations

```
./cabean -compositional 2 -rmPert undesiredPert.txt -control OT -sin 1
    -tin 3 toy.ispl
```

The output is given below. There exists only one minimal OT control satisfying the constraint.

Output of the minimal OT control with constraints on perturbations

```
====== ONE-STEP TEMPORARY SOURCE-TARGET CONTROL (DECOMP) ======
source - 1 target - 3
PATH 1 - #perturbations: 2
        Control set: x1=1 x3=1
execution time for control = 0.004 seconds
```

Attractor-based sequential source-target control

We take AST control from attractor $A_1 = \{000\}$ to attractor $A_3 = \{111\}$ as an example to show how to compute sequential source-target control with and without constraints on intermediate attractors. CABEAN takes the number of perturbations required by the minimal OT control as the threshold θ for AST control and computes all the AST control paths with at most θ perturbations. The AST control without any constraints is computed with the following command line:

Command for AST control

```
./cabean -compositional 2 -control AST -sin 1 -tin 3 toy.ispl
```

There exists two control paths. The first path is an OT path from A_1 to A_3 (Sequence of the attractors: 1 → 3), which can be realised by two control sets, denoted as $A_1 \xrightarrow[x1=1,x3=1]{x2=1,x3=1} A_3$. The second path is a sequential path from A_1 to A_3 through the intermediate attractor A_2, denoted as $A_1 \xrightarrow[x1=1]{x2=1} A_2 \xrightarrow{x3=1} A_3$. All the control paths require two perturbations.

Output of AST control

```
==ATTRACTOR-BASED SEQUENTIAL TEMPORARY SOURCE-TARGET CONTROL (DECOMP)==
source - 1 target - 3
PATH 1 - #perturbations: 2
Sequence of the attractors: 1 -> 3
        STEP 1
                Control set 1: x2=1 x3=1
                Control set 2: x1=1 x3=1

PATH 2 - #perturbations: 2
Sequence of the attractors: 1 -> 2 -> 3
        STEP 1
                Control set 1: x2=1
                Control set 2: x1=1
        STEP 2
                Control set 1: x3=1
execution time of control=0.003 seconds
```

Suppose attractor A_2 is the undesired intermediate attractor, the specification file, saved as ‘rmAtt.txt’, can be written as follows.

Specification file for undesired attractors

```
2
```

Now we add the constraint on intermediate attractors by inserting ’-rmID <file name> to the command line. The output is given at the next page. The sequential paths passing though the undesired attractors are eliminated from the results.

Command for AST control with constraints on intermediate attractors

```
./cabean -compositional 2 -rmID rmAtt.txt -control AST -sin 1 -tin 3
    toy.ispl
```

Output of AST control with constraints on intermediate attractors

```
==ATTRACTOR-BASED SEQUENTIAL TEMPORARY SOURCE-TARGET CONTROL (DECOMP)==
Indices of undesired attractors: 2
source - 1 target - 3
PATH 1 - #perturbations: 2
Sequence of the attractors: 1 -> 3
        STEP 1
                Control set 1: x2=1 x3=1
                Control set 2: x1=1 x3=1
execution time of control=0.001 seconds
```

Target control
Taking TTC as the representative, TTC of attractor A_3 can be computed with the following command line:

Command for TTC

```
./cabean -compositional 2 -control TTC -tin 3 toy.ispl
```

Option '-control TTC' selects the TTC method and option '-tin 3' sets A_3 as the target attractor. The results show that the network can reach A_3 from any initial state with the control of {x1=1 x3=1} or {x2=1 x3=1}.

Output of TTC control

```
====== TEMPORARY TARGET COTNROL (DECOMP) ======
***********************************************
TARGET ATTRACTOR #3
***********************************************
Control set 1: x1=1 x3=1
Control set 2: x2=1 x3=1

Time for temporary target control = 0.002 seconds
```

Chapter 8

Conclusions and Future Work

8.1 Limitations of existing biological networks

Direct cell reprogramming has opened up an unprecedented opportunity for tissue engineering and regenerative medicine. Throughout the thesis, we have introduced several methods to identify intervention targets for direct cell reprogramming via the control of asynchronous Boolean networks.

Even though the dynamics of asynchronous Boolean networks are non-deterministic, our methods guarantee to find solutions with 100% success rate *in silico*. Experimental validation is necessary to confirm their therapeutic efficacy *in vivo*. It is worth noting that the consistency of their efficacy *in silico* and *in vivo* highly relies on the quality of the Boolean network being used. That is, the upper bound on the performance of our methods are determined by the underlying network. The identified intervention targets can effectively modulate the dynamics as expected, provided that the adopted network well captures the structural and dynamical properties of the real-life biological system. We cannot expect that a control method can make predictions out of the scope of the network.

