



BOOLEAN NETWORK INFERENCE AND CONTROL USING METAHEURISTIC ALGORITHMS

By

NING SHI

A thesis submitted to
the University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
University of Birmingham
December 2020

ABSTRACT

This thesis focuses on the topic of gene regulatory network inference and control based on the Boolean network model. Though investigated for decades, inferring gene regulatory networks from the experimental data remains a challenging problem in computational systems biology. The major goal of the network inference is to discover and model the interactions between the genes and the regulatory rules that govern their dynamical changes. The control of gene regulatory networks is also an outstanding challenge in computational systems biology. One of the central tasks is to achieve an understanding of the network dynamics that guiding a cellular state to a desired state with control targets. This thesis addresses these problems based on the Boolean network model. The main contributions of this thesis include:

- Propose an And/Or tree ensemble algorithm for inferring gene regulatory networks from short and noisy data
- Propose a genetic algorithm with marker-based encoding for inferring gene regulatory networks
- Propose a genetic algorithm with a coding of directed ordered tree for controlling gene regulatory networks

Contents

	Page
1 Introduction	1
1.1 Background	3
1.1.1 Gene regulatory networks	3
1.1.2 Gene expression data	4
1.1.3 Boolean network model	5
1.2 Gene regulatory network inference	7
1.3 Gene regulatory network control	9
1.4 Research questions	10
1.4.1 How to infer more accurate Boolean network topology and its dynamics from short and noisy gene expression data?	10
1.4.2 How to improve the scalability and accuracy of Boolean network in- ference?	11
1.4.3 How to find the control kernel for a Boolean network?	13
1.5 Thesis contributions	13
1.6 Thesis organization	15
2 Literature Review	17
2.1 Gene regulatory network inference	17
2.1.1 Optimization methods	17
2.1.2 Feature selection methods	23
2.1.3 Scoring Methodology	27

2.2	Gene regulatory network control	29
2.2.1	Dynamics-based methods	29
2.2.2	Structure-based methods	30
2.3	Summary	33
3	Inferring Boolean networks through And/Or tree ensemble	34
3.1	Introduction	34
3.2	Methods	37
3.2.1	And/Or trees representation of Boolean functions	37
3.2.2	Algorithm	38
3.2.2.1	And/Or tree inference	39
3.2.2.2	Prime implicant importance measure	42
3.2.2.3	Recursive feature reconstruction and elimination	44
3.3	Computational experiments	46
3.3.1	Datasets	46
3.3.1.1	Artificial network	46
3.3.1.2	Real-world network	48
3.3.1.3	Experimental design	49
3.3.2	Results	53
3.3.2.1	Performance on artificial network	53
3.3.2.2	Performance on real-world network	58
3.4	Summary	63
4	Inferring Boolean networks using a genetic algorithm with marker-based encoding	64
4.1	Introduction	64
4.2	Methods	66
4.2.1	Marker-based encoding scheme	67
4.2.2	Algorithm	68

4.3	Computational experiments	73
4.3.1	Datasets	73
4.3.1.1	Artificial networks	73
4.3.1.2	Real-world network	73
4.3.2	Experimental design	76
4.3.3	Results	77
4.3.3.1	Performance on artificial networks	77
4.3.3.2	Performance on real-world network	79
4.4	Summary	86
5	Finding the control kernel for controlling gene regulatory networks	88
5.1	Introduction	88
5.1.1	Problem definition and intuition	90
5.2	Method	92
5.2.1	Encoding scheme for directed ordered tree	92
5.2.2	Algorithm	94
5.3	Results	100
5.3.1	Experimental design	100
5.3.2	Performance on control kernel identification	103
5.4	Summary	107
6	Conclusion and discussion	110
6.1	Conclusion	110
6.2	Discussion	111
A	Supplementary to Chapter 3	114
A.1	Upper limit on tree size	114
A.2	Experimental results	116
A.2.1	Performance on artificial network	116

A.2.2 Performance on real-world network	117
References	126

List of Figures

1.1	An example of a Boolean network including 3 nodes. It presents the topology of the network, the Boolean functions and all possible state transitions. . . .	6
1.2	An example of a state transition diagram and its control problem. (a) The original state transition diagram of a Boolean network. (b)-(d) By controlling a set of genes, any states of the network can flow into the the desired attractor (marked as a red node) eventually.	7
2.1	Network $\mathbf{G}(\mathbf{A}, \mathbf{B})$ correspond to the system $(\mathbf{A}, \mathbf{B})$	32
3.1	And/Or tree example. An example of a three-level And/Or tree with 5 leaves that represents a DNF formula $(x_1 \wedge x_2) \vee (x_3 \wedge \neg x_4 \wedge x_5)$	38
3.2	Illustration of bootstrap samples and the corresponding out-of-bag (oob) samples. Each column represents an expression vector $\mathbf{x}^j(t)$. Each row denotes an input feature (i.e., a gene or a prime implicant). For each target node, we generate a set of pairs of a bootstrap and an oob sample. Bootstrap samples are used to infer ensembles of trees. Oob samples are used to compute the importances of prime implicants based on the tree ensemble.	43
3.3	Part of the topology of a single parasegment primordium network. The whole network is an interconnected network of four identical cells. Nodes located inside the dashed rectangle are in one cell. Nodes 16-21 are in the adjacent cells. The directed edges represent the existence of regulation without implying the activation or inhibition.	48

3.4	ATEN inferred more accurate real-world network topology from short and noisy data. Experimental results of ATEN, MIBNI, Best-Fit and REACT for the inferred network from the datasets including 1% noise and 5% noise. The maximum in-degree of each target node inferred by ATEN, MIBNI and Best-Fit was limited to 8. The results of ATEN are presented using average and standard deviation values. The experimental results of REACT, i.e., the maximum and minimum Recall and FPR values were collected from the literature (i.e., Vera-Licona et al. (2014)), and the F-score values cannot be collected.	61
4.1	An example of a chromosome with specified values. According to specific definitions in Table 1, the first <i>Start</i> (i.e., 10) marks the beginning of sequence 1. Next to <i>Start</i> is 1 and it denotes the target node x_1 whose state is determined by the input nodes x_2 , x_3 , and x_1 with the logical operators <i>OR</i> and <i>AND</i> . Similarly, the sequence 2 and sequence 3 correspond to the second and the third Boolean functions, respectively.	69
4.2	The topology of the idealized protein-DNA interaction network. The directed edges represent the existence of regulation without implying the activation or inhibition.	75
4.3	The topology of the cAMP receptor of D.discoideum network. The directed edges represent the existence of regulation without implying the activation or inhibition.	75
5.1	An example for identifying the control kernel. (a) The original state transition diagram of a Boolean network. (b) By controlling the second gene when the network reaches particular state, the desired attractor (marked as a red node) is reachable from all possible initial states.	91

5.2	An example of a tree and an array to represent it. In the tree, each basin corresponds to a node and each edge between two basins is presented by the dashed arrow. The number attached in an edge specifies the order for the children of a basin. The lower part presents this tree as an array consisting of three strings.	93
5.3	An example of crossover operation by recombining two parent trees to produce one offspring.	96
5.4	An example of mutation operator by removing a subtree and appending this subtree.	98
5.5	The state transition diagram of the budding yeast cell-cycle regulatory network after regulating the genes of the control kernel. Each node denotes a network state and each black arrow denotes the original state transition between two states. The attractor with the largest basin is used as the desired attractor. All red arrows constitute the global paths defined by our algorithm.	103
5.6	The state transition diagram of the fission yeast cell-cycle regulatory network after regulating the genes of the control kernel. Each node denotes a network state and each black arrow denotes the original state transition between two states. The attractor with the largest basin is used as the desired attractor. All red arrows constitute the global paths defined by our algorithm.	105
5.7	The state transition diagram of the gene regulatory network underlying mammalian cortical area after regulating the genes of the control kernel. Each node denotes a network state and each black arrow denotes the original state transition between two states. The attractor with the largest basin is used as the desired attractor. All red arrows constitute the global paths defined by our algorithm.	106

5.8	The state transition diagram of the gene regulatory network underlying mouse myeloid development after regulating the genes of the control kernel. Each node denotes a network state and each black arrow denotes the original state transition between two states. The attractor with the largest basin is used as the desired attractor. All red arrows constitute the global paths defined by our algorithm.	107
5.9	The state transition diagram of the ERBB-receptor regulated network after regulating the genes of the control kernel. Each node denotes a network state and each black arrow denotes the original state transition between two states. The attractor with the largest basin is used as the desired attractor. All red arrows constitute the global paths defined by our algorithm.	108
5.10	The state transition diagram of the CREB signalling network after regulating the genes of the control kernel. Each node denotes a network state and each black arrow denotes the original state transition between two states. The attractor with the largest basin is used as the desired attractor. All red arrows constitute the global paths defined by our algorithm.	109
A.1	Experimental results of ATEN for the inferred network with 150 nodes from the datasets including 1% noise and 5% noise. The results are presented using average values. The maximum number of input genes for each target gene was limited to 3, 4, 5, 6, 8 and 10, respectively.	115

List of Tables

3.1	Boolean functions included in the part of a single parasegment primordium network	50
3.2	Steady states of the part of a single parasegment primordium network. . . .	51
3.3	Experimental results of ATEN, MIBNI and Best-Fit for the artificial networks with 50 nodes inferred from the noise-free data. The maximum in-degree of each target node inferred by ATEN, MIBNI and Best-Fit was limited to 8. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.	52
3.4	Experimental results of ATEN, MIBNI and Best-Fit for the artificial networks with 100 nodes inferred from the noise-free data. The maximum in-degree of each target node inferred by ATEN, MIBNI and Best-Fit was limited to 8. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.	54
3.5	Experimental results of ATEN, MIBNI and Best-Fit for the artificial networks with 150 nodes inferred from the noise-free data. The maximum in-degree of each target node inferred by ATEN, MIBNI and Best-Fit was limited to 8. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.	55

3.6	ATEN inferred more accurate artificial network topology from short and noisy data. Experimental results of ATEN, MIBNI and Best-Fit for the networks with 50 nodes inferred from noisy data. The maximum in-degree of each target node inferred by ATEN, MIBNI and Best-Fit was limited to 8. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.	56
3.7	ATEN inferred more accurate artificial network topology from short and noisy data. Experimental results of ATEN, MIBNI and Best-Fit for the networks with 100 nodes inferred from the datasets including 1% noise and 5% noise. The maximum in-degree of each target node inferred by ATEN, MIBNI and Best-Fit was limited to 8. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface. The symbol ‘-’ indicates that the results of Best-Fit cannot be obtained due to the walltime limit (8 days).	57
3.8	ATEN inferred more accurate artificial network topology from short and noisy data. Experimental results of ATEN, MIBNI and Best-Fit for the networks with 150 nodes inferred from noisy data. The maximum in-degree of each target node inferred by ATEN, MIBNI and Best-Fit was limited to 8. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface. The symbol ‘-’ indicates that the results of Best-Fit cannot be obtained due to the walltime limit.	58

3.9	Experimental results of ATEN, MIBNI and Best-Fit for the real-world networks inferred from the noise-free data. The maximum in-degree of each target node inferred by ATEN, MIBNI and Best-Fit was limited to 8. The results of ATEN are presented using average and standard deviation values. The experimental results of REACT, i.e., the best Recall and FPR values which are presented outside brackets and the worst Recall and FPR values which are showed in brackets, were collected from Vera-Licona et al. (2014). The best result among three algorithms is highlighted in boldface. The symbol ‘*’ indicates that the corresponding results cannot be collected.	59
3.10	ATEN inferred more accurate real-world dynamic model from short and noisy data. The maximum in-degree of each target node inferred by ATEN, MIBNI and Best-Fit was limited to 8. The numbers indicate the number of steady states that was successfully retrieved. The best result among three algorithms is highlighted in boldface. Note that the experimental results of REACT was collected from Vera-Licona et al. (2014) and the symbol ‘*’ indicates that the corresponding results cannot be collected.	62
4.1	The encoding details for the components in the chromosome.	69
4.2	The Boolean functions of the idealized protein-DNA interaction network. . .	74
4.3	The Boolean functions of the cAMP receptor of D. discoideum network. . . .	74
4.4	The parameters in the proposed algorithm.	76
4.5	Experimental results of Best-Fit, ATEN, GABNI and MGANI for inferring the network topology which includes 20 nodes. The results of ATEN, GABNI and MGANI are presented using average and standard deviation values. The best result among four algorithms is highlighted in boldface.	77

4.6	Experimental results of Best-Fit, ATEN, GABNI and MGANI for inferring the network topology which includes 30 nodes. The results of ATEN, GABNI and MGANI are presented using average and standard deviation values. The best result among four algorithms is highlighted in boldface.	79
4.7	Experimental results of Best-Fit, ATEN and MGANI for inferring the dynamic model of the network with 20 nodes. The results of ATEN and MGANI are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.	80
4.8	Experimental results of Best-Fit, ATEN and MGANI for inferring the dynamic model of the network with 30 nodes. The results of ATEN and MGANI are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.	81
4.9	Experimental results of Best-Fit, ATEN, GABNI and MGANI for inferring the the idealized protein-DNA network topology from the experimental data. The results of ATEN, GABNI and MGANI are presented using average and standard deviation values. The best result among four algorithms is highlighted in boldface.	82
4.10	Experimental results of Best-Fit, ATEN, GABNI and MGANI for inferring the cAMP receptor of D. discoideum network topology from the experimental data. The results of ATEN, GABNI and MGANI are presented using average and standard deviation values. The best result among four algorithms is highlighted in boldface.	83
4.11	Experimental results of Best-Fit, ATEN, GABNI and MGANI for inferring the Drosophila segment network topology from the experimental data. The results of ATEN, GABNI and MGANI are presented using average and standard deviation values. The best result among four algorithms is highlighted in boldface.	84

4.12	Experimental results of Best-Fit, ATEN and MGANI for inferring the dynamic model of the protein-DNA interaction network. The results of ATEN and MGANI are presented using average and standard deviation values.	85
4.13	Experimental results of Best-Fit, ATEN and MGANI for inferring the dynamic model of the cAMP receptor of D. discoideum network. The results of ATEN and MGANI are presented using average and standard deviation values. . . .	86
4.14	Experimental results of Best-Fit, ATEN and MGANI for inferring the dynamic model of the Drosophila segment polarity network. The results of ATEN and MGANI are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.	87
5.1	Gene regulatory networks used in our experiment	102
5.2	The control kernel found for real-world gene regulatory networks. Our method were compared with CKGA, MSS and CKSTP. The symbol ‘-’ indicates that the corresponding results cannot be collected.	104
A.1	Experimental results of ATEN, MIBNI and Best-Fit for the artificial networks with 50 nodes inferred from the noise-free data. The maximum in-degree of each target gene inferred by ATEN, MIBNI and Best-fit was limited to 5. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.	117
A.2	ATEN inferred more accurate artificial network topology from short and noisy data. Experimental results of ATEN, MIBNI and Best-Fit for the networks with 50 nodes inferred from noisy data. The maximum in-degree of each target gene inferred by ATEN, MIBNI and Best-fit was limited to 5. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.	118

A.3	Experimental results of ATEN, MIBNI and Best-Fit for the artificial networks with 100 nodes inferred from the noise-free data. The maximum in-degree of each target gene inferred by ATEN, MIBNI and Best-fit was limited to 5. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.	119
A.4	ATEN inferred more accurate artificial network topology from short and noisy data. Experimental results of ATEN, MIBNI and Best-Fit for the networks with 100 nodes inferred from noisy data. The maximum in-degree of each target gene inferred by ATEN, MIBNI and Best-fit was limited to 5. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.	120
A.5	Experimental results of ATEN, MIBNI and Best-Fit for the artificial networks with 150 nodes inferred from the noise-free data. The maximum in-degree of each target gene inferred by ATEN, MIBNI and Best-fit was limited to 5. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.	121
A.6	ATEN inferred more accurate artificial network topology from short and noisy data. Experimental results of ATEN, MIBNI and Best-Fit for the networks with 150 nodes inferred from noisy data. The maximum in-degree of each target gene inferred by ATEN, MIBNI and Best-fit was limited to 5. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.	122

A.7	Experimental results of ATEN, MIBNI and Best-Fit for the real-world networks inferred from the noise-free data. The maximum in-degree of each target gene inferred by ATEN, MIBNI and Best-fit was limited to 4. The results of ATEN are presented using average and standard deviation values. The experimental results of REACT, i.e., the best Recall and FPR values which are presented outside brackets and the worst Recall and FPR values which are showed in brackets, were collected from Vera-Licona et al., 2014. The best result among three algorithms is highlighted in boldface. The symbol ‘*’ indicates that the corresponding results cannot be collected.	123
A.8	ATEN inferred more accurate real-world network topology from short and noisy data. Experimental results of ATEN, MIBNI, Best-Fit and REACT for the inferred network from noisy data. The maximum in-degree of each target gene inferred by ATEN, MIBNI and Best-fit was limited to 4. The results of ATEN are presented using average and standard deviation values. The experimental results of REACT, i.e., the best Recall and FPR values which are presented outside brackets and the worst Recall and FPR values which are showed in brackets, were collected from Vera-Licona et al., 2014. The symbol ‘*’ indicates that the corresponding results cannot be collected.	124
A.9	ATEN inferred more accurate real-world dynamic model from short and noisy data. The maximum in-degree of each target gene inferred by ATEN, MIBNI and Best-fit was limited to 4. The numbers indicate the number of steady states that was successfully retrieved. The best result among three algorithms is highlighted in boldface. The experimental results of REACT was collected from Vera-Licona et al., 2014 and the symbol ‘*’ indicates that the corresponding results cannot be collected.	125

Chapter One

Introduction

In systems biology, understanding biomolecular structure and dynamics is crucial to elucidate the complex biological processes of a living cell. Many crucial biological functions and cellular behaviour are based on a consequence of complex interactions among genes, proteins and small molecules (Hartwell et al., 1999; Albert, 2005). In order to understand the regulatory mechanisms to coordinate multiple biological processes, network based approaches emerged as powerful tools in the past decade (Barabasi and Oltvai, 2004; Albert, 2005; Barabási et al., 2011). They have been widely used in delineating the topology and dynamics of various types of biological networks, such as gene regulatory networks (Kerlebach and Shamir, 2008) which play a vital role in accelerating the development of biotechnology and medicine (Emmert-Streib et al., 2014).