Our methods are designed for Boolean networks that model diverse biological systems, but essentially network inference itself is a fundamental challenge in systems biology [BK18]. During the analysis of existing networks, we spotted some flaws that should be paid attention to during the network inference, summarised below.

1. Simulation is often used to evaluate the stable behaviour of dynamics and to cross-validate the *in silico* perturbations with related studies in most of the works. However, simulation is less likely to cover the entire transition system, which is exponential in the size of the network. As a consequence, the information on attractors is usually incomplete or even missing, especially for networks of medium or large sizes. Without a complete knowledge of the attractors, the inferred networks might have a considerable number of attractors and/or huge attractors, that do not appear in real systems. The attractor detection method based on our near-optimal decomposition can solve this problem by identifying all the exact attractors within a reasonable time. This also makes it possible to study biologically interpretations of the identified attractors, which plays an essential role in making the reprogramming or control meaningful.

2. We noticed that attractors of some large networks are purely induced by input nodes. For instance, suppose a network with 2 input nodes (nodes without upstream regulators) has 2^2 attractors. Each attractor corresponds to one combination of the input nodes $(00, 01, 10, 11)$. For such networks, the input nodes, that have different values in the source and target attractors, are intuitively the key nodes that should be perturbed for reprogramming the dynamics.

3. In some networks, cell phenotypes or cell fates, such as apoptosis, proliferation and differentiation, are represented as marker nodes. Benefiting from this, we can classify attractors by only looking at the expressions of those nodes. However, a problem we often encounter is that there does not exist any reprogramming paths without perturbing these marker nodes. Because our control methods focus on the original attractors of the Boolean network, we conjecture that these networks are more suitable for the control problems that are not restricted to the original attractors. In this way, the marker nodes can be used to define desired properties and the purpose of the control is to identify interventions that can drive the network to the original or newly generated attractors that meet the desired properties.

The performance of inferred networks is usually assessed based on the structural accuracy and dynamics accuracy [BK18]. Specifically, the dynamics accuracy is computed by comparing the trajectories generated by the inferred network and the observed time-series data for all the nodes. The network inference is not a one-off step. One often needs to repeatedly come back to refine the model. The limitations we identified indicate that the inferred networks can be refined with the aid of our methods for attractor detection [MPQY19] and control [PSPM18, PSPM19, SPP19b, MSH+19, MSP+19]. Our methods provide accurate information of the networks, which can be used to polish up the networks by updating the Boolean functions or adding/deleting regulators. This is mutually beneficial to both the inference and the analysis of biological systems.

8.2 Conclusions

In this thesis, we perform a comprehensive study on the scalable control of asynchronous Boolean networks. More specifically, given a Boolean network, we compute a subset of nodes of the network, whose perturbation can drive the dynamics from a given source state or any source state to the desired attractor.

Prior to the development of the control methods, we worked on the identification of attractors and the computation of strong basins. We developed the near-optimal decomposition of Boolean networks to improve the efficiency of the decomposition-based attractor detection [MPQY19]. After that, we proposed the decomposition-based method to compute the strong basin of an attractor in a block-wise manner. These two methods pave the way for the development of the control methods.

We worked on bridging the gap between computational control methods and practical applications from three perspectives.

First, we explored three kinds of perturbations: instantaneous, temporary and permanent perturbations. All of them are feasible to conduct in biological experiments and each type of perturbations has its own advantages and disadvantages. Instantaneous perturbations only require instantaneous applications, which are the least invasive to the dynamics, at the cost that a relatively larger number of perturbations is needed to reach the target. Thanks to the extended effects on the network dynamics, temporary and permanent perturbations can achieve the goal with much fewer perturbations than instantaneous perturbations. Nevertheless, permanent perturbations should be treated with care due to their permanent influences on the network dynamics.

Second, we investigated different strategies for reprogramming the dynamics of networks. Remarkably, we developed six source-target control methods and three target control methods. Among these methods, attractor-based sequential source-target control brings all the attractors of the network into play to lead the network from the source attractor to the target attractor though other attractors. Because there is no universal standard of 'good' strategies for the control of Boolean networks, the current recommendation is to compute all the control paths with available control methods for a specific task. Various sets of identified targets serve as candidates, such that biologists can choose appropriate targets, the modulation of which will not disrupt physiological functions of biological systems.