The gene regulatory networks are comprised of nodes, i.e., genes, proteins and small molecules, and edges representing direct as well as indirect regulatory interactions. On the genomic level, the genes in the network can activate or inhibit each other to influence the expression level of the corresponding genes or proteins. Genes and proteins participate in diverse biochemical reactions and further affect numerous cellular processes (De Jong, 2002; Chai et al., 2014). The gene regulatory network thus not only describes the functional interactions among genes but also characterises the dynamical behaviour of the network.

Due to its importance, many computational models and approaches have been developed to infer and control the gene regulatory network. The roles of those models need to describe the regulatory rules among genes, enable artificial simulation of the dynamical behaviour and analyze the dynamical changes through the external signal.

A popular model to capture the characterization of a gene regulatory network is the Boolean network model, which were first proposed by Kauffman (1969). In the Boolean network, the state of a gene is represented by binary values that define whether the gene is active (1) or inactive (0). The dynamical changes of the states of genes are governed by Boolean functions. The Boolean network model has been successfully applied to model various biological networks, e.g., *Drosophila* segment polarity gene network (Albert and Othmer, 2003).

With the rapid developments in high-throughput data-collection techniques, the use of gene expression data to reconstruct gene regulatory networks has received much attention (Blais and Dynlacht, 2005). The major challenge in the gene regulatory network inference is to discover the interactions and regulatory rules among genes from prior knowledge and gene expression data. The network inference can help us gain a better understanding of using gene expression data to elucidate functional mechanisms of cellular processes and identify pharmacological targets (Margolin et al., 2006). It can also help researchers to generate testable hypotheses in experimental design (Lee and Tzou, 2009). Once the gene regulatory network has been defined, finding the minimum set of genes required to design control strategies in the control of gene regulatory networks is another challenging and important problem in system biology. From a practical viewpoint, control of gene regulatory networks is useful for drug discovery and disease treatment (Kim et al., 2013; Hou, Ruan, et al., 2018).

This thesis addresses the above two challenges and three specific research questions. Based on the Boolean network models, we propose two network inference algorithms from short and noisy data that can infer more accurate networks. We validate the effectiveness

of our algorithms on both artificial networks and real-world gene regulatory networks. We present an algorithm for designing control strategies with the minimum set of genes for enforcing real-world gene regulatory network to the desired attractor.

The rest of this chapter gives an introduction of gene regulatory network inference and control. Section 1.1 presents the background of the gene regulatory network and gene expression data. It further presents Boolean networks which are used for modelling the gene regulatory networks. Section 1.2 and Section 1.3 discuss the problem of gene regulatory network inference and control. Section 1.4 presents the research questions. Finally, Section 1.5 lists the main contributions and Section 1.6 states the thesis organization.

1.1 Background

1.1.1 Gene regulatory networks

The gene regulatory network elucidates the relationships among genes and it further provides new biological insight and understanding of the molecular and cellular process (De Jong, 2002; Chai et al., 2014). A gene regulatory network is often modelled as a directed graph consisting of a collection of nodes (i.e., genes, proteins and small molecules) and edges (i.e., regulatory interactions/rules). For an edge pointed from gene x_i to gene x_j , x_i is said to be the regulatory gene that can regulate the expression of the target gene x_j . The regulation can be either activation and inhibition.

In recent years, there have been many computational models proposed for representing gene regulatory networks. Those models include Bayesian network (Friedman et al., 2000; Sanchez-Castillo et al., 2018), information theory model (Liang et al., 1998; Margolin et al., 2006), Boolean network (Kauffman, 1969; Lähdesmäki et al., 2003), ordinary differential equations (ODEs) (Bansal et al., 2006) and neural network (Günther et al., 2009). Bayesian

network is able to handle noisy data but it fails to capture the underlying information of time series data (Chai et al., 2014). Information theory model is suitable to infer gene regulatory network but it not suitable to present the regulatory relationships between multiple genes. On the other hand, the problem of data insufficiency impedes the inference of gene regulatory networks based on ODEs and neural network (Hecker et al., 2009; Chai et al., 2014). Of these different models, one of the simplest models is the Boolean network, which can describe both network topology and dynamics. In this thesis, we focus on using Boolean networks to infer and control the gene regulatory network.

1.1.2 Gene expression data

With the advent of high-throughput genomics, many technologies can be used to study the levels of gene expression. Gene expression data allows us to observe the changes in the expression level of genes over time. Through the analysis of the time series data, we can investigate both the interactions among genes and the temporal pattern in the network. However, different levels of noise may occur in the time series data. And the time series data has typically a small number of samples (Bar-Joseph, 2004).

Massive quantities of biological data have been generated and stored in a variety of databases (Barabási et al., 2011), e.g. Gene Expression Omnibus (GEO) database (Edgar et al., 2002) and ArrayExpress gene expression database (Brazma et al., 2003). Such diverse and complementary types of biological data (e.g. genome sequence data, transcriptome data and proteome data) are supportive to the reconstruction of gene regulatory networks (Hecker et al., 2009).

1.1.3 Boolean network model

Boolean network (BN) is said to be the simplest models in the field of modelling gene regulatory networks. Although Boolean networks are limited as the state of gene expression level can only be described using two states, they can capture some fundamental characteristics of signalling and gene regulatory mechanisms, such as cellular proliferation and apoptosis (Huang, 1999). Modelling genetic algorithm using the Boolean network model was first proposed in Kauffman (1969). Later on, Boolean networks have been widely used to reveal both the gene regulatory network topology (i.e. the interactions between genes) and dynamics (i.e. the regulatory rules) (Liang et al., 1998; Lähdesmäki et al., 2003; Hecker et al., 2009).

A Boolean network $G(X, F)$ is a set of nodes $X = \{x_1, \dots, x_n\}$ representing genes and a set of Boolean functions $F = \{f_1, \dots, f_n\}$ describing the regulatory rules between genes. Each node (i.e., gene) x_i is a Boolean variable taking 1 or 0 to describe its state, where 1 represents that the gene is expressed and 0 represents that the gene is not expressed. The directed edges, or the interactions between the target gene and input genes are associated with the regulatory rule, i.e., the Boolean function. The Boolean function consists of three basic Boolean operators (i.e. $\vee$, $\wedge$ and $\neg$) and Boolean variables.

The Boolean function f_i determines the dynamical changes of the state of gene x_i over time: $x_i(t+1) = f_i(x_{i_1}(t), \dots, x_{i_k}(t)), \forall t$, where $x_i(t+1)$ represents the state of gene x_i at time point $t+1$, and $x_{i_k}(t)$ represents the state of the k th input gene of gene x_i at time point t . The size of the entire state space is 2^n . In synchronous Boolean networks, given a state $s(t)$, we then obtain a state $s(t+1)$ by evaluating all Boolean functions in F at the same time. We present an example of a Boolean network in Figure 1.1 where we present the network topology, Boolean functions and all possible state transitions.

Thus all state transitions in a synchronous Boolean network can be mapped into a directed graph, which is known as the state transition diagram. As shown in Figure 1.2(a),

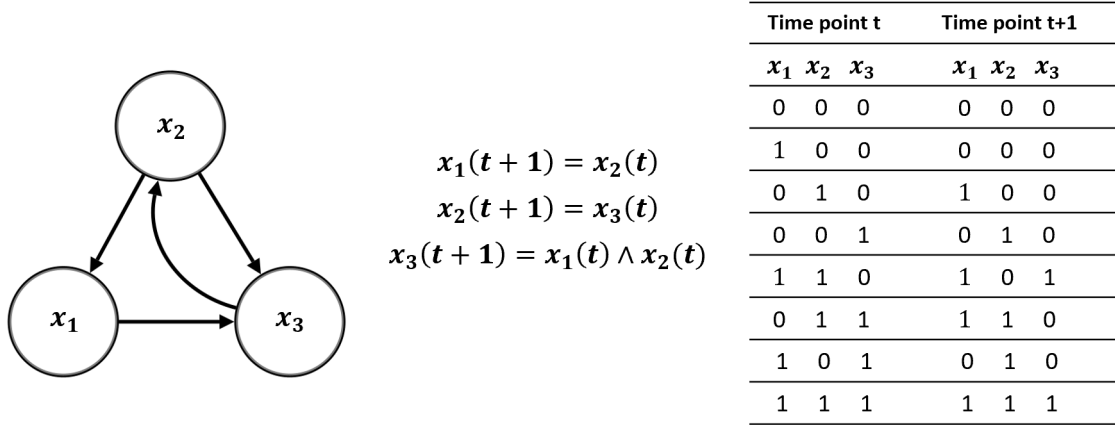


Figure 1.1: An example of a Boolean network including 3 nodes. It presents the topology of the network, the Boolean functions and all possible state transitions.

we present the state transition diagram of a Boolean network including 3 genes. Each node denotes a network state $s_i, i \in \{1, 2, \dots, 2^3\}$ and each edge represents a trajectory between two states.

In the state transition diagram, all trajectories can eventually reach at least one previously visited state. If a trajectory can only fall into a single state, then this single state is called a point attractor (or fixed point, singleton attractor, steady state). Otherwise, more than one states that can reach each other in a trajectory are called a cyclic attractor. We use A to denote the attractor. In Figure 1.2(a), each directed edge denotes the state transition from one state to another one. For instance, the state $(0, 0, 1)$ represents that x_1, x_2 and x_3 takes 0, 0 and 1, respectively at a specific time point. Then at the next time point, it transits to $(0, 1, 1)$. There exist in total three attractors/basins where A_1 is $((0, 0, 0), (0, 0, 1), (0, 1, 1))$, A_2 is $((1, 1, 1), (1, 1, 0))$ and A_3 is a fixed point $(1, 0, 1)$. The set of states that leads to the attractor is called the basin of the attractor and we denote it as B . For example, $B_1 = ((1, 0, 0), (0, 0, 0), (0, 0, 1), (0, 1, 1))$. The attractor denoting a set of stable states in the gene regulatory network can be associated with a biological phenotype or cellular process (Bloomingdale et al., 2018).

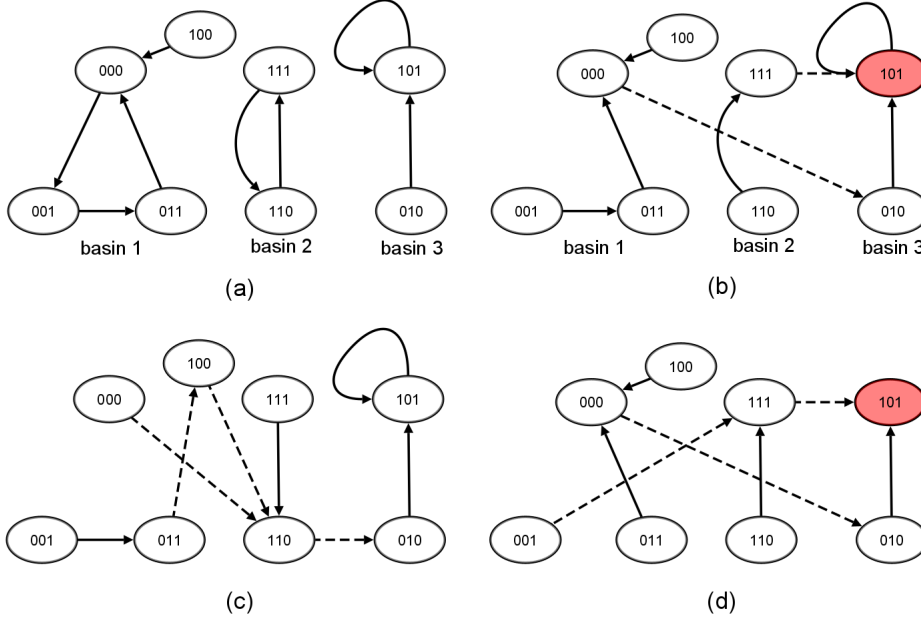


Figure 1.2: An example of a state transition diagram and its control problem. (a) The original state transition diagram of a Boolean network. (b)-(d) By controlling a set of genes, any states of the network can flow into the the desired attractor (marked as a red node) eventually.

1.2 Gene regulatory network inference

The reconstruction of a gene regulatory network based on gene expression data is a challenging problem in systems biology. Hecker et al. (2009) have defined two major tasks in network inference. The first task is to reconstruct the network structure. The structure optimization aims at finding the network connectivity that best fits the provided experimental data. The structural features and constrains can be obtained from the prior knowledge (e.g. limited inputs gene (Arnone and Davidson, 1997) and scale-free topology (Barabasi and Oltvai, 2004)) should also be taken into account. Another task is to identify the model parameters (e.g. type of interactions) from the data. The type of interactions refers to the regulatory rules

which determine the dynamical changes of the gene regulatory network.

For the purpose of inferring a Boolean network, it is necessary to discretize continuous gene expression values in a time series to binary data. The expression of a gene can affect the expression of other genes independently. And some genes play as a functional class to influence the expression of other genes. The fundamental goal of the network inference problem is to discover the underlying interactions and regulatory rules among genes from the time series data. It can also be considered as identifying the Boolean function associated with input genes for each target gene.

Many computational approaches have been developed for inferring gene regulatory networks based on the Boolean network model. A lot of existing algorithms focus primarily on the inference of the topology of Boolean networks. For example, Liang et al. (1998) proposed REVEAL to use mutual information to evaluate dependencies between potential input genes and the target gene. Maucher et al. (2011) developed a correlation approach to reconstruct the topological structure of the network. There are also many algorithms that infer both the Boolean network topology and its dynamics. They have formulated the inference problem mainly as the optimization problem, in which the cost function is to find the best regulatory rules that fit the experimental data. To this end, the genetic algorithm (Barman and Kwon, 2018; Vera-Licona et al., 2014) and other representative algorithms, e.g., Best-Fit (Lähdesmäki et al., 2003) have been proposed. Alternatively, the inference problem has also been solved using the feature selection approach (Barman and Kwon, 2017; Irrthum et al., 2010). A more detailed review of these algorithms is presented in Section 2.1 in Chapter 2.

1.3 Gene regulatory network control

Controlling gene regulatory networks is also a crucial topic for understanding the regulation of biological processes in the field of systems biology. The purpose of network control is to drive the network to a desired cellular state. The cellular state can be modelled by the attractors in the state transition diagram in the Boolean network model. Thus the attractors correspond to different cell states, e.g. apoptosis, aberrant proliferation, normal proliferation (Recanatini and Cabrelle, 2020). This motivates the question of how to drive any undesirable attractors toward a desired attractor. Investigating such control strategy corresponds to the radiation therapy or drug treatment (Xiao and Dougherty, 2007).

Extensive studies have been done on the controllability of Boolean networks. Many methods have been proposed to find control strategies to lead an initial state to a desired state over finite time steps (Datta et al., 2003; Akutsu, Hayashida, Ching, et al., 2007; Langmead and Jha, 2009). Recently, finding the control kernel which leads the network to a desired state has attracted considerable attention (Kim et al., 2013; Hou, Tamura, et al., 2016; Nacher, Ishitsuka, et al., 2019). Kim et al. (2013) developed an algorithm based on a genetic algorithm for discovering a control kernel (i.e., the minimum set of genes). Based on the concept of the state transition diagram, the control kernel is the minimum set of genes which must be used in the control strategy to make any possible initial states flow into the target attractor. Wu et al. (2016) defined the minimum driver node sets (MSSs) by analyzing the network structural controllability. Rafimanzelat and Bahrami (2018b) proposed a method based on semi-tensor product for finding such control kernels. We discuss more details about these algorithms in Section 2.2 in Chapter 2.

We present an example of this network control problem in Figure 1.2. Figure 1.2(a) presents the original state transition diagram of a Boolean network including three genes. Assuming the attractor $(1, 0, 1)$ is the desired attractor, then it is not reachable from any states in basin 1 and basin 2.

To make all possible initial states in basin 1 flow into the desired attractor, as shown in Figure 1.2(b), we take a control action to flip the state of gene x_2 to be 1 when the network reaches the state $(0, 0, 0)$. And we force $(1, 1, 1)$ to $(1, 0, 1)$ by controlling x_2 . We use dashed line to show new trajectories made by our control actions where we only need to control only one gene (i.e., $\{x_2\}$) to make the desired attractor be reachable from any initial states. Thus $\{x_2\}$ is the control kernel of this network. We assume that every state and every gene can be controlled according to the specific control strategies. Alternatively, we can propose other two control strategies as present in Figure 1.2(c)(d). In Figure 1.2(c), we make $(0, 0, 0) \xrightarrow{x_1=1, x_2=1} (1, 1, 0) \xrightarrow{x_1=0} (0, 1, 0) \rightarrow (1, 0, 1)$ and $(0, 1, 1) \xrightarrow{x_1=0, x_2=0, x_3=1} (1, 0, 0) \xrightarrow{x_2=1} (1, 1, 0) \xrightarrow{x_1=1} (0, 1, 0)$ where three genes are required. In Figure 1.2(d), two new paths $(0, 0, 0) \xrightarrow{x_2=1} (0, 1, 0) \rightarrow (1, 0, 1)$ and $(0, 0, 1) \xrightarrow{x_1=1, x_2=1} (1, 1, 1) \xrightarrow{x_2=0} (1, 0, 1)$ are built and it requires two genes. These two alternative control strategies require two and three genes, thus they are not the control kernel for the network.

1.4 Research questions

Based on the introduction of network inference and control problems and the Boolean network model, the thesis aims at investigating and addressing the following key research questions:

1.4.1 How to infer more accurate Boolean network topology and its dynamics from short and noisy gene expression data?

As we introduced in Section 1.2, many previous methods are focusing on identifying the directed interactions between genes. They mainly discovered the linear or nonlinear relationships between a set of input genes and a target gene. For instance, a pair of two genes

are predicted to regulate another gene if the mutual information or the correlation coefficient of their expression levels is above a predefined threshold. Then they tried to define the dynamical changes of the network based on the existing input data and the inferred input combinations. However, as no specific Boolean function is inferred, it is unclear how the dynamics evolve when the input data contains non-deterministic cases. But uncovering the dynamics and functionality of gene regulation is a focal point in systems biology (Hecker et al., 2009). On the other hand, the non-deterministic cases also hamper the identification of the network.

Akutsu, Hayashida, and Tamura (2008) have discussed that 2^n time points are required to identify a unique Boolean network without any restrictions on a Boolean network, where n is the number of genes in a network. For a network including 20 genes, the total number of possible regulatory combinations for a target gene is over 1,000,000. However, an exponential number of time points is non-realistic and the truth is that data is typically short and noisy (Bar-Joseph, 2004; Aalto et al., 2020). In other words, the dataset includes a small number of time series with low sampling frequency and the state of gene expression is affected by the measurement noise, thus it is difficult to generate accurate network representation. On the other hand, the inferred network might not be unique when the scale of the network increases. There may exist multiple Boolean networks fitting the data equally. Accordingly, **the first research question is: How to infer more accurate Boolean network topology and its dynamics from short and noisy gene expression data?**

1.4.2 How to improve the scalability and accuracy of Boolean network inference?

Although there are many algorithms that have been proposed for network inference, but most of them are not efficient in identifying the regulatory rules which include dozens of input genes (Barman and Kwon, 2018). For example, the well-known algorithms such as REVEAL

(Liang et al., 1998) and Best-Fit (Lähdesmäki et al., 2003) require high computational cost as they need to exhaustively search all possible input combinations. To solve the scalability problem, meta-heuristics have been used for network inference (Mendoza and Bazzan, 2011; Vera-Licona et al., 2014; Barman and Kwon, 2018; Shi et al., 2020). Our work ATEN (Shi et al., 2020) uses a simulated annealing (SA) algorithm to find the best Boolean function for each target gene. In each iteration, ATEN searches the Boolean function through a limited number of neighbours, it results in ATEN that needs a long enough amount of time to infer a complex Boolean function associated with a large number of input genes. To infer the large-scale networks, genetic algorithm (GA), one type of metaheuristic, was proposed (Barman and Kwon, 2018). It is easy to cope well with a large solution space (Mendoza and Bazzan, 2011). In GA, each solution can be represented by binary encoding, in which the binary values corresponds to the underlying regulatory relationships between a target gene and input genes. But the scalability problem remains as binary encoding is not efficient for variation operators when the gene is regulated by dozens of input genes.

On the other hand, most of network inference algorithms such as recent work GABNI (Barman and Kwon, 2018) and ATEN (Shi et al., 2020) infer each gene independently. Thus to infer an entire Boolean network, one have to apply the parallelization. We consider to encode all Boolean functions into an individual solution.

Additionally, existing network inference algorithms focus on inferring network topology, or commonly describes the network dynamics using basic operators (i.e., AND, OR, NOT). Thus they are not efficient to reconstruct the Boolean network with complex Boolean functions (e.g., NAND). Then, **our second research question is: How to improve the scalability and accuracy of Boolean network inference?**

1.4.3 How to find the control kernel for a Boolean network?

As we introduced in Section 1.3, finding the minimum set of genes (called control kernel (Kim et al., 2013)) required for controlling the network to a desired cellular state is also a challenge in system biology. Many studies have been done on determining such a set of nodes in complex networks based on the structure of the networks (Lin, 1974; Liu et al., 2011; Gao, Liu, et al., 2014). However, since Boolean network is highly non-linear, these results cannot make all types of Boolean network completely controllable (Wu et al., 2016; Hou, Tamura, et al., 2016). In recent years, only a few studies have investigated this control problem from the perspective of dynamics (Kim et al., 2013; Rafimanzelat and Bahrami, 2018b; Hou, Ruan, et al., 2018). Their methods provided not only the specific control strategies but also the minimum set of controlled genes. Despite these advantages, reliable methods that can guarantee the desired attractor is reachable from any possible states are still lacking. Thus **the third research question is: How to develop an algorithm to find the control kernel for a Boolean network?**

1.5 Thesis contributions

This thesis provides a comprehensive study of gene regulatory network inference algorithms based on Boolean network model and presents the following contributions:

- To infer more accurate Boolean network topology and its dynamics from short and noisy gene expression data, which corresponds to the first research question, we develop an And/Or tree ensemble algorithm for network inference (Chapter 3).
 - We introduce a feature selection approach in network inference, where each feature is not an individual gene but the regulatory combinations of a target gene.

- We propose the first And/Or tree ensemble algorithm to search for robust solutions. We are the first to bridge the gap between ensemble learning and And/Or trees.
- To answer the second research question, i.e., how to improve the scalability and accuracy of Boolean network inference, we propose a genetic algorithm with marker-based encoding for network inference (Chapter 4).
 - We propose to encode the entire Boolean network in a candidate solution in which the marker distinguishes each Boolean function and enables flexible operations.
 - We introduce more logical operators to make the algorithm efficiently infer more diverse Boolean functions.
- For finding the control kernel for a Boolean network (i.e., the third research question), we present a genetic algorithm based on the state transition diagram (Chapter 5).
 - We propose a novel efficient coding scheme of directed ordered tree in a genetic algorithm framework. We also design the initialization, crossover and mutation operators for this coding.
 - Our algorithm can guarantee the desired attractor is reachable from any possible initial states.

This thesis includes the contribution from the following publications:

- Shi, N., Zhu, Z., Tang, K., Parker, D., & He, S. (2020). ATEN: And/Or tree ensemble for inferring accurate Boolean network topology and dynamics. *Bioinformatics*, 36(2), 578-585.
- Liu, X., Shi, N., Wang, Y., Ji Z., & He, S. Data-driven Boolean network inference using a genetic algorithm with marker-based encoding. Major revision required at *IEEE/ACM Transaction on Computational Biology and Bioinformatics*.

- Shi, N., Zhu, Z., Tang, K., Parker, D., & He, S. Finding the minimum set of nodes to control gene regulatory networks. In preparation.

1.6 Thesis organization

This thesis consists of six chapters. The first chapter introduces the background of gene regulatory network inference and control. The rest of this thesis is organized as follows:

Chapter 2 reviews part of important literature in related fields. Section 2.1 introduces representative algorithms developed for gene regulatory network inference based on the Boolean network model. Section 2.2 reviews related works in the identification of control kernel in the Boolean network.

In Chapter 3, we study the network inference problem by explaining our And/Or Tree ensemble algorithm. Section 3.1 describes the motivation and the advantages of the algorithm. Section 3.2 presents the details of our algorithm where we identify important features for inferring the Boolean network. Section 3.3 shows the experimental results on inferring both network topology and dynamics from short and noisy data.

In Chapter 4, we propose a genetic algorithm with a novel encoding scheme for inferring a Boolean network. Section 4.1 introduces the research gap between existing algorithms. Section 4.2 describes our inference algorithm. Section 4.3 validates our algorithm using both artificial and real-world networks.

In Chapter 5, we develop an algorithm for identifying control strategies with the minimum set of genes for controlling the Boolean network. Section 5.1 reviews the related work and formulate the control problem. Section 5.2 describes the framework of our proposed algorithm. Section 5.3 shows the related control strategies for different real-world Boolean networks and presents the state space after performing control strategies.

In Chapter 6, we conclude this thesis, discuss the limitation of the proposed algorithm and present promising future work.

Chapter Two

Literature Review

This chapter is a literature review on existing algorithms for gene regulatory network inference and control based on the Boolean network model. The first section reviews the representative algorithms on the Boolean network inference. The second section reviews representative works on finding control kernels in Boolean networks. The last section gives a summary of this chapter.

2.1 Gene regulatory network inference

2.1.1 Optimization methods

In Section 1.2, we introduce the problem of gene regulatory network inference based on the Boolean network model, where the fundamental goal of the network inference problem is to infer both the network topology and its dynamics. This problem can be considered as an optimization problem, where the cost function is designed by comparing the inferred behaviour of the model with the observed gene expression data and the ground truth network.

The mutual information (MI) which is a measure of the mutual dependence between

two variables, can also be used to describe the relationships between a target gene and a set of input genes. Therefore, many algorithms have also been proposed to maximize a discrete mutual information value for reconstructing the gene regulatory network. One of the most famous and probably the first Boolean network inference algorithm is REVEAL (REVerse Engineering ALgorithm), which is proposed by Liang et al. (1998). The algorithm uses mutual information measures to describe the regulatory relationships between a gene and input combinations based on the state transitions.

In a Boolean network, as each gene has 0 or 1 in the expression data, then we can obtain the probability p_i of $x_i = 0$ and the probability $1 - p_i$ of $x_i = 1$ to describe the state of x_i in the expression data. Then the entropy of x_i is measured by $H(x_i) = -p_i \log p_i - (1 - p_i) \log(1 - p_i)$. For instance, given a time series data of x_i as a sequence (0, 1, 1, 1, 1, 1, 0, 0, 0, 1), then we have $H(x_i) = -0.4 \log(0.4) - 0.6 \log(0.6) = 0.97$. In this way, we can also have the entropy of a set of two genes $H(x_i, x_j) = -\sum p_{i,j} \log p_{i,j}$ where $p_{i,j}$ refers to the probability of value pairs (i.e., $(x_i = 0, x_j = 1)$, $(x_i = 0, x_j = 0)$, $(x_i = 1, x_j = 1)$, $(x_i = 1, x_j = 0)$). The mutual information between two genes is given by $I(x_i, x_j) = H(x_i) + H(x_j) - H(x_i, x_j)$. Since the mutual information can be explained as $I(x_i, x_j)$ is the reduction in the uncertainty of x_i by knowing x_j , if $I(x_i, x_j)/H(x_j)$ equals 1, then x_j exactly knows x_i . Following this idea, the core of REVEAL aims at looking for a class of genes X that maximizes the mutual information $I(X, x_i)$ for each target gene x_i by brute force search. But the REVEAL algorithm have difficulties in reconstructing Boolean networks with the noise. It is also obvious to see that REVEAL is not suitable for large-scale networks of a high number of input genes per gene.

The conventional exhaustive search algorithms have also been used in Akutsu, Miyano, et al. (1999) and Lähdesmäki et al. (2003) for network inference. Specifically, Best-Fit (Lähdesmäki et al., 2003) is also a well-known Boolean network inference algorithm. It searches input genes with all possible combinations for a target gene and finds the best Boolean function that minimizes the error size. Later on, (Nam et al., 2006) proposed

a randomized network search algorithm and present Best-Fit using a much simpler and intuitive way. They suggested that finding the Boolean functions with the minimum error size is equivalent to define the Boolean function with the minimum inconsistency ratio (see Equation 2.1).

Apart from the above algorithms, evolutionary algorithms (EAs) have been widely used due to its ability to cope well with a large solution space (Fogel and Corne, 2002). The genetic algorithm (GA), which is an efficient, adaptive and robust optimization technique in EAs, have attracted researchers for network inference (Pal et al., 2006). The GA is a metaheuristic inspired by the process of natural selection. The main idea of GA is to generate a population of candidate solutions in an iterative process evolving toward better solutions. The population in each iteration is called a generation. Each candidate solution is evaluated by a fitness function, which is also what the GA wants to solve. The more fit individuals are selected to be modified for generating new generation. The general framework of a genetic algorithm that searches for the fittest solution is present in Algorithm 2.1. In the real application, specific objective functions and genetic operators are defined depending on the problem formulation.

Algorithm 2.1: General framework of a genetic algorithm

Input: Parameters (e.g., population size, the maximum number of generations)

Output: Solution

- 1 Initialize a population;
 - 2 Evaluate the fitness of each individual;
 - 3 **while** *termination criteria not met* **do**
 - 4 Select individuals from the population;
 - 5 Apply genetic operators (e.g., crossover and mutation operators) to the individuals;
 - 6 Evaluate the fitness of each individual;
-

Mendoza and Bazzan (2011) have proposed an optimization method to investigate the consistent typologies using the genetic algorithm (GA) without including any prior knowledge. In the proposed method, it aims to find the Boolean network which best represents the hidden relationships between the input data and input combinations. The GA encodes each potential solution in the population as a binary string. Each solution denotes a candidate Boolean network. The binary string can be divided to n parts where n refers to the number of genes in the Boolean network. Each part encodes the input genes and the most consistent output to the corresponding input genes. However, it is unclear which Boolean functions are most consistent with the chromosome if the input data is noisy.