Third, the evaluation results confirm that our methods can find a small subset of nodes to fulfil the control purpose. Moreover, the identified perturbations appear consistent with empirical knowledge. This is beneficial to practical applications, as fewer perturbations can tremendously decrease experimental costs and improve the practicality of experiments. Regardless, controlling more nodes may shorten the time for reaching the target attractors, i.e. generating sufficient desired cells [GD19]. To address this, we integrate our methods with a threshold on the number of perturbations, such that when the threshold is defined, all the solutions within the threshold will be returned.

Overall, we believe that our works have established a solid foundation for the analysis of asynchronous Boolean networks. They can provide deep insights into regulatory mechanisms of biological processes and promote the application of direct cell reprogramming.

8.3 Future work

"Our imagination is the only limit to what we can hope to have in the future."

Charles F. Kettering

Although we consider this thesis has achieved its goal, by looking back critically, we summarise some works that could be investigated for future work.

One of the greatest bottlenecks of our methods is scalability. Even though we can substantially reduce the network size by focusing on essential nodes for the biological process under study, by extending the scalability of our methods, we will be able to analyse more extensive networks that cover more relevant nodes. Currently, our methods make use of the structural information in the computation of strong basins. Except for that, they are merely based on the network dynamics. Owing to the state space explosion problem, we conjecture that our methods are not powerful enough to analyse real-life biological networks with thousands of nodes. Recent studies on positive and negative cycles of the dependency graphs provide a hint on how to accelerate the efficiency of our methods [Ric19].

Besides the high-complexity, uncertainty is also an important factor that impedes the mathematical modelling of biological systems. Uncertainty may be induced by experimental observations or by the biological system itself intrinsically or extrinsically [GGC16]. For instance, even though the GRN is identical in all the cells in the human body, different parts of the network may be expressed in different cell types [SD16]. Moreover, structural changes in a GRN might occur over time and conditions [MDCR$^+$16]. Thus, only looking at Boolean networks is insufficient. It is essential to move on to more complicated networks, such as Bayesian networks and probabilistic Boolean networks that incorporate uncertainty.

Various biological networks are inferred from different types of experimental data to describe specific biological processes. Common types of biological networks include GRNs, signalling networks, metabolic networks, neuronal networks and protein-protein interaction networks. Network-based computational methods can reveal novel features of various biological systems, identify drug targets and provide a good understanding of the mechanism-of-action of interventions from a systematic perspective. However, they usually deal with homogeneous networks, which contain one type of nodes or edges. In real-life biological systems, there exist various biological entities and interactions [GJFM18]. There is a surge for the study of heterogeneous networks, which will lead to a deeper and more complete understanding of cellular functions.

Further, to handle intrinsically large and complex biological networks, network-based computational methods require significant computational power and network inference requires large quantities of biomedical data, which is often unavailable [MKH$^+$19]. Recently, emerging methods based on biomedical data, such as machine learning and deep learning methods, explore features in accessible datasets and have high predictive power for the identification of intervention targets [MKH$^+$19]. Such methods can be applied to heterogeneous data to uncover diverse information, which is another way of addressing the limitation of network-based computational methods mentioned in the previous paragraph. Future studies on such data-driven methodologies could be quite beneficial to drug discovery and clinical applications.

Bibliography

[AB02] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47, 2002.

[Aku18] Tatsuya Akutsu. *Algorithms for Analysis, Inference, and Control of Boolean Networks*. World Scientific, 2018.

[ATE08] Ali Abdi, Mehdi Baradaran Tahoori, and Effat S. Emamian. Fault diagnosis engineering of digital circuits can identify vulnerable molecules in complex cellular pathways. *Science Signaling*, 1(42):ra10–ra10, 2008.

[BBAIDB07] Mukesh Bansal, Vincenzo Belcastro, Alberto Ambesi-Impiombato, and Diego Di Bernardo. How to infer gene networks from expression profiles. *Molecular Systems Biology*, 3(1):78, 2007.

[BD18] 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.

[BGMN20] Roberto Barbuti, Roberta Gori, Paolo Milazzo, and Lucia Nasti. A survey of gene regulatory networks modelling methods: from differential equations, to Boolean and qualitative bioinspired models. *Journal of Membrane Computing*, pages 1–20, 2020.

[BK18] Shohag Barman and Yung-Keun Kwon. A Boolean network inference from time-series gene expression data using a genetic algorithm. *Bioinformatics*, 34(17):i927–i933, 2018.

[Bor05] Stefan Bornholdt. Less is more in modeling large genetic networks. *Science*, 310(5747):449–451, 2005.