To formulate the fitness function, they introduced the inconsistency ratio (IR) suggested by Nam et al. (2006). Given a target gene with K input genes in total, then there exist 2^K possible input combinations. The total weight of the output value of each predicted combination are stored in variables $w_c(0)$ and $w_c(1)$ respectively where c denotes the c th input combination ($c = 1, \dots, 2^K$). Then for each target gene x_i , the IR can be defined as:

$$IR_i = w^{-1} \sum_{c=1}^{2^K} \min(w_c(0), w_c(1)) \quad (2.1)$$

where w refers to the number of time points. For instance, considering two input genes x_2 and x_3 that regulate x_1 where if $x_2 = 0$ and $x_3 = 0$ (the 1th combination) at five time points and $x_1 = 1$ is always true, then we have $w_1(0) = 0$ and $w_1(1) = 5$. If the second (i.e., $c = 2$) combination $x_2 = 1$ and $x_3 = 0$ corresponds to $x_1 = 1$ at three time points and $x_1 = 0$ at two time points, then $w_2(0) = 2$ and $w_2(1) = 3$. Thus intuitively, the most consistent combination for a target gene keeps the minimum IR . The inconsistency of a network with n genes is accordingly determined by:

$$IRN = \sum_{i=1}^n IR_i \quad (2.2)$$

Then the goal of the genetic algorithm is to search for the best combination of input genes

for each target gene that minimizes IRN . Finally the fitness function is defined as:

$$\phi = \frac{1}{1 + \frac{IRN}{n \times 0.5}} \quad (2.3)$$

To infer a Boolean network, the first step is to generate the initial population where each individual encodes a Boolean network with input genes and regulatory rules. The second step is to compute the fitness value and apply the elitism selection. The crossover and mutation operators are then applied to generate new offspring. The GA finally returns a set of candidate networks. Instead of choosing the network with the best fitness, Mendoza and Bazzan (2011) applied the concept of majority voting where every candidate network votes on the classification of each as present or absent. The existence of an edge is determined according to the majority voting and then the final Boolean network's structure is built.

Vera-Licona et al. (2014) further combined the algebraic approach with the evolutionary algorithm to infer both network topology and its dynamics. The feature of this algorithm is to use polynomials to represent Boolean functions. The basic Boolean function can be expressed using addition and multiplication modulo 2: $x_1 \wedge x_2 = x_1 \cdot x_2$, $x_1 \vee x_2 = x_1 + x_2 + x_1 \cdot x_2$, $\neg x_1 = x_1 + 1$ and $x_1 \oplus x_2 = x_1 + x_2$ where $\oplus$ denotes the logical exclusive OR function XOR. Thus each Boolean function is described by a square-free polynomial.

In the inference framework, each individual is made of n polynomial functions representing a whole Boolean network. The fitness is evaluated by:

$$W_{HM}(1 - \frac{\sum_{j=1}^J H_j}{n \times (\sum_{j=1}^J N_j)}) \quad (2.4)$$

where H_j refers to the Hamming distance between the input data and the data generated by the inferred Boolean network where j refers the j th time series covering N_j time points in the dataset including J time series in total, and W_{HM} is the weight predefined to the dataset. Accordingly, the fitness of a polynomial function f_i can be defined as $W_{HP}(1 - \frac{\sum_{j=1}^J H_j^i}{\sum_{j=1}^J N_j})$ where W_{HP} is the weight parameter assigned to this function and H_j^i is the Hamming distance between the input data and the data generated by f_i .

The main idea of their genetic evolutionary is similar to Algorithm 2.1. First, they parallelize the EA to generate n subpopulations where each subpopulation corresponds to one polynomial function for a target gene. Then using a roulette wheel selection to pick polynomial function for performing the crossover operator. For each target gene, a polynomial function is picked and all polynomials will be assembled to be a Boolean network. The number of assembled Boolean networks is limited by a predefined value. The next step is to conduct the mutate operator to the Boolean networks where polynomial functions are mutated by randomly changing their monomials. After that, the top fittest Boolean networks obtained from the mutation remain in the population and the assembled Boolean networks continue to exist in the population. They also integrated prior knowledge in the inference procedure. The prior knowledge can be used to reduce the size of the search space (Dorier et al., 2016).

Barman and Kwon (2018) have developed a genetic algorithm-based Boolean network inference (GABNI) method aims for inferring large-scale Boolean networks. In general, GABNI infers the regulatory rule for every target gene independently and then integrate all rules into a final Boolean network. Each target gene corresponds to a chromosome which includes not only the set of input genes but also the regulatory rule. The chromosome is represented by a binary vector of length n where i th bit denotes whether x_i is an input node or not. Since no Boolean function is explicitly represented in the chromosome, they specified the corresponding regulatory rule using the observed data. Denoting a set of K input genes expression at time point t $x(t), i \in \{1, \dots, K\}$ as $\mathbf{x}(t)$, then $y(t+1) = \arg \min_{b \in \{0,1\}} \Pr((x(t+1) = b) \mid \mathbf{x}(t) = \mathbf{b})$ where $\mathbf{b} = b_1 b_2 \dots b_k \in \{0, 1\}^K$. In this way, they defined the gene-wise dynamics consistency $C(x, y)$ to measure the difference between the output state y generated by the inferred Boolean function of x and the true state of x in the dataset, as follows:

$$C(x, y) = \frac{\sum_{t=2}^N I(x(t) = y(t))}{N - 1} \quad (2.5)$$

where N is the number of time points in a time series and $I(\cdot)$ is an indicator function that returns 1 if $x(t)$ equals $y(t)$, otherwise 0. Then the fitness of an individual can be computed

as follows:

$$Fitness = \frac{1}{(1 - C(x, y)) \times \gamma + K} \quad (2.6)$$

where K is the number of input genes included in the individual and γ is a weight factor that to make $C(x, y)$ as the primary or the secondary factor for the fitness function.

2.1.2 Feature selection methods

Apart from the optimization methods, the network inference problem can also be solved using feature selection techniques, which is scalable to large networks (Meyer et al., 2007). Barman and Kwon (2018) have proposed a novel algorithm called MIBNI to efficiently computing the multivariate mutual information based on a mutual information-based feature selection (MIFS) method (Battiti, 1994). MIFS was originally proposed for selecting an informative set of candidate features for supervised neural net learning. In the Boolean network inference problem, the MIFS method can be described in Algorithm 2.2.

The algorithm starts from a set $X_{selected}$ including an input gene x_j that maximizes the mutual information with the target gene x . Then next step is to include the variable that maximizes the $I(x; x_i) - \sum_{x_j \in X_{selected}} I(x_i; x_j)$ into $X_{selected}$ where $x_i \in X$. This step is repeated until the number of genes in $X_{selected}$ is equal to K , where K represents the number of input genes. To further infer a more accurate set of input genes, MIBNI applied the SWAP subroutine to the selected set $X_{selected}$ generated by MIFS. The SWAP is shown in Algorithm 2.3.

Given two sets $X_{selected}$ and W , the main idea of SWAP is to see whether swapping the gene in $X_{selected}$ and the gene W can increase the gene-wise dynamics consistency (see Equation 2.5). The *search_update_rule*($x, X_{selected}$) is used to find the specific regulatory rule of x that maximizes the gene-wise dynamics consistency. The genes in a regulatory rule can be associated with $\neg$ or not, there are $2^{|X_{selected}|+1}$ rules that can be inferred in SWAP

Algorithm 2.2: Mutual information-based feature selection method used in
(Barman and Kwon, 2018)

Input: The target gene x , a set of all genes $X = \{x_1, x_2, \dots, x_n\}$, the number of
input genes K

Output: A set of selected genes $X_{selected}$

```
1  $X_{selected} \leftarrow \emptyset;$ 
2  $x_j \leftarrow \arg \min_{x_i \in X} I(x; x_i);$ 
3  $X_{selected} \leftarrow X_{selected} \cup x_j;$ 
4  $X \leftarrow X \setminus \{x_j\};$ 
5 repeat
6    $x_h \leftarrow \arg \min_{x_i \in X} (I(x; x_i) - \sum_{x_j \in X_{selected}} I(x_i; x_j));$ 
7    $X_{selected} \leftarrow X_{selected} \cup x_h;$ 
8    $X \leftarrow X \setminus \{x_h\};$ 
9 until  $|X_{selected}| = K;$ 
10 return  $X_{selected}$ 
```

Algorithm 2.3: SWAP subroutine proposed by (Barman and Kwon, 2018)**Input:** The target gene x , the set of selected input genes $X_{selected}$ such that
$$I(x; x_j) \geq I(x; x_i) \text{ if } j < i \text{ for all } x_j, x_i \in X_{selected}, \text{ the set of unselected genes } W \text{ such that } I(x; w_j) \geq I(x; w_i) \text{ if } j < i \text{ for all } w_j, w_i \in W, \text{ the number of input genes } K$$
Output: A set of selected genes $X_{selected}$ and gene-wise dynamics consistency C_{MAX}

```

1  $y \leftarrow search\_update\_rule(x, X_{selected});$ 
2  $C_{MAX} \leftarrow C(x, y);$ 
3 for  $j = 1$  to  $|W|$  do
4   for  $i = 1$  to  $K$  do
5      $y \leftarrow search\_update\_rule(x, X_{selected} \cup \{w_j\} \setminus \{x_i\})$  where
        $w_j \in W, x_i \in X_{selected};$ 
6     if  $C(x, y) > C_{MAX}$  then
7        $X_{selected} \leftarrow X_{selected} \cup \{w_j\}, X_{selected} \leftarrow X_{selected} \setminus \{x_j\};$ 
8        $W \leftarrow W \cup \{x_i\}, W \leftarrow W \setminus \{w_i\};$ 
9        $C_{MAX} \leftarrow C(x, y)$ 
10 return  $X_{selected}$  and  $C_{MAX}$ 

```

as it considers the combinations using only $\wedge$ or $\vee$.

The overall framework of MIBNI is present in Algorithm 2.4. For each target gene x_i , it first check if the entropy value of x_i equals to 0. If $H(x_i) = 0$, then it implies that this gene has no regulatory genes. Otherwise MIFS looks for a set of genes $X_{selected}$ that includes K most informative genes. Next, SWAP optimizes this set by swapping the same number of genes between $X_{selected}$ and $X \setminus X_{selected}$. When it finds K input genes or $C_{MAX} = 1$, it outputs the regulatory rule for x_i .

Algorithm 2.4: Mutual information-based Boolean network inference

(MIBNI) algorithm proposed by Barman and Kwon (2018)

Input: $X = \{x_1, x_2, \dots, x_n\}$, the number of input genes K **Output:** the set of input genes $X_{selected}$ with a regulatory rule.

```

1  $i \leftarrow 1$ ;
2 while  $i \neq n$  do
3   if  $H(x_i) = 0$  then
4      $i \leftarrow i + 1$ ;
5   else
6      $k \leftarrow 1$ ;
7      $X_{selected} \leftarrow MIFS(x_i, X, k)$ ;
8      $(X_{selected}, C_{MAX}) \leftarrow SWAP(x_i, X_{selected}, X \setminus X_{selected})$ ;
9     if  $C_{MAX} = 1$  or  $k = K$  then
10      the regulatory rule for  $x_i$  is inferred;
11       $i \leftarrow i + 1$ ;
12    else
13       $k \leftarrow k + 1$ ;
14    skip to line 7;

```

Tree-based methods were also used for gene regulatory network inference. Irrthum et al. (2010) proposed GENIE3 based on feature selection with an ensemble of regression trees. The basic idea of GENIE3 is to formulate the inference problem as a feature selection problem. In general, for a target gene, the first step is to build a collection of regression trees (Loh, 2011) where each node corresponds to a candidate input gene. The regression trees grow under the random forest framework (Breiman, 2001) or Extra-Trees (Geurts et al., 2006). Then the second step is to rank importance features (i.e., genes) from the genes included in the ensemble of trees. We can see that although GENIE3 were not applied to

the time series data with binary values, their framework can still be extended to Boolean network inference through building classification trees.

2.1.3 Scoring Methodology

The assessment of network inference algorithms requires both artificial and real-world networks as benchmark. We will describe benchmark networks as well as their data simulation in the following chapters. In general, both synthetic datasets and experimental datasets are employed. The quality of a Boolean network model must be evaluated from both the topological and dynamical view.

First, to evaluate the quality of the network topology, one has to compare the inferred structure with the true network structure. Supposing that the true network structure is known, then the number of truly and falsely inferred edges can be counted. Specifically, the following numbers can be introduced:

- TP = the number of inferred interactions that are present in the ground truth network
- FP = the number of inferred interactions that are absent in the ground truth network
- TN = the number of absent interactions in the inferred network that are also absent in the ground truth network
- FN = the number of absent interactions in the inferred network that are present in the ground truth network

Then we can compute the following performance metrics that are commonly used to evaluate the quality of the inferred topology (Hecker et al., 2009; Maucher et al., 2011; Vera-Licona et al., 2014). $\text{Recall} = \frac{TP}{TP+FN}$ defines the fraction of correctly inferred edges out of all true edges, $\text{Precision} = \frac{TP}{TP+FP}$ denotes the fraction of correctly inferred edges in relation to

the number of all inferred edges. And there are other commonly used metrics False Positive Rate (FPR) = $\frac{FP}{FP+TN}$, Accuracy = $\frac{TP+TN}{TP+FP+TN+FN}$ and F-score = $\frac{2TP}{2TP+FP+FN}$.

Note that these performance scores cannot be used to distinguish the regulatory rules (e.g. inhibition and activation). To evaluate the regulatory rules, one can check whether the inferred Boolean model can correctly predict the dynamic behaviours of the ground truth network. To this end, we have to compare the inferred states with the observed states in the gene expression data. There are two ways to validate the inferred dynamics: one way is to retrieve the steady states using the inferred Boolean model and get the fraction of correctly identified steady states out of the number of all steady states (Vera-Licona et al., 2014). For a Boolean network with a set of steady state S , the dynamic accuracy (DA) can be evaluated by:

$$DA = \sum_{s_i \in S} I(s_i, F(s_i)) \quad (2.7)$$

where $F(s_i)$ is the output state predicted by the inferred Boolean function F and the input state s_i . $I(\cdot)$ is an indicator function that returns 1 if s_i and $F(s_i)$ are the same, otherwise 0.

Another way is to use a scoring function to compute the difference between two trajectories generated by the inferred network and the observed states and count the number of correctly predicted state of each gene in a time series (Barman and Kwon, 2017; Barman and Kwon, 2018). The DA is given as:

$$DA = 1 - \frac{\sum_{t=2}^N \sum_{i=1}^n (y_i(t) - x_i(t))^2}{n \times (N - 1)} \quad (2.8)$$

where N refers to the number of time points in a time series, $y_i(t)$ denotes the output state predicated by the inferred Boolean function of x_i with the input state at time point $t - 1$, $x_i(t)$ is the true state of the target gene at time point t and n is the number of nodes in the Boolean network.

2.2 Gene regulatory network control

2.2.1 Dynamics-based methods

The concept of control kernel was proposed by Kim et al. (2013), where the control kernel refers to the minimal set of genes that must be regulated to make any possible initial network states converge to a desired attractor. For finding the control kernel, they developed a method using a genetic algorithm (GA). The control strategies to force an arbitrary network state to a desired attractor are implied in a chromosome. The chromosome is a binary string $X = \{X_1, X_2, \dots, X_n\}$ where each X_i represents whether gene x_i is controlled $X_i = 1$ or not $X_i = 0$. Then the genes with value 1 in the final solution compose a candidate control kernel. Given a full state transition diagram which we have described in Section 1.1.3, the control actions are made by forcing the state of each gene marked with 1 in the candidate solution to the state of the same gene expressed in the desired attractor.

To find the best candidate control kernel, the GA searches for the best chromosome (i.e., a candidate gene set) that minimizes the fitness

$$fitness = \begin{cases} |B|^3 \times (n - \sum_{i=1}^n X_i)^2 \times 2, & \text{if } |B| = 1 \\ |B|^3 \times (n - \sum_{i=1}^n X_i)^2, & \text{otherwise} \end{cases} \quad (2.9)$$

where $|B|$ is the basin size of the desired attractor (i.e., the number of states included in the basin of the desired attractor) of the controlled network, n is the number of genes included in the network, X_i represents whether node x_i is selected (1) or not (0) for taking control actions. For the small-scale networks with the number of genes no more than 16, the whole state transition is constructed. Otherwise for the large-scale networks, random 10000 initial states are used to generate the state transition diagram.

The algorithm initializes with a population of random chromosomes. In each generation, one can construct the state transition diagram of the controlled network by changing

the state of selected genes according to the chromosome. Then the fitness of each chromosome is evaluated to investigate the convergence of all possible initial network states to the desired attractor. The next step is to utilize the fitness proportionate selection to pick pairs of chromosomes for applying crossover operators to generate offspring. The mutation operation is then applied to each chromosome in the offspring. Finally, the control kernel is the set of controlled genes encoded in the chromosome with the best fitness value. It is obvious to see that this method cannot guarantee the desired attractor is reachable from any possible states.

Besides using the genetic algorithm, Rafimanzelat and Bahrami (2018b) developed an algorithm based on the concept of the semi-tensor product. They used a forward selection method and the GA method to find the kernels for a given target attractor. And further, they investigated the set of genes that can make the Boolean network converge from any of its attractors to any other one (Rafimanzelat and Bahrami, 2018a). Hou, Ruan, et al. (2018) studied the expected number of controlled nodes theoretically by formulating the control problem to Coupon Collector's problem.

2.2.2 Structure-based methods

In recent work on structure controllability, some of standard control theory approaches have been proposed. For example, Liu et al. (2011) studied the problem of structural controllability in terms of graph theory and identified the minimum number of driver nodes (MDS). The driver nodes are a set of nodes that can be actuated to make the network from any initial state to reach a required final state in finite time. To understand the structural controllability, we consider the dynamic model:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \quad (2.10)$$

where $\mathbf{x}(t)$ is a state vector describing the state of all n gene, $\mathbf{A}$ is an $n \times n$ matrix that denotes the interactions between genes in a Boolean network, $\mathbf{B}$ is an $n \times m$ matrix where m are input control signals and $\mathbf{B}$ indicates which nodes are steering nodes, and $\mathbf{u}(t)$ is an input vector of m input control signals. Specifically, $a_{ij}, i \neq j$ in the matrix A denotes the strength of the influence of gene x_j on x_i .

The system described by Equation 2.8 is said to be controllable if and only if the $n \times nm$ controllability matrix

$$C = (B, AB, A^2B, \dots, A^{n-1}B) \quad (2.11)$$

has full rank of n .

To explain the structural controllability from a graph theoretical point of view, we build a graph $\mathbf{G}(\mathbf{A})$ which includes the nodes $X_A = \{x_1, x_2, \dots, x_n\}$ and edges that are indicated in the matrix $\mathbf{A}$. Then we can have a graph $\mathbf{G}(\mathbf{A}, \mathbf{B})$ including two sets of nodes and $X_A \cup X_U$ where $X_U = \{u_1, u_2, \dots, u_m\}$. The X_U denotes the input nodes in terms of the matrix $\mathbf{B}$. Edges in $\mathbf{G}(\mathbf{A}, \mathbf{B})$ from a node u_i in X_U to nodes in X_A correspond to the

i th column in the matrix $\mathbf{B}$. For instance $\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 \\ a_{21} & 0 & a_{23} \\ 0 & a_{32} & 0 \end{bmatrix}$ and $B = \begin{bmatrix} b_{11} \\ 0 \\ 0 \end{bmatrix}$ correspond

to the graph shown in Figure 2.1.

Theorem 1 *The network $\mathbf{G}(\mathbf{A}, \mathbf{B})$ is completely structurally controllable if and only if (Lin, 1974):*

1. *the network $\mathbf{G}(\mathbf{A}, \mathbf{B})$ contains no dilation and*
2. *All nodes are accessible from X_U*

Now we consider Theorem 1 (Lin, 1974), which holds clear graph-theoretic interpretations to structural controllability. The node is accessible means there is only one directed

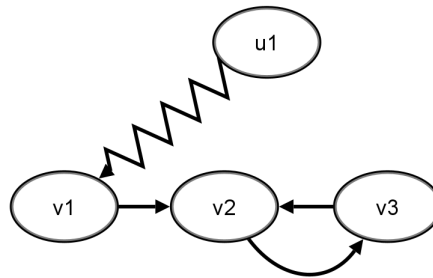


Figure 2.1: Network $\mathbf{G}(\mathbf{A}, \mathbf{B})$ correspond to the system $(\mathbf{A}, \mathbf{B})$

path from X_U . And a dilation exists if and only if the number of nodes in a subset X_S of X_A is greater than a node set X_{NS} where X_{NS} contains all neighbourhood nodes of X_S that point to X_S in the digraph. Intuitively, each node must have its own “superior” node so that each node can be accessible independently. Then to find the minimum number of driver nodes to fully control a network, Liu et al. (2011) formulated the problem to the maximum matching problem. The results showed that if there is a perfect matching in $\mathbf{G}(\mathbf{A})$, the minimum number of driver nodes is one. Otherwise, it equals the number of unmatched nodes.

Following their work, there are many other methods proposed to study the network control problem based on structural controllability. For the purpose of control design, Nacher and Akutsu (2012) investigated the relation between the size of the minimum dominating set and structural controllability. Gao, Liu, et al. (2014) proposed to use ‘k-walk theory’ and a greedy algorithm for identifying the minimum set of driver nodes to control a specified target set of nodes. However, their methods cannot be directly used to highly nonlinear Boolean networks (Hou, Tamura, et al., 2016).

Yin and Zhang (2016) have also shown that the MDS are not sufficient to fully control if the network contains inaccessible strongly connected components (SCCs) that have a perfect matching. They then proposed a linear integer programming method to identify

the minimum set of controlled nodes. Wu et al. (2016) also took account SCCs and further transformed control problem as a minimum-cost maximum flow problem. They first introduced the source node, sink node, and strongly connected components to build a flow network. Then they combined Hopcroft–Karp algorithm (Hopcroft and Karp, 1973) and Ford–Fulkerson algorithm (Ford Jr and Fulkerson, 2015) to solve it.

2.3 Summary

This chapter has presented existing works on Boolean network inference and control. We first introduce a series of works on Boolean network inference. The Boolean network inference problem can be formulated as an optimization problem where the different optimization algorithms (e.g., meta-heuristic) have been developed. It can be also transformed into a feature selection problem where the feature corresponds to every single gene. We have found that many algorithms focus on inferring the network topology (i.e., the interactions between genes). And a few of them infer both network topology and its dynamics simultaneously. Later we introduce the scoring methodology that is commonly used to evaluate the algorithms. Then we present the corresponding performance matrices used to measure the quality of the inferred network structure. In addition, we introduce two ways to quantify the quality of the dynamical behaviour of the inferred Boolean functions. Last we present a few algorithms that identify the minimum set of genes required for steering the Boolean network to a desired attractor.

Chapter Three

Inferring Boolean networks through And/Or tree ensemble

In the last chapter, we have discussed the inference algorithm for gene regulatory networks. In this chapter, we describe an algorithm based on And/Or tree ensemble (i.e., a collection of And/Or trees) for inferring accurate Boolean network topology and dynamics from short and noisy time series data (i.e., time series data with a small number of samples and low sampling frequency). Section 3.1 presents the motivation and the main contributions of our proposed algorithm. Section 3.2 introduces the And/Or tree representation of Boolean functions and describes the details of our proposed algorithm. Section 3.3 discusses the experimental results where we validate our algorithm on both artificial networks and real-world gene regulatory networks. Section 3.4 summarizes the chapter.

3.1 Introduction

As we introduced in the previous chapter, the network inference problem can be formulated as an optimization problem or feature selection problem. Most of the algorithms focus on inferring the topological structure of the network, e.g, Mendoza and Bazzan (2011) uses

a genetic algorithm to find the most consistent network structure. But it is necessary to investigate how dynamic process changes by the interactions in the network (Albert, 2004).

Although we have also presented several algorithms that can infer both the Boolean network topology and its dynamics. However, there are still shortcomings in existing algorithms. The exhaustive search has been first used in Liang et al. (1998) and Lähdesmäki et al. (2003) for searching all possible combinations of the input genes, thus they are not efficient in inferring the regulatory rules when there exist a large number of genes. To reduce the computational cost, Barman and Kwon (2017) have designed a Boolean inference algorithm named MIBNI for efficiently selecting the most informative input genes for a target node based on mutual information. The selected input genes are used to infer the putative Boolean functions. The algorithm REACT (Vera-Licona et al., 2014) uses Boolean polynomials to represent Boolean functions and applies population-based heuristic optimization algorithm to search the best candidate network that minimizes the fitness score. However, they only included limited regulatory rules in the inference procedure.

On the other hand, the quality of the inferred network depends on the quality of the experimental data (Hecker et al., 2009). It is still challenging to infer accurate Boolean networks from experimental data, which is typically noisy with a small number of time points (Bar-Joseph, 2004; Aalto et al., 2020). In addition, the network inference problem also refers to the “large p, small n” (i.e., the number of input genes is larger than the number of time points) problem that impedes the inference of Boolean networks. To address those problems, we propose an And/Or tree ensemble (ATEN) algorithm for inferring both the Boolean network topology and its dynamics. Each node in the Boolean network corresponds to a gene in the gene regulatory network. The main idea of our algorithm is that, for each target node, we build a collection of And/Or trees to select prime implicants of which each is a conjunction (AND) of a set of non-redundant input nodes. The selected prime implicants are important features for predicting the states of the target node. Using these important features we then infer the Boolean function of the target node. Finally, the Boolean functions

of all target nodes are combined as a Boolean network. Using the data generated from artificial and real-world gene regulatory networks, we show that our algorithm can infer more accurate Boolean network topology and dynamics from short and noisy time series data than other algorithms.

The advantages of our ATEN algorithm are as follows. First, ATEN is efficient because And/Or trees provide an efficient representation for inferring prime implicants using binary time series data. Second, by using an ensemble algorithm and feature selection, ATEN can better handle the “large p small n” problem (Genuer et al., 2010) to select more reliable important features (Irrthum et al., 2010), i.e., prime implicants. Third, ATEN introduces the sub-sampling and ensemble process to deal with small-scale and noisy datasets. Fourth, based on the selected features, our algorithm can infer a minimal Boolean function without redundant input nodes, which results in a more accurate Boolean network.

To summarize, our main contributions are:

1. We propose the first And/Or tree ensemble algorithm. Although And/Or trees have been widely used to search for solutions in combinatorial and artificial intelligence (Marinescu and Dechter, 2009), no And/Or tree ensemble algorithm exists to search for robust consensus solutions. We are the first to bridge the gap between ensemble learning and And/Or trees.
2. We apply our And/Or tree ensemble algorithm to select important prime implicants as features, which is different from most existing algorithms, e.g., GENIE3 (Irrthum et al., 2010) that select individual input nodes for each target node.
3. We validate ATEN on both artificial and real-world gene regulatory networks. Our experimental results show that ATEN compares favourably with existing algorithms in terms of prediction accuracy of network topology and dynamics, even when the gene expression time series data is short and noisy.

3.2 Methods

3.2.1 And/Or trees representation of Boolean functions

To represent a Boolean function associated with input nodes and Boolean/logical operators, we can use a logical formula in disjunctive normal form (DNF). A DNF formula is a disjunction (OR) of one or more implicants, where an implicant pt is the conjunction (AND) of literals (i.e., Boolean variables or their negations). An implicant pt is called a prime implicant if the deletion of any literal from pt results in a non-implicant. For example, a Boolean function $(x_1 \wedge x_2) \vee (x_3 \wedge \neg x_4 \wedge x_5)$ for the target node x_1 represents two prime implicants $(x_1 \wedge x_2)$ and $(x_3 \wedge \neg x_4 \wedge x_5)$, where x_1, x_2, x_3, x_4, x_5 are input nodes. In addition, we are interested in obtaining a minimal Boolean function which includes a minimum set of prime implicants by eliminating the redundant prime implicants and input nodes. For example, a Boolean function $(\neg x_1 \wedge x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge \neg x_3) \vee (x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (x_1 \wedge \neg x_2 \wedge x_3)$ can be minimized to $(\neg x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2)$.

A Boolean/logic function, or more specifically, a DNF formula can be represented as a three-level And/Or tree, which provides an efficient data structure to fit the binary time series data. In such a tree, the root node represents logical operator $\vee$, other internal nodes represent logical operator $\wedge$, and a leaf denotes a Boolean variable or its negation (i.e., the Boolean variable associated with $\neg$). The output of a Boolean function is determined by the valuation of the And/Or tree with an assignment of "TRUE (1)" or "FALSE (0)" to each of the leaves. In Figure 3.1, we show an example of a three-level And/Or tree l which is equivalent to the DNF formula $(x_1 \wedge x_2) \vee (x_3 \wedge \neg x_4 \wedge x_5)$.

We can represent a Boolean function in DNF as a three-level And/Or tree as follows:

- (First Level) The root is labelled by the logical operator $\vee$;

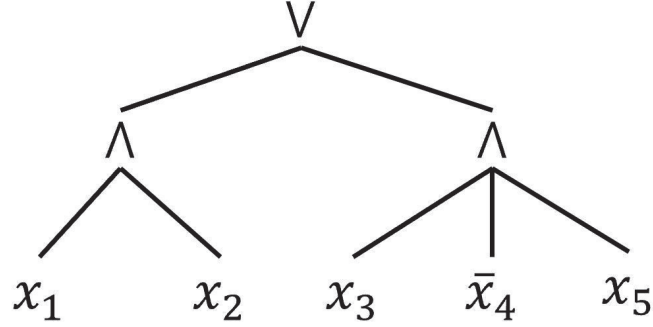


Figure 3.1: And/Or tree example. An example of a three-level And/Or tree with 5 leaves that represents a DNF formula $(x_1 \wedge x_2) \vee (x_3 \wedge \neg x_4 \wedge x_5)$.

- (Second Level) The internal node is labelled by the logical operator $\wedge$;
- (Third Level) A leaf is labelled by a Boolean variable, or by a negation (i.e., a Boolean variable associated with $\neg$);

3.2.2 Algorithm

Algorithm 3.1: ATEN for Boolean network inference

Input: A binary time series dataset with n genes.

Output: A Boolean network.

- 1 **foreach** *target node* x_i , $1 \leq i \leq n$ **do**
 - 2 Infer a set of prime implicants using Algorithms 3.3;
 - 3 Select a set of important prime implicants using Algorithm 3.4;
 - 4 Infer the final associated Boolean function using Algorithm 3.2.
 - 5 Combine all Boolean functions of all target nodes as a Boolean network;
-

To infer an accurate Boolean network from short and noisy gene expression time series data, our And/Or tree ensemble (ATEN) algorithm consists of three key steps as summarized in Algorithm 3.1. For each target node, the first step is to infer a set of prime implicants based

on an ensemble of And/Or trees. Each tree in the ensemble represents a putative Boolean function of the target node. The second step is to select and reconstruct a set of prime implicants as important features using a recursive feature reconstruction and elimination procedure (see Algorithm 3.3 and 3.4). The third step is to infer a minimal Boolean function without redundant prime implicants using the selected features (see Algorithm 3.2). The Boolean functions of all target nodes are finally combined to represent the Boolean network.

In addition, ATEN can incorporate prior knowledge (e.g., the number of input nodes for target nodes, the known input nodes), which reduces the search space and might improve inference accuracy (Studham et al., 2014). The ways to deal with the prior knowledge are detailed in the following subsections.

3.2.2.1 And/Or tree inference

We summarize the And/Or tree inference algorithm in Algorithm 3.2. Given binary time series data including N time points and n genes, ATEN aims at inferring an And/Or tree that best predicts the state of the target node x_i , $1 \leq i \leq n$. The inference is achieved by searching the optimal input features (i.e., single genes or prime implicants) and logical operators of the And/Or tree using a simulated annealing (SA) algorithm (Van Laarhoven and Aarts, 1987).