[BPSP19] Alexis Baudin, Soumya Paul, Cui Su, and Jun Pang. Controlling large Boolean networks with single-step perturbations. *Bioinformatics*, 35(14):i558–i567, 2019.

[Bry85] Randal E. Bryant. Symbolic verification of mos circuits. pages 419–438, 1985.

[CGC$^+$16] Eugen Czeizler, Cristian Gratie, Wu Kai Chiu, Krishna Kanhaiya, and Ion Petre. Target controllability of linear networks. In *Proc. 14th International Conference on Computational Methods in Systems Biology*, volume 9859 of *LNCS*, pages 67–81. Springer, 2016.

[CGWR18] Rion B. Correia, Alexander J. Gates, Xuan Wang, and Luis M. Rocha. CANA: a python package for quantifying control and canalization in Boolean networks. *Frontiers in Physiology*, 9:1046, 2018.

[CHS+14] Brittany D. Conroy, Tyler A. Herek, Timothy D. Shew, Matthew Latner, Joshua J. Larson, Laura Allen, Paul H. Davis, Tomáš Helikar, and Christine E. Cutucache. Design, assessment, and in vivo evaluation of a computational model illustrating the role of CAV1 in CD4+ T-lymphocytes. *Frontiers in Immunology*, 5:599, 2014.

[CKM13] Sean P. Cornelius, William L. Kath, and Adilson E. Motter. Realistic control of network dynamics. *Nature Communications*, 4(1):1–9, 2013.

[CLB94] Jason Cong, Zheng Li, and Rajive Bagrodia. Acyclic multi-way partitioning of Boolean networks. In *Proc. 31st Annual Design Automation Conference*, pages 670–675. IEEE, 1994.

[CLL+14] Lian En Chai, Swee Kuan Loh, Swee Thing Low, Mohd Saberi Mohamad, Safaai Deris, and Zalmiyah Zakaria. A review on the computational approaches for gene regulatory network construction. *Computers in Biology and Medicine*, 48:55–65, 2014.

[CLW16] Hongwei Chen, Jinling Liang, and Zidong Wang. Pinning controllability of autonomous Boolean control networks. *Science China Information Sciences*, 59(7):070107, 2016.

[CMR+15] David P.A. Cohen, Loredana Martignetti, Sylvie Robine, Emmanuel Barillot, Andrei Zinovyev, and Laurence Calzone. Mathematical modelling of molecular pathways enabling tumour cell invasion and migration. *PLOS Computational Biology*, 11(11):e1004571, 2015.

[COAM15] Vaishali L. Chudasama, Meric A. Ovacik, Darrell R. Abernethy, and Donald E. Mager. Logic-based and cellular pharmacodynamic modeling of bortezomib responses in U266 human myeloma cells. *Journal of Pharmacology and Experimental Therapeutics*, 354(3):448–458, 2015.

[CvOO+17] Samuel Collombet, Chris van Oevelen, Jose Luis Sardina Ortega, Wassim Abou-Jaoudé, Bruno Di Stefano, Morgane Thomas-Chollier, Thomas Graf, and Denis Thieffry. Logical modeling of lymphoid and myeloid cell specification and transdifferentiation. *Proceedings of the National Academy of Sciences*, 114(23):5792–5799, 2017.

[CZD+11] Benoit Charloteaux, Quan Zhong, Matija Dreze, Michael E. Cusick, David E. Hill, and Marc Vidal. Protein–protein interactions and networks: forward and reverse edgetics. In *Yeast Systems Biology*, pages 197–213. Springer, 2011.

[DB08] Maria I. Davidich and Stefan Bornholdt. Boolean network model predicts cell cycle sequence of fission yeast. *PLOS ONE*, 3(2):e1672, 2008.

[dSB14] Antonio del Sol and Noel J. Buckley. Concise review: A population shift view of cellular reprogramming. *Stem Cells*, 32(6):1367–1372, 2014.

[EMMP16] Jennifer Enciso, Hector Mayani, Luis Mendoza, and Rosana Pelayo. Modeling the pro-inflammatory tumor microenvironment in acute lymphoblastic leukemia predicts a breakdown of hematopoietic-mesenchymal communication networks. *Frontiers in Physiology*, 7:349, 2016.

[FER+13] Naznin Fauzia, Venmugil Elango, Mahesh Ravishankar, Jagannathan Ramanujam, Fabrice Rastello, Atanas Rountev, Louis-Noël Pouchet, and Ponnuswamy Sadayappan. Beyond reuse distance analysis: Dynamic analysis for characterization of data locality potential. *ACM Transactions on Architecture and Code Optimization*, 10(4):1–29, 2013.