Specifically, the And/Or tree of a target node is initialized with an input feature (e.g., a random gene, or a random prime implicant, or known prime implicants). Then at each iteration r of the SA algorithm, a new tree l_r is proposed by randomly executing a permissible move based on the old tree l_{r-1} . The fully permissible move set includes:

- Growing a subtree: We allow a new subtree to grow. For instance, the tree $(x_1 \wedge x_2) \vee (x_3)$ can grow up to $(x_1 \wedge x_2) \vee (x_3) \vee (x_5)$ by appending a new subtree. Notice that this move is probably not allowed if the prior knowledge is given to limit the maximum

Algorithm 3.2: And/Or tree inference for a target node

Input: Binary time series data.

Output: An And/Or tree l_r of the target node.

```

1  $r = 0$ ;
2 Initialize an And/Or tree  $l_0$  using an input feature;
3 repeat
4    $r \leftarrow r + 1$ ;
5    $l_r \leftarrow$  Update  $l_{r-1}$  to propose a new tree;
6   if  $l_r$  is rejected according to the probability  $Pr_{l_r}$  then
7      $l_r \leftarrow l_{r-1}$ ;
8 until  $r$  reaches the maximum iteration;

```

number of input nodes.

- **Deleting a subtree:** We allow the deletion of a subtree. For instance, the tree $(x_1 \wedge x_2) \vee (x_3)$ turns to be x_3 by deleting the subtree $(x_1 \wedge x_2)$. If the prior knowledge provides a fixed subtree, then this subtree would not be changed.
- **Alternating a leaf:** We allow the replacement of a leaf. The leaf can be replaced with another Boolean variable or the negation of another Boolean variable. The tautology is prohibited by avoiding replacing the leaf with itself or its negation. For instance, the leaf x_2 in the tree $(x_1 \wedge x_2) \vee (x_3)$ can be replaced by x_4 , and cannot be replaced by $\neg x_1$. In addition, if the leaf refers to a known input node, then it would not be replaced with another one.
- **Deleting a leaf:** We allow to delete a leaf. For instance, x_1 or x_2 can be deleted from the tree $(x_1 \wedge x_2) \vee (x_3)$. And in the case when a leaf is also a subtree (e.g. x_3 in the instance), we do not allow to delete it in this deletion-move. We also do not allow to delete known input node according to the prior knowledge.

- Adding a leaf: We allow to add a leaf in a subtree. This move is done by appending a leaf to an existing subtree. For instance, the tree $(x_1 \wedge x_2) \vee (x_3)$ turns to be $(x_1 \wedge x_2) \vee (x_3 \wedge x_5)$ by appending the leaf x_5 to a subtree. This move also depends on the prior knowledge if the maximum number of input nodes is provided.

The acceptance probability for the new tree l_r is determined by the misclassification rates of l_{r-1} and l_r and the temperature T specified in the SA cooling scheme, i.e., $Pr_{l_r} = \min\{1, \exp((E_{l_{r-1}} - E_{l_r})/T)\}$, where $E_{l_{r-1}}$ and E_{l_r} are the misclassification rates of l_{r-1} and l_r respectively. The misclassification rate E_{l_r} of l_r is calculated by

$$E_{l_r} = \frac{1}{N-1} \sum_{t=2}^N I(y_{l_r}(t-1) \neq x_i(t)),$$

where $I(\cdot)$ is an indicator function $I(\cdot) = \begin{cases} 1, & \text{if } y_{l_r}(t-1) \neq x_i(t) \\ 0, & \text{if } y_{l_r}(t-1) = x_i(t) \end{cases}$, and $y_{l_r}(t-1)$ is the output state predicated by l_r based on the states of the input nodes at time point $t-1$, and $x_i(t)$ is the true state of the target node at time point t , $2 \leq t \leq N$. The acceptance probability indicates: 1) if l_r is better than l_{r-1} (i.e., $E_{l_r} < E_{l_{r-1}}$), then l_r is always accepted; 2) if l_r is not better than l_{r-1} (i.e., $E_{l_r} > E_{l_{r-1}}$), then l_r is accepted with a very small probability which gradually converges to zero as the temperature decreases. The maximum number of iterations is predefined to stop the searching scheme. The tree size which is defined as the number of input nodes in the tree can be set according to prior knowledge, i.e., the maximum in-degree of the network, or determined experimentally or empirically. In addition, if the interactions between some pairs of nodes are known, ATEN can include those interactions as fixed prime implicants when inferring the tree.

The finally accepted And/Or tree may contain redundant regulatory interactions (i.e., prime implicants) between the target node and input nodes. The redundant prime implicants are unnecessary for predicting the state of the target node. Thus we introduce a Boolean minimization method aimed at obtaining a minimal And/Or tree (i.e., a minimal DNF)

based on the Quine-McCluskey algorithm (Quine, 1955; McCluskey, 1956), where the Quine-McCluskey algorithm tries to find essential prime implicants and other necessary prime implicants to simplify Boolean functions.

Specifically, given a Boolean function, we first construct a truth table of the Boolean function. Then we can get a group of minterms where each minterm corresponds to a row of the function’s truth table whose output value is 1. Secondly, we arrange minterms in an ascending order and group them based on the number of ones present in their binary representations (for example, minterms 0010 and 1000 contain three zeros are grouped together). Thirdly, we compare all pairs of minterms in adjacent groups, combine the pair of those two minterms that there is a change in only one-bit position, and replace ‘—’ in the differed bit position and keep the remaining bits as it is (for example, minterms 0010 and 0110 are combined to be 0-10). We repeat the third step till all prime implicants are found. After that, we construct prime implicant table where the prime implicants are placed in row wise and minterms are placed in column wise. And in this new table, we place 1 to the common cells corresponding to the minterms that are covered in each prime implicant. Next we look for the minterm that is covered only by one prime implicant, then we identify it as essential prime implicant. Finally, all essential prime implicant compose the simplified Boolean function.

3.2.2.2 Prime implicant importance measure

We select a set of important prime implicant to build the final And/Or tree (i.e., the Boolean function). To this end, we compute the importances of prime implicants as shown in Algorithm 3.3 based on logicFS (Ruczinski et al., 2003; Schwender and Ickstadt, 2008). The main idea is to construct an And/Or tree ensemble to quantify how much a prime implicant improves the predictive accuracy of the state of the target node. As shown in Figure 3.2, given a binary time series dataset D , for a target node, we draw G pairs of a bootstrap

sample IB_g and an out-of-bag (oob) sample OB_g , $1 \leq g \leq G$ for inferring ensembles of trees and computing the importances of prime implicants, respectively.

Formally, given J time series covering N time points each, the j th ($1 \leq j \leq J$) time series can be presented as a matrix of states of all genes, denoted as

$$\mathbf{X}^j = (\mathbf{x}^j(1), \dots, \mathbf{x}^j(t), \dots, \mathbf{x}^j(N)),$$

where $\mathbf{x}^j(t) = (x_1^j(t), x_2^j(t), \dots, x_n^j(t))^T$ is a vector containing the states of all n genes at time point t . We then construct the binary time series dataset matrix D by concatenating the J time series, i.e.,

$$D = \{\mathbf{X}^1, \dots, \mathbf{X}^j, \dots, \mathbf{X}^J\}.$$

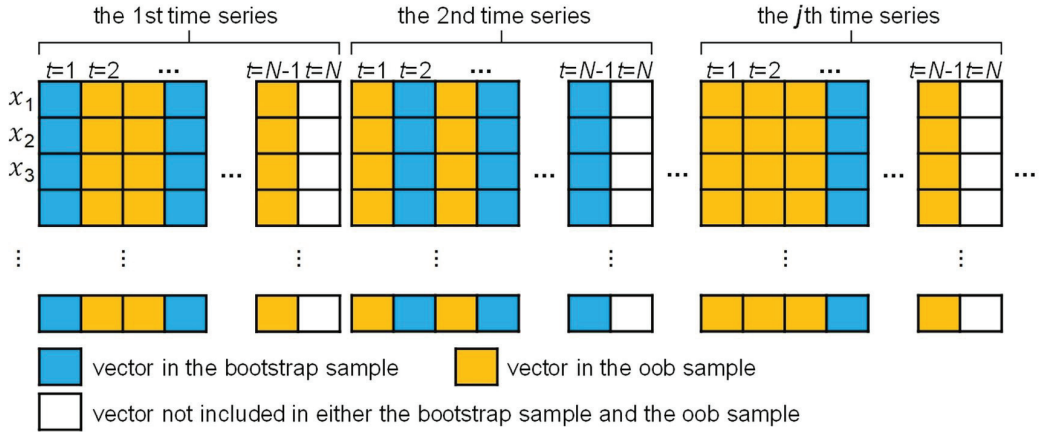


Figure 3.2: Illustration of bootstrap samples and the corresponding out-of-bag (oob) samples. Each column represents an expression vector $\mathbf{x}^j(t)$. Each row denotes an input feature (i.e., a gene or a prime implicant). For each target node, we generate a set of pairs of a bootstrap and an oob sample. Bootstrap samples are used to infer ensembles of trees. Oob samples are used to compute the importances of prime implicants based on the tree ensemble.

We draw a bootstrap sample by random sampling D with replacement, which is done by randomly selecting a column from D . This sampling procedure stops when the bootstrap samples IB_g reaches the size of $J * (N - 1)$. We also obtain a corresponding oob sample OB_g , i.e., $OB_g = \{\mathbf{x}^j(t) | \mathbf{x}^j(t) \in D, \mathbf{x}^j(t) \notin IB_g, t \neq N\}$. Note that vectors $\mathbf{x}^j(N), \forall j$ are

not included in both bootstrap samples and oob samples, because the true states of the target node recorded in $\mathbf{x}^j(N+1)$ are unknown for estimating the misclassification rate of an inferred tree. Notice that, our sampling process also works for the time series that have different number of time points.

We use the G bootstrap samples in IB_g as inputs of Algorithm 3.2 to get ensembles of G And/Or trees. From the And/Or tree ensembles, we then extract all distinct prime implicants denoted by PT . For each prime implicant pt_i in PT , we compute its importance based on pt_i 's presence or absence over every And/Or tree l_g in the ensemble as shown in Algorithm 3.3. For each pt_i , if pt_i is present in an And/Or tree l_g , we compute the misclassification rate $E_{l_g^-}$ of the new tree l_g^- obtained by removing pt_i from l_g . On the other hand, if pt_i is absent in the l_g , we compute the misclassification rate $E_{l_g^+}$ of the new tree l_g^+ obtained by appending pt_i to the l_g as a prime implicant (i.e., a new subtree). In the end, the importance of pt_i is determined by $VI(pt_i) = \sum_{g=1}^G \left(\sum_{pt_i \in l_g} (E_{l_g^-} - E_{l_g}) + \sum_{pt_i \notin l_g} (E_{l_g} - E_{l_g^+}) \right)$.

3.2.2.3 Recursive feature reconstruction and elimination

We propose a recursive feature reconstruction and elimination (RFRE) procedure for selecting a set of important prime implicants as features, which will be finally used for inferring Boolean functions. The main idea is to recursively reconstruct the selected important prime implicants, and eliminate the non-important ones. We summarize RFRE in Algorithm 3.4.

Initially, we apply Algorithm 3.3 to a binary time series dataset matrix D containing J time series covering N time points each, which results in a set of prime implicants PT_0 with their importances. We then eliminate the least important prime implicants, i.e., prime implicants with negative importance and the 20% of prime implicants with the lowest importances.

Then at each recursion, we first update the time series dataset matrix D_{r-1} based on

Algorithm 3.3: Compute the importances of prime implicants of a target

node

Input: A binary time series dataset D .

Output: The prime implicants with their importances

Initialization: The number of bootstrap samples G ;

```

1 Generate bootstrap samples  $IB_g$  by drawing  $G$  samples ;
2 foreach  $IB_g, 1 \leq g \leq G$  do
3    $l_g \leftarrow$  Infer an And/Or tree using Algorithm 3.2;
4    $OB_g \leftarrow$  Generate an oob sample;
5  $PT \leftarrow$  Extract all distinct prime implicants  $pt_i, i = 1, 2, \dots$ ;
6 foreach  $pt_i \in PT$  do
7   foreach  $l_g$  do
8      $E_{l_g} \leftarrow$  Compute the misclassification of  $l_g$  using  $OB_g$ ;
9      $l_g^- / l_g^+ \leftarrow$  Remove/Append  $pt_i$  from/to  $l_g$ ;
10     $E_{l_g^-} / E_{l_g^+} \leftarrow$  Compute the misclassification of  $l_g^- / l_g^+$  using  $OB_g$ ;
11    Determine the importance of  $pt_i$  by
        
$$VI(pt_i) = \sum_{g=1}^G \left( \sum_{pt_i \in l_g} (E_{l_g^-} - E_{l_g}) + \sum_{pt_i \notin l_g} (E_{l_g} - E_{l_g^+}) \right).$$


```

the important prime implicants:

$$D_{r-1} = \{\mathbf{PT}_{r-1}^1, \dots, \mathbf{PT}_{r-1}^j, \dots, \mathbf{PT}_{r-1}^J\},$$

where $\mathbf{PT}_{r-1}^j$ denotes the j th time series dataset matrix of all prime implicants, i.e.,

$$\mathbf{PT}_{r-1}^j = (\mathbf{pt}_{r-1}^j(1), \dots, \mathbf{pt}_{r-1}^j(t), \dots, \mathbf{pt}_{r-1}^j(N)).$$

$\mathbf{pt}_{r-1}^j(t)$ is a vector containing the states of all prime implicants in PT_{r-1} at time point t . We then apply Algorithm 3.3 to D_{r-1} to reconstruct the prime implicants. This step results in a set of important prime implicants, i.e., PT_r , which possibly contain some new ones (i.e., PT_{new}) that are reconstructed from input prime implicants.

The RFRE procedure stops when the newly reconstructed prime implicants are not important. Specifically, we compare $VI(pt_v)$ to $VI(pt_w)$, where $pt_v \in PT_{old}$ and $pt_w \in PT_{new}$. If we cannot find any new prime implicants (i.e., pt_w) that are more important than the input prime implicants remained in PT_r (i.e. pt_v), we terminate the recursion and return a set of prime implicants for inferring the final Boolean function.

3.3 Computational experiments

3.3.1 Datasets

3.3.1.1 Artificial network

We first validated the performance of ATEN by inferring three artificial Boolean networks generated by BoolNet (Müssel et al., 2010), where three networks include 50 nodes, 100 nodes, and 150 nodes, respectively. The maximum in-degree of each network is set to 5 (i.e., the maximum number of input nodes for each target node is 5). For each network, we generated 12 datasets $D_i, 1 \leq i \leq 12$ where each dataset includes 10 time series. Each time

Algorithm 3.4: RFRE for prime implicants selection

Input: A binary time series dataset D .