[FLNP00] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7(3-4):601–620, 2000.

[FMKS13] Bernold Fiedler, Atsushi Mochizuki, Gen Kurosawa, and Daisuke Saito. Dynamics and control at feedback vertex sets. I: Informative and determining nodes in regulatory networks. *Journal of Dynamics and Differential Equations*, 25(3):563–604, 2013.

[FTS20] Laura Cifuentes Fontanals, Elisa Tonello, and Heike Siebert. Control strategy identification via trap spaces in Boolean networks. In *Proc. 18th International Conference on Computational Methods in Systems Biology*, volume 12314 of *LNCS*, pages 159–175. Springer, 2020.

[GCBP+13] Luca Grieco, Laurence Calzone, Isabelle Bernard-Pierrot, François Radvanyi, Brigitte Kahn-Perles, and Denis Thieffry. Integrative modelling of the influence of MAPK network on cancer cell fate decision. *PLOS Computational Biology*, 9(10):e1003286, 2013.

[GD19] Alexander Grath and Guohao Dai. Direct cell reprogramming for tissue engineering and regenerative medicine. *Journal of Biological Engineering*, 13(1):14, 2019.

[GGC16] Liesbet Geris and David Gomez-Cabrero. An introduction to uncertainty in the development of computational models of biological processes. In *Uncertainty in Biology*, pages 3–11. Springer, 2016.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and intractability: a guide to the theory of incompleteness*. A series of books in mathematical sciences. W. H. Freeman and Co., 1979.

[GJFM18] Shawn Gu, John Johnson, Fazle E. Faisal, and Tijana Milenković. From homogeneous to heterogeneous network alignment via colored graphlets. *Scientific Reports*, 8(1):1–16, 2018.

[GLDB14] Jianxi Gao, Yang-Yu Liu, Raissa M. D'Souza, and Albert-László Barabási. Target control of complex networks. *Nature Communications*, 5:5415, 2014.

[GLSG01] F.J. Geske, R. Lieberman, R. Strange, and L.E. Gerschenson. Early stages of p53-induced apoptosis are reversible. *Cell Death & Differentiation*, 8(2):182–191, 2001.

[Gol19] Michael S. Goligorsky. New trends in regenerative medicine: reprogramming and reconditioning. *Journal of the American Society of Nephrology*, 30(11):2047–2051, 2019.

[GR16] Alexander J. Gates and Luis M. Rocha. Control of complex networks requires both structure and dynamics. *Scientific Reports*, 6(1):1–11, 2016.

[GSPP19] 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.

[GTZ$^+$18] Diego Germini, Tatiana Tsfasman, Vlada V. Zakharova, Nikolajs Sjakste, Mar Lipinski, and Yegor Vassetzky. A comparison of techniques to evaluate the effectiveness of genome editing. *Trends in Biotechnology*, 36(2):147–159, 2018.

[Gur62] John B. Gurdon. The developmental capacity of nuclei taken from intestinal epithelium cells of feeding tadpoles. *Development*, 10(4):622–640, 1962.

[HGZ$^+$12] Franziska Herrmann, Alexander Groß, Dao Zhou, Hans A Kestler, and Michael Kühl. A Boolean model of the cardiac gene regulatory network determining first and second heart field identity. *PLOS ONE*, 7(10):e46798, 2012.

[HKU$^+$17] Julien Herrmann, Jonathan Kho, Bora Uçar, Kamer Kaya, and Ümit V Çatalyürek. Acyclic partitioning of large directed acyclic graphs. In *Proc. 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 371–380. IEEE, 2017.

[HLT$^+$09] Michael Hecker, Sandro Lambeck, Susanne Toepfer, Eugene Van Someren, and Reinhard Guthke. Gene regulatory network inference: data integration in dynamic models - a review. *BioSystems*, 96(1):86–103, 2009.

[Hua01] Sui Huang. Genomics, complexity and drug discovery: insights from Boolean network models of cellular regulation. *Pharmacogenomics*, 2(3):203–222, 2001.

[Kau69] Stuart Kauffman. Homeostasis and differentiation in random genetic control networks. *Nature*, 224:177–178, 1969.

[Ker71] Brian W. Kernighan. Optimal sequential partitions of graphs. *Journal of the ACM*, 18(1):34–40, 1971.

[KMST11] Jan Krumsiek, Carsten Marr, Timm Schroeder, and Fabian J. Theis. Hierarchical differentiation of myeloid progenitors is encoded in the transcription factor network. *PLOS ONE*, 6(8):e22649, 2011.

[KPC13] Junil Kim, Sang-Min Park, and Kwang-Hyun Cho. Discovery of a kernel for controlling biomolecular regulatory networks. *Scientific Reports*, 3:2223, 2013.

[KS08] Guy Karlebach and Ron Shamir. Modelling and analysis of gene regulatory networks. *Nature Reviews Molecular Cell Biology*, 9(10):770–780, 2008.

[KSS17] 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.

[LCL17] Jinling Liang, Hongwei Chen, and James Lam. An improved criterion for controllability of Boolean control networks. *IEEE Transactions on Automatic Control*, 62(11):6012–6018, 2017.

[LK12] Pey-Chang Kent Lin and Sunil P. Khatri. Application of Max-SAT-based ATPG to optimal cancer therapy design. *BMC Genomics*, 13(S6):S5, 2012.

[LQR17] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. MCMAS: An open-source model checker for the verification of multi-agent systems. *International Journal on Software Tools for Technology Transfer*, 19(1):9–30, 2017.

[LSB11] Yang-Yu Liu, Jean-Jacques Slotine, and Albert-László Barabási. Controllability of complex networks. *Nature*, 473:167–173, 2011.

[LZH$^+$16] Jianquan Lu, Jie Zhong, Daniel W.C. Ho, Yang Tang, and Jinde Cao. On controllability of delayed Boolean control networks. *SIAM Journal on Control and Optimization*, 54(2):475–494, 2016.

[MDCR$^+$16] Alberto J.M. Martin, Calixto Dominguez, Sebastian Contreras-Riquelme, David S. Holmes, and Tomas Perez-Acle. Graphlet based metrics for the comparison of gene regulatory networks. *PLOS ONE*, 11(10):e0163497, 2016.

[MFKS13] Atsushi Mochizuki, Bernold Fiedler, Gen Kurosawa, and Daisuke Saito. Dynamics and control at feedback vertex sets. II: A faithful monitor to determine the diversity of molecular activities in regulatory networks. *Journal of Theoretical Biology*, 335:130–146, 2013.

[MGFA19] Mohammad Moradi, Sama Goliaei, and Mohammad-Hadi Foroughmand-Araabi. A Boolean network control algorithm guided by forward dynamic programming. *PLOS ONE*, 14(5):e0215449, 2019.

[MHK10] 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.

[MHP16] Hugues Mandon, Stefan Haar, and Loïc Paulevé. Relationship between the reprogramming determinants of Boolean networks and their interaction graph. In *Proc. 5th International Workshop on Hybrid Systems Biology*, volume 9957 of *LNCS*, pages 113–127. Springer, 2016.

[MHP17] Hugues Mandon, Stefan Haar, and Loïc Paulevé. Temporal reprogramming of Boolean networks. In *Proc. 15th International Conference on Computational Methods in Systems Biology*, volume 10545 of *LNCS*, pages 179–195. Springer, 2017.

[MKH$^+$19] Neel S. Madhukar, Prashant K. Khade, Linda Huang, Kaitlyn Gayvert, Giuseppe Galletti, Martin Stogniew, Joshua E. Allen, Paraskevi Giannakakou, and Olivier Elemento. A bayesian machine learning approach for drug target identification using diverse data types. *Nature Communications*, 10(1):1–14, 2019.

[MPQY19] 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, 2019.

[MPS17] Orlando Moreira, Merten Popp, and Christian Schulz. Evolutionary acyclic graph partitioning. *arXiv preprint arXiv:1709.08563*, 2017.

[MPSY18] 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, 2018.

[MPY15] Andrzej Mizera, Jun Pang, and Qixia Yuan. ASSA-PBN: a tool for approximate steady-state analysis of large probabilistic Boolean networks. In *Proc. 13th International Symposium on Automated Technology for Verification and Analysis*, volume 9364 of *LNCS*, pages 214–220. Springer, 2015.

[MSH$^+$19] Hugues Mandon, Cui Su, Stefan Haar, Jun Pang, and Loïc Paulevé. Sequential reprogramming of Boolean networks made practical. In *Proc. 17th International Conference on Computational Methods in Systems Biology*, volume 11773 of *LNCS*, pages 3–19. Springer, 2019.

[MSP$^+$19] Hugues Mandon, Cui Su, Jun Pang, Soumya Paul, Stefan Haar, and Loïc Paulevé. Algorithms for the sequential reprogramming of Boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16(5):1610–1619, 2019.