Output: A small set of prime implicants.

```

1  $PT_0 \leftarrow$  Apply Algorithm 3.3 to  $D$  and eliminate the non-important prime
   implicants
2  $r \leftarrow 1$ ;
3 repeat
4    $D_{r-1} \leftarrow$  Update a dataset matrix based on  $PT_{r-1}$ ;
5    $PT_r \leftarrow$  Apply Algorithm 3.3 to  $D_{r-1}$  and eliminate the non-important prime
     implicants;
6    $PT_{new} \leftarrow PT_r \setminus PT_{r-1}$ ;
7    $PT_{old} \leftarrow PT_r \setminus PT_{new}$ ;
8    $r \leftarrow r + 1$ ;
9 until  $VI(pt_v) > VI(pt_w), \forall pt_v \in PT_{old}, \forall pt_w \in PT_{new}$ ;
```

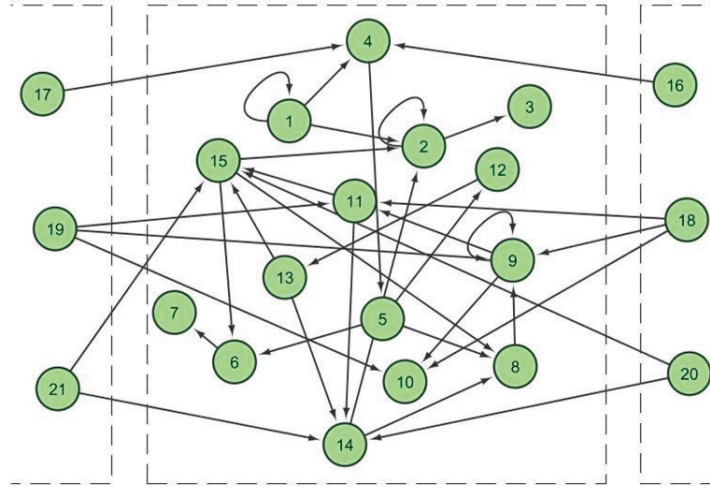


Figure 3.3: Part of the topology of a single parasegment primordium network. The whole network is an interconnected network of four identical cells. Nodes located inside the dashed rectangle are in one cell. Nodes 16-21 are in the adjacent cells. The directed edges represent the existence of regulation without implying the activation or inhibition.

series in $D_i(i = 1, 2, 3)$, $D_i(i = 4, 5, 6)$, $D_i(i = 7, 8, 9)$ and $D_i(i = 10, 11, 12)$ has 5, 10, 15, 20 time points, respectively. Additionally, we added 1% and 5% noise to $D_i(i = 2, 5, 8, 11)$ and $D_i(i = 3, 6, 9, 12)$, respectively. The noise was introduced by randomly flipPTng the state of each node with the probability of 1% or 5%.

3.3.1.2 Real-world network

We also applied ATEN to infer a real-world gene regulatory network using experimentally-observed gene expression data. The network is derived from a well-studied *Drosophila* segment polarity gene regulatory network (Albert and Othmer, 2003) and modified by REACT (Vera-Licona et al., 2014). It is considered as a single parasegment primordium of four identical cells (Albert and Othmer, 2003; Laubenbacher and Stigler, 2004). As shown in Figure 3.3 and Table 3.1, we present part of the network topology and its dynamic model (i.e., Boolean functions), respectively. The network of one cell consists of 15 nodes and 6

extracellular nodes, where the 6 extracellular nodes are connected with other nodes (not shown in Figure 3.3) in the adjacent cells. In the experiment, the whole network of a single parasegment primordium which consists of in total $15 \times 4 = 60$ nodes was used for network inference.

For the dataset used for inferring the real-world gene regulatory network, we introduced three datasets that were used in REACT. The datasets were initialized using experimentally observed gene expression data (Albert and Othmer, 2003) and knockout perturbations. These three datasets included 0%, 1%, 5% data noise, respectively. Each dataset has 24 time series containing 202 time points in total. Moreover, we used the same initialization to generate 9 additional datasets $D_i, 1 \leq i \leq 9$, where each dataset has 10 time series. Each time series in $D_i (i = 1, 2, 3)$, $D_i (i = 4, 5, 6)$ and $D_i (i = 7, 8, 9)$ has 5, 10, 15 time points, respectively. We also added 1% and 5% noise into $D_i (i = 2, 5, 8)$ and $D_i (i = 3, 6, 9)$, respectively.

3.3.1.3 Experimental design

We compared ATEN with the other three existing network inference algorithms, i.e., MIBNI (Barman and Kwon, 2017), Best-Fit (Lähdesmäki et al., 2003) and REACT (Vera-Licona et al., 2014). Because ATEN uses a heuristic algorithm, we executed ATEN 30 times with $g = 100$ on every dataset. Due to the unavailable implementation of REACT, we adopted the results from the literature (i.e., (Vera-Licona et al., 2014)) for comparison.

For inferring the artificial network, we compared the performance of ATEN with that of Best-Fit and MIBNI. For inferring the real-world gene regulatory network, we compared ATEN with MIBNI, Best-Fit and REACT.

One important control parameter of ATEN is the upper limit on tree size, i.e., the maximum number of input nodes of a target node. This parameter essentially defines the

Table 3.1: Boolean functions included in the part of a single parasegment primordium network

Target Node	Boolean function
X_1	X_1
X_2	$(\neg X_{15} \wedge X_{14} \wedge X_2) \vee [(X_2 \wedge (X_{14} \vee X_1)) \wedge (\neg X_{15})]$
X_3	X_2
X_4	$\neg X_1 \wedge (X_{16} \vee X_{17})$
X_5	X_4
X_6	$(\neg X_{15} \wedge X_5)$
X_7	X_6
X_8	$\neg X_{15} \wedge \neg X_5 \wedge X_{14}$
X_9	$\neg X_8 \vee (X_9 \wedge \neg X_{18} \wedge \neg X_{19})$
X_{10}	$((X_{18} \vee X_{19}) \wedge X_9)$
X_{11}	$(\neg X_9 \vee X_{18} \vee X_{19})$
X_{12}	$\neg X_5$
X_{13}	X_{12}
X_{14}	$X_{13} \wedge (X_{11} \vee X_{20} \vee X_{21})$
X_{15}	$\neg X_{11} \wedge \neg X_{20} \wedge \neg X_{21} \wedge X_{13}$

Table 3.2: Steady states of the part of a single parasegment primordium network.

Steady States
s1=(000111100010000110000)
s2=(011000011111110000101)
s3=(000000001001101000000)
s4=(000111100010000100000)
s5=(000111100010000100101)
s6=(000000011111110000101)

search space of the problem, which affects not only the prediction accuracy but also the computational time of ATEN. However, finding a suitable value of this parameter is challenging not only for ATEN but also for other algorithms (Barman and Kwon, 2017). To determine the value of this parameter, we executed experiments on an artificial network with 150 nodes (See Supplementary Data). Based on the experimental results, we set the upper limit on tree size to 8. For a fair comparison, MIBNI and Best-Fit were also limited to infer up to 8 input nodes for each target node. Additionally, we also presented the results with the known maximum number of input nodes of the artificial and real-world networks. More details can be found in the Supplementary Data. Except for a fixed K for network inference, there are other control parameters used in ATEN. They mainly affects the computational complexity of our ATEN, such as R , the maximum number of iterations of the Simulated Annealing algorithm. In ATEN, we essentially multiply the work of growing an And/Or tree by G to get and measure npt ($npt = |PT|$) prime implicants, where each tree examines $J \cdot (N - 1)$ time points. It would take $\mathcal{O}(G \cdot R \cdot J \cdot N + npt \cdot G \cdot J \cdot N)$ time in the worst case. Next we recursively apply this procedure to get new prime implicants no greater than R' times to get the final prime implicants. However their effects on prediction accuracy are not as critical as the upper limit on tree size based on the general guideline on how to set these parameters in our R package (available at <https://github.com/ningshi/ATEN>).

To quantify the quality of the inferred network topology, we used the performance metrics described in Section 2.1.3, namely $\text{Recall} = \frac{TP}{TP+FN}$, False Positive Rate (FPR) $= \frac{FP}{FP+TN}$ and F-score $= \frac{2TP}{2TP+FP+FN}$. To evaluate the quality of the dynamic model of the inferred networks, we used 6 steady states that have been investigated in existing literature as detailed in (Vera-Licona et al., 2014) in Table 3.2. We checked whether the Boolean functions inferred by each algorithm can retrieve these steady states using Equation 2.7.

Table 3.3: Experimental results of ATEN, MIBNI and Best-Fit for the artificial networks with 50 nodes inferred from the noise-free data. The maximum in-degree of each target node inferred by ATEN, MIBNI and Best-Fit was limited to 8. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.

Number of Time Points	Algorithm	Noise 0%		
		Recall	FPR	F-score
50	ATEN	0.941±0.023	0.004 ±0.002	0.892 ±0.026
	MIBNI	0.813	0.143	0.224
	Best-Fit	1	0	1
100	ATEN	0.930±0.014	0.002 ±0.001	0.890±0.026
	MIBNI	0.797	0.143	0.220
	Best-Fit	1	0	1
150	ATEN	0.983 ±0.022	0.003±0.004	0.944±0.051
	MIBNI	0.797	0.014	0.220
	Best-Fit	1	0	1
200	ATEN	0.984 ±0.002	0.002±0.001	0.961±0.011
	MIBNI	0.797	0.071	0.220
	Best-Fit	1	0	1

3.3.2 Results

3.3.2.1 Performance on artificial network

We summarize experimental results from ATEN, Best-Fit and MIBNI on the artificial networks with 50, 100, 150 nodes inferred from noise-free data in Table 3.3, Table 3.4 and Table 3.5. For the network consisting of 50 nodes as shown in Table 3.3, we can see that when the data is noise-free, Best-Fit always outperformed ATEN and MIBNI. Best-Fit also gained 100% Recall, 0% FPR and 100% F-score which means Best-Fit correctly predicted all edges that present in the benchmark and did not infer even one more edge. That might because Best-Fit enumerated all possible Boolean functions and finally identified a Boolean network that perfectly fit the dataset. ATEN was worse than Best-Fit but it always performed better than MIBNI. In Table 3.4 and Table 3.5, Best-Fit also outperformed ATEN and MIBNI. And ATEN achieved better performance than MIBNI.

However, although Best-Fit used the exhaustive search, we find that in Tables 3.4 and 3.5, for large Boolean networks, even the data is noise-free, Best-Fit cannot always obtain 100% Recall and 100% F-score and 0% FPR. This is because inferring Boolean functions of a large network from time series data is an under-determined problem, which means there are many Boolean functions that are equally good in terms of the prediction error. Best-Fit was able to find and offer multiple Boolean functions with the best prediction accuracy, but Best-Fit did not select one of them as the final solution. To address this problem, we have to randomly select one solution from the outputs of Best-Fit. It means we might miss the true Boolean function and select an equally good Boolean function. On the other hand, since MIBNI always inferred up to 8 input nodes, its values of FPR and F-score are always lower than Best-Fit and ATEN because many false positive interactions were inferred.

We present experimental results from ATEN, Best-Fit and MIBNI on the artificial networks with 50, 100, 150 nodes inferred from noisy data (with 1% and 5% noise) in Table

Table 3.4: Experimental results of ATEN, MIBNI and Best-Fit for the artificial networks with 100 nodes inferred from the noise-free data. The maximum in-degree of each target node inferred by ATEN, MIBNI and Best-Fit was limited to 8. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.

Number of Time Points	Algorithm	Noise 0%		
		Recall	FPR	F-score
50	ATEN	0.887±0.012	0.005±0.000	0.782±0.015
	MIBNI	0.734	0.072	0.203
	Best-Fit	0.914	0.001	0.914
100	ATEN	0.932±0.013	0.002 ±0.001	0.891±0.026
	MIBNI	0.766	0.071	0.211
	Best-Fit	0.969	0.000	0.947
150	ATEN	0.945±0.012	0.001±0.000	0.930±0.014
	MIBNI	0.789	0.071	0.218
	Best-Fit	0.992	0.000	0.992
200	ATEN	0.958±0.012	0.001±0.000	0.946±0.011
	MIBNI	0.789	0.071	0.218
	Best-Fit	0.984	0.001	0.969

3.6, Table 3.7 and Table 3.8, respectively. In addition, the results of inferring all artificial networks using known maximum in-degree are detailed in Supplementary Data.

From Tables 3.6, which shows the experimental results of ATEN, MIBNI and Best-Fit for inferring the network with 50 nodes from noisy data, we can see that ATEN always obtained the best Recall, FPR and F-score when the datasets are noisy (i.e., 1% and 5% noise). As the level of noise increases, all algorithms got worse Recall, FPR and F-score. Particularly, all algorithms got much poorer F-score when the data includes 5% noise. It may be because many false positive edges were inferred. From Tables 3.7 and 3.8, which

Table 3.5: Experimental results of ATEN, MIBNI and Best-Fit for the artificial networks with 150 nodes inferred from the noise-free data. The maximum in-degree of each target node inferred by ATEN, MIBNI and Best-Fit was limited to 8. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.

Number of Time Points	Algorithm	Noise 0%		
		Recall	FPR	F-score
50	ATEN	0.893±0.013	0.003 ±0.000	0.787 ±0.017
	MIBNI	0.769	0.047	0.206
	Best-Fit	0.930	0.001	0.930
100	ATEN	0.947±0.009	0.002 ±0.000	0.881±0.010
	MIBNI	0.811	0.047	0.218
	Best-Fit	1	0	1
150	ATEN	0.962±0.009	0.001±0.000	0.915±0.015
	MIBNI	0.806	0.047	0.216
	Best-Fit	0.995	0.000	0.995
200	ATEN	0.965 ±0.006	0.001±0.000	0.913±0.010
	MIBNI	0.806	0.047	0.216
	Best-Fit	0.989	0.000	0.989

include the experimental results of all algorithms for inferring the network including 100 nodes and 150 nodes, we can find the same results that ATEN always obtained the best Recall, FPR and F-score when the datasets are noisy (i.e., 1% and 5% noise). On the other hand, we can find that since Best-Fit uses exhaustive search, it cannot successfully infer the networks from the datasets including 200 time points with 1% and 5% noise within the walltime. Additionally, as the number of time points increases, we do not observe the overall performance turns to be better when the time point exceeds 100.

In conclusion, when the data is noisy, we can see that the overall performance of

Table 3.6: ATEN inferred more accurate artificial network topology from short and noisy data. Experimental results of ATEN, MIBNI and Best-Fit for the networks with 50 nodes inferred from noisy data. The maximum in-degree of each target node inferred by ATEN, MIBNI and Best-Fit was limited to 8. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.

Number of Time Points	Algorithm	Noise 1%			Noise 5%		
		Recall	FPR	F-score	Recall	FPR	F-score
50	ATEN	0.963 ±0.017	0.009±0.001	0.834±0.023	0.871±0.029	0.049±0.005	0.469±0.026
	MIBNI	0.750	0.144	0.207	0.750	0.144	0.207
	Best-Fit	0.953	0.015	0.753	0.797	0.063	0.381
100	ATEN	0.972±0.020	0.021±0.004	0.707±0.039	0.945±0.049	0.049±0.005	0.498±0.026
	MIBNI	0.797	0.143	0.220	0.197	0.143	0.220
	Best-Fit	0.922	0.058	0.445	0.750	0.104	0.262
150	ATEN	0.977 ±0.022	0.030±0.003	0.628±0.027	0.938±0.021	0.064 ±0.002	0.428±0.010
	MIBNI	0.797	0.143	0.220	0.813	0.143	0.224
	Best-Fit	0.953	0.077	0.391	0.719	0.132	0.213
200	ATEN	0.959±0.016	0.033±0.004	0.597±0.027	0.957±0.016	0.056±0.003	0.468±0.014
	MIBNI	0.766	0.144	0.211	0.781	0.144	0.216
	Best-Fit	0.906	0.099	0.320	0.781	0.142	0.218

Table 3.7: ATEN inferred more accurate artificial network topology from short and noisy data. Experimental results of ATEN, MIBNI and Best-Fit for the networks with 100 nodes inferred from the datasets including 1% noise and 5% noise. The maximum in-degree of each target node inferred by ATEN, MIBNI and Best-Fit was limited to 8. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface. The symbol ‘-’ indicates that the results of Best-Fit cannot be obtained due to the walltime limit (8 days).

Number of Time Points	Algorithm	Noise 1%			Noise 5%		
		Recall	FPR	F-score	Recall	FPR	F-score
50	ATEN	0.898±0.010	0.009±0.001	0.702±0.016	0.827±0.018	0.022±0.001	0.468±0.012
	MIBNI	0.758	0.071	0.209	0.758	0.071	0.209
	Best-Fit	0.797	0.018	0.499	0.664	0.031	0.327
100	ATEN	0.950±0.010	0.007±0.001	0.765±0.020	0.922±0.017	0.016±0.002	0.579±0.019
	MIBNI	0.773	0.071	0.213	0.766	0.071	0.211
	Best-Fit	0.906	0.027	0.458	0.781	0.046	0.293
150	ATEN	0.953±0.010	0.008±0.001	0.755±0.018	0.913±0.013	0.011±0.002	0.667±0.016
	MIBNI	0.758	0.071	0.209	0.766	0.071	0.211
	Best-Fit	0.930	0.029	0.447	-	-	-
200	ATEN	0.901±0.009	0.007±0.001	0.735±0.019	0.881±0.015	0.013±0.001	0.615±0.010
	MIBNI	0.766	0.071	0.211	0.773	0.071	0.213
	Best-Fit	-	-	-	-	-	-

ATEN is better than the performance of MIBNI and Best-Fit. In contrast, the performance of Best-Fit was the best for noise-free data but it deteriorated when applied to noisy data. This is because as an exhaustive search algorithm, Best-Fit might generate Boolean functions that overfit the noise but does not capture the true Boolean relationship.