[MVCAL16] David Murrugarra, Alan Veliz-Cuba, Boris Aguilar, and Reinhard Laubenbacher. Identification of control targets in Boolean molecular network models via computational algebra. *BMC Systems Biology*, 10(1):94, 2016.

[NCCT10] Aurélien Naldi, Jorge Carneiro, Claudine Chaouiya, and Denis Thieffry. Diversity and plasticity of th cell types predicted from regulatory network modelling. *PLOS Computational Biology*, 6(9):e1000912, 2010.

[NV12] Tamás Nepusz and Tamás Vicsek. Controlling edge dynamics in complex networks. *Nature Physics*, 8(7):568–573, 2012.

[ODRS14] 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, 2014.

[OKS+16] Barbara Offermann, Steffen Knauer, Amit Singh, María L Fernández-Cachón, Martin Klose, Silke Kowar, Hauke Busch, and Melanie Boerries. Boolean modeling reveals the necessity of transcriptional regulation for bistability in PC12 cell differentiation. *Frontiers in Genetics*, 7:44, 2016.

[Pei14] Tiago P. Peixoto. The graph-tool python library. *figshare*, 2014. Available at http://graph-tool.skewed.de.

[PG18] Arnaud Poret and Carito Guziolowski. Therapeutic target discovery using Boolean network attractors: improvements of kali. *Royal Society Open Science*, 5(2):171852, 2018.

[PSPM18] Soumya Paul, Cui Su, Jun Pang, and Andrzej Mizera. A decomposition-based approach towards the control of Boolean networks. In *Proc. 9th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 11–20. ACM Press, 2018.

[PSPM19] 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*, 2019. accepted.

[PWHL17] Shao-Peng Pang, Wen-Xu Wang, Fei Hao, and Ying-Cheng Lai. Universal framework for edge controllability of complex networks. *Scientific Reports*, 7(1):1–12, 2017.

[Ric19] Adrien Richard. Positive and negative cycles in Boolean networks. *Journal of Theoretical Biology*, 463:67–76, 2019.

[RRC+15] 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.

[RRL$^+$08] Sobia Raza, Kevin A. Robertson, Paul A. Lacaze, David Page, Anton J. Enright, Peter Ghazal, and Tom C. Freeman. A logic-based diagram of signalling pathways central to macrophage activation. *BMC Systems Biology*, 2(1):36, 2008.

[SB07] Thomas Schlitt and Alvis Brazma. Current approaches to gene regulatory network modelling. *BMC Bioinformatics*, 8(S6):S9, 2007.

[SD16] Deepak Srivastava and Natalie DeWitt. In vivo cellular reprogramming: the next generation. *Cell*, 166(6):1386–1396, 2016.

[SDKZ02] 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.

[SFL$^+$09] Özgür Sahin, Holger Fröhlich, Christian Löbke, Ulrike Korf, Sara Burmester, Meher Majety, Jens Mattern, Ingo Schupp, Claudine Chaouiya, Denis Thieffry, Annemarie Poustka, Stefan Wiemann, Tim Beissbarth, and Dorit Arlt. Modeling ERBB receptor-regulated G1/S transition to find novel targets for de novo trastuzumab resistance. *BMC Systems Biology*, 3(1):1, 2009.

[SKI$^+$20] 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.

[SNK$^+$12] Amit Singh, Juliana M. Nascimento, Silke Kowar, Hauke Busch, and Melanie Boerries. Boolean approach to signalling pathway modelling in HGF-induced keratinocyte migration. *Bioinformatics*, 28(18):495–501, 2012.

[Som15] Fabio Somenzi. CUDD: CU decision diagram package - release 2.5.1. http://vlsi.colorado.edu/ fabio/CUDD/, 2015.

[SP20a] Cui Su and Jun Pang. CABEAN: a software for the control of asynchronous Boolean networks. *Bioinformatics*, 2020. accpeted.

[SP20b] Cui Su and Jun Pang. A dynamics-based approach for the target control of Boolean networks. In *Proc. 11th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*. ACM Press, 2020. accepted.

[SP20c] Cui Su and Jun Pang. Sequential temporary and permanent control of Boolean networks. In *Proc. 18th International Conference on Computational Methods in Systems Biology*, volume 12314 of *LNCS*, pages 234–251. Springer, 2020.

[SPP19a] Cui Su, Jun Pang, and Soumya Paul. Towards optimal decomposition of Boolean networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2019. accepted.

[SPP19b] Cui Su, Soumya Paul, and Jun Pang. Controlling large Boolean networks with temporary and permanent perturbations. In *Proc. 23rd International Symposium on Formal Methods*, volume 11800 of *LNCS*, pages 707–724. Springer, 2019.

[SPRF08] Manish Dev Shrimali, Awadhesh Prasad, Ram Ramaswamy, and Ulrike Feudel. The nature of attractor basins in multistable systems. *International Journal of Bifurcation and Chaos*, 18(06):1675–1688, 2008.

[Sto18] Lewi Stone. The feasibility and stability of large complex biological networks: a random matrix approach. *Scientific Reports*, 8(1):1–12, 2018.

[TMP$^+$13] 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):46, 2013.

[TPM$^+$12] 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.

[TTO$^+$07] Kazutoshi Takahashi, Koji Tanabe, Mari Ohnuki, Megumi Narita, Tomoko Ichisaka, Kiichiro Tomoda, and Shinya Yamanaka. Induction of pluripotent stem cells from adult human fibroblasts by defined factors. *Cell*, 131(5):861–872, 2007.

[VCS13] Nedumparambathmarath Vijesh, Swarup Kumar Chakrabarti, and Janardanan Sreekumar. Modeling of gene regulatory networks: a review. *Journal of Biomedical Science and Engineering*, 6(02):223, 2013.

[VSRGS17] Santiago Videla, Julio Saez-Rodriguez, Carito Guziolowski, and Anne Siegel. caspo: a toolbox for automated reasoning on the response of logical signaling networks families. *Bioinformatics*, 33(6):947–950, 2017.

[WKM15] Daniel K. Wells, William L. Kath, and Adilson E. Motter. Control of stochastic and induced switching in biophysical networks. *Physical Review X*, 5(3):031036, 2015.

[WSH$^+$16] Le-Zhi Wang, Ri-Qi Su, Zi-Gang Huang, Xiao Wang, Wen-Xu Wang, Celso Grebogi, and Ying-Cheng Lai. A geometrical approach to control and controllability of nonlinear dynamical networks. *Nature Communications*, 7(1):1–11, 2016.

[WSZS19] 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.

[Yam07] Shinya Yamanaka. Strategies and new developments in the generation of patient-specific pluripotent stem cells. *Cell Stem Cell*, 1(1):39–49, 2007.

[YKOO$^+$18] Ayako Yachie-Kinoshita, Kento Onishi, Joel Ostblom, Matthew A. Langley, Eszter Posfai, Janet Rossant, and Peter W. Zandstra. Modeling signaling-dependent pluripotency with boolean logic to predict cell fate transitions. *Molecular Systems Biology*, 14(1):e7952, 2018.

[YMPQ19] 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.

[YYCJ19] Jumei Yue, Yongyi Yan, Zengqiang Chen, and Xin Jin. Identification of predictors of Boolean networks from observed attractor states. *Mathematical Methods in the Applied Sciences*, 42(11):3848–3864, 2019.

[ZH14] Peican Zhu and Jie Han. Asynchronous stochastic Boolean networks as gene network models. *Journal of Computational Biology*, 21(10):771–783, 2014.

[ZKF13] 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.

[ZL08] Ren Zhang and Yan Lin. Deg 5.0, a database of essential genes in both prokaryotes and eukaryotes. *Nucleic Acids Research*, 37(suppl_1):D455–D458, 2008.

[ZLK$^+$19] Jie Zhong, Yang Liu, Kit Ian Kou, Liangjie Sun, and Jinde Cao. On the ensemble controllability of Boolean control networks using STP method. *Applied Mathematics and Computation*, 358:51–62, 2019.

[ZLLC18] Qunxi Zhu, Yang Liu, Jianquan Lu, and Jinde Cao. Further results on the controllability of Boolean control networks. *IEEE Transactions on Automatic Control*, 64(1):440–442, 2018.

[ZnA15] Jorge G.T. Zañudo and Réka Albert. Cell fate reprogramming by control of intracellular network dynamics. *PLOS Computational Biology*, 11(4):e1004193, 2015.

[ZYA17] Jorge Gomez Tejeda Zañudo, Gang Yang, and Réka Albert. Structure-based control of complex networks with nonlinear dynamics. *Proceedings of the National Academy of Sciences*, 114(28):7234–7239, 2017.

[ZYL$^+$13] 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.

Curriculum Vitae

2016 – 2020	Ph.D. student, University of Luxembourg, Luxembourg
2013 – 2016	M.E. in Systems Engineering, Yanshan University, China.
2009 – 2013	M.E. in Automation, Yanshan University, China

Born on January 29, 1991, Hebei, China.