We also ran ATEN with the upper limit on tree size equal to the known maximum in-degree of the networks, i.e., 5. We summarize the experimental results of all algorithms (i.e., ATEN, MIBNI and Best-Fit) for inferring networks with 50, 100, 150 nodes from both noise-free and noisy data in Supplementary Data. It is obvious to see that when the data is noisy, ATEN still achieved the best overall performance than the other two algorithms.

Table 3.8: ATEN inferred more accurate artificial network topology from short and noisy data. Experimental results of ATEN, MIBNI and Best-Fit for the networks with 150 nodes inferred from noisy data. The maximum in-degree of each target node inferred by ATEN, MIBNI and Best-Fit was limited to 8. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface. The symbol ‘-’ indicates that the results of Best-Fit cannot be obtained due to the walltime limit.

Number of Time Points	Algorithm	Noise 1%			Noise 5%		
		Recall	FPR	F-score	Recall	FPR	F-score
50	ATEN	0.875±0.011	0.008±0.000	0.622±0.013	0.873±0.017	0.016±0.000	0.445±0.015
	MIBNI	0.769	0.047	0.209	0.769	0.047	0.206
	Best-Fit	0.828	0.008	0.587	0.640	0.020	0.317
100	ATEN	0.951±0.009	0.009±0.000	0.609±0.011	0.898±0.013	0.020±0.001	0.423±0.013
	MIBNI	0.780	0.047	0.209	0.769	0.047	0.206
	Best-Fit	0.881	0.016	0.459	-	-	-
150	ATEN	0.955±0.005	0.011±0.001	0.573±0.009	0.925±0.014	0.024±0.002	0.385±0.012
	MIBNI	0.806	0.047	0.216	0.812	0.047	0.218
	Best-Fit	-	-	-	-	-	-
200	ATEN	0.958±0.009	0.013±0.001	0.546±0.011	0.924±0.014	0.025±0.001	0.379±0.009
	MIBNI	0.811	0.047	0.218	0.801	0.047	0.215
	Best-Fit	-	-	-	-	-	-

3.3.2.2 Performance on real-world network

We used the real-world gene regulatory network (i.e., *Drosophila* segment polarity gene regulatory network (Albert and Othmer, 2003)) to evaluate ATEN’s performance on inferring network topology and its dynamic model (i.e., Boolean functions), in comparison with other three algorithms (i.e., MIBNI, Best-Fit and REACT).

Inference of network topology We first present the experimental results of ATEN, MIBNI, Best-Fit and REACT for the real-world network inferred from the noise-free data in Table 3.9, where the maximum in-degree of each target node was limited to 8 (i.e., all algorithms inferred up to 8 nodes as input nodes except for REACT due to unavailable implementation). Table 3.9 shows the similar results where we obtained on artificial net-

Table 3.9: Experimental results of ATEN, MIBNI and Best-Fit for the real-world networks inferred from the noise-free data. The maximum in-degree of each target node inferred by ATEN, MIBNI and Best-Fit was limited to 8. The results of ATEN are presented using average and standard deviation values. The experimental results of REACT, i.e., the best Recall and FPR values which are presented outside brackets and the worst Recall and FPR values which are showed in brackets, were collected from Vera-Licona et al. (2014). The best result among three algorithms is highlighted in boldface. The symbol “*” indicates that the corresponding results cannot be collected.

Number of Time Points	Algorithm	Noise 0%		
		Recall	FPR	F-score
50	ATEN	0.591 $\pm$ 0.017	0.022 $\pm$ 0.000	0.553 $\pm$ 0.018
	MIBNI	0.465	0.120	0.214
	Best-Fit	0.639	0.009	0.689
	REACT	*	*	*
100	ATEN	0.776 $\pm$ 0.015	0.011 $\pm$ 0.000	0.760 $\pm$ 0.017
	MIBNI	0.486	0.119	0.224
	Best-Fit	0.840	0.005	0.855
	REACT	*	*	*
150	ATEN	0.838 $\pm$ 0.009	0.007 $\pm$ 0.000	0.830 $\pm$ 0.013
	MIBNI	0.501	0.118	0.233
	Best-Fit	0.924	0.003	0.920
	REACT	*	*	*
202	ATEN	0.910 $\pm$ 0.006	0.009 $\pm$ 0.000	0.860 $\pm$ 0.012
	MIBNI	0.486	0.119	0.224
	Best-Fit	0.940	0.001	0.950
	REACT	0.778(0.61)	0.082(0.2)	*

works, where Best-Fit always achieved the best Recall and F-score values for all datasets. And ATEN always performed better than MIBNI and REACT. Different from the results

obtained from inferring artificial network of 50 nodes, even for Best-Fit, it cannot infer 100% true positive edges and obtained 0.639 in terms of Recall for inferring the real-world network due to its more complex network structure.

On the other hand, ATEN outperformed the other three algorithms for inferring the network topology from short and noisy data. We present their performance among the datasets including 1% and 5% noise with setting the maximum in-degree of target nodes to 8 in Figure 3.4. In Figure 3.4, the average and standard deviation values of Recall and FPR obtained from ATEN were used for comparison. From Figure 3.4, we can see that among all these four algorithms ATEN always obtained the best FPR and F-score values. Compared with the performance of MIBNI, it can be seen that ATEN was significantly better. With respect to the network inferred from the dataset including 50 time points and 1% noise, ATEN achieved similar experimental results with Best-Fit. However, when the datasets including 5% noise, we can find that ATEN achieved significantly better recall, FPR and F-score values than Best-Fit. Compared with the best Recall and FPR values of REACT, we can see that ATEN achieved similar but more stable results. Moreover, from Figure 3.4, it can be concluded that ATEN outperformed the other three algorithms when the data was shorter (i.e., 50 time points) with higher noise level (i.e., 5% noise).

In addition, we also ran ATEN with the upper limit on tree size set to 4, i.e., the maximum in-degree of the real-world network. In general, from Supplementary Data, when the data is noise-free, we can see that the overall performance of Best-Fit was better than other algorithms. When the data includes noise, ATEN outperformed other algorithms. More detailed results can be found at Supplementary Data.

Inference of dynamic model The overall performance of ATEN for predicting the dynamic model of the real-world network was better than MIBNI, Best-Fit and similar to REACT. Table 3.10 presents the number of steady states that can be retrieved by ATEN and other algorithms when the maximum in-degree of the inferred network is unknown.

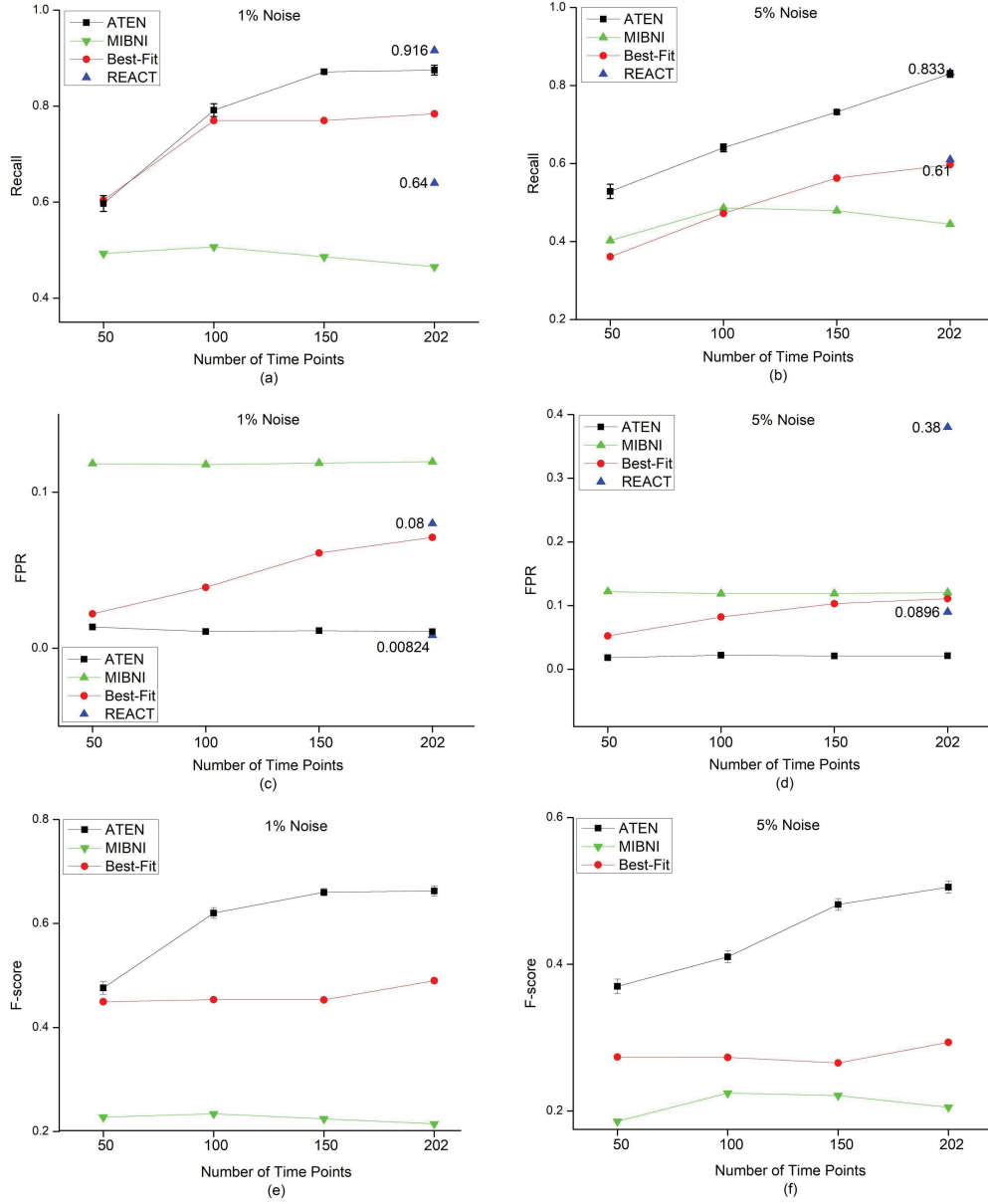


Figure 3.4: ATEN inferred more accurate real-world network topology from short and noisy data. Experimental results of ATEN, MIBNI, Best-Fit and REACT for the inferred network from the datasets including 1% noise and 5% noise. The maximum in-degree of each target node inferred by ATEN, MIBNI and Best-Fit was limited to 8. The results of ATEN are presented using average and standard deviation values. The experimental results of REACT, i.e., the maximum and minimum Recall and FPR values were collected from the literature (i.e., Vera-Licona et al. (2014)), and the F-score values cannot be collected.

Table 3.10: ATEN inferred more accurate real-world dynamic model from short and noisy data. The maximum in-degree of each target node inferred by ATEN, MIBNI and Best-Fit was limited to 8. The numbers indicate the number of steady states that was successfully retrieved. The best result among three algorithms is highlighted in boldface. Note that the experimental results of REACT was collected from Vera-Licona et al. (2014) and the symbol ‘*’ indicates that the corresponding results cannot be collected.

Number of Time Points	50			100			150			202		
Algorithms	Noise 0%	1%	5%	Noise 0%	1%	5%	Noise 0%	1%	5%	Noise 0%	1%	5%
ATEN	5	5	3	4	5	5	4	4	6	4	4	5
MIBNI	0	0	0	0	0	0	0	0	0	0	0	0
Best-Fit	3	2	2	4	1	1	5	2	2	5	2	2
REACT	*	*	*	*	*	*	*	*	*	5	3	5

From Table 3.10, we can observe that ATEN always retrieved at least 50% steady states even when the input data was short and noisy (i.e., 50 time points and 5% noise). MIBNI did not find any steady states, that is because MIBNI can infer only conjunctive or disjunctive Boolean functions. Compared with Best-Fit and MIBNI, we can see that ATEN was significantly better than both the compared algorithms among all datasets. Compared with REACT, ATEN correctly retrieved the same number of steady states. The experimental results also show that ATEN was more robust to different levels of noise and different numbers of time points for inferring the network dynamic model.

Furthermore, when the tree size is limited to 4, the overall performance of ATEN for predicting the dynamic model was better than MIBNI, Best-Fit and similar to REACT. Related description can be found in the Supplementary Data.

3.4 Summary

Inferring topological structure and network dynamics of gene regulatory networks from gene expression time series data is a major challenge in systems biology. Additionally, given the truth that gene expression data is always collected from small number of samples with noise, it may lead to unreliable network inference.

The fundamental problem of network inference is to identify the regulatory rules with input nodes for each gene in the network. To address this problem, we introduce the feature selection method and then use the optimization method to infer the final Boolean function. Specifically, we recursively reconstruct and select prime implicants as features to represent the regulate rule between the target node and a set of input nodes. Then we use the simulated annealing algorithm to build a Boolean function based on the selected features.

We apply the Simulated Annealing algorithm because of its simplicity and good search performance. However, any other heuristic algorithms such as Genetic Algorithms could be suitable for this inference task. It would be interesting to investigate whether other heuristic algorithms could improve the performance. Another reason that we used the heuristic algorithm is because it is faster than exhaustive search method (e.g. Best-Fit).

Both artificial and real-world case studies are presented and the results were compared to the ones inferred from similar network inference algorithms. Our experimental results have shown that, when the data was short and noisy, ATEN performed better than the other three algorithms in terms of the prediction accuracy. When the data was noise-free, Best-Fit achieved the best results. We therefore recommend using ATEN for inferring Boolean networks for noisy data, while Best-Fit for noise-free data.

Chapter Four

Inferring Boolean networks using a genetic algorithm with marker-based encoding

In this chapter, we proposed another algorithm for inferring both network topology and its dynamics using a genetic algorithm with a marker-based encoding scheme. Section 4.1 explains the background of the genetic algorithm used in network inference and introduces the motivation of our proposed algorithm. Section 4.2 describes the mark-based encoding scheme and the details of our algorithm. Section 4.3 presents the experimental results. And Section 4.4 summarizes this chapter.

4.1 Introduction

As we introduced in Section 2.1, the genetic algorithm is a population-based search and optimization algorithm which has been successfully applied in gene regulatory network. However, despite their advances in network inference, there are still shortcomings in existing algorithms.

First, several genetic algorithms used binary encoding to encode the Boolean network. Mendoza and Bazzan (2011) marked each gene from 1 to N and then encoded all genes as binary values. They also presented an example where the standard output values of a Boolean function was also encoded into the chromosome. We can see it would lead to a long binary string for inferring the target nodes with a large number of input nodes. The similar problem also exists in Barman and Kwon (2018). It uses the binary encoding, in which each bit is only ‘0’ or ‘1’. The position with ‘1’ indicates the gene corresponding to that position is selected as an input gene, otherwise ‘0’ means not. Thus the length of each chromosome is n where n is the number of total genes. On the other hand, the crossover and mutation operators may be ineffective and cannot ensure sufficient genetic diversity when the chromosome exists many ‘0’ or ‘1’ positions for a large-scale network (with the number of nodes > 16 (Kim et al., 2013)). Another issue for their algorithms is that the non-deterministic cases cannot be handled as they did not infer the specific regulatory rule for each target gene. Thus it is unclear how their inferred model is used for predicting the output value of an input combination if the observed data includes non-deterministic cases.

For the inference algorithms that reconstruct the Boolean function, e.g., (Barman and Kwon, 2017), the Boolean function is always described by three basic logic operators (i.e., $\vee$, $\wedge$, $\neg$). Then the genetic algorithm is inefficient to find the complex functions (e.g. XOR, XNOR). Additionally, most existing inference algorithms including our work (Shi et al., 2020) described in Chapter 3 infer each gene individually and then combine all genes as a final Boolean network. Inspired by (Mendoza and Bazzan, 2011; Vera-Licona et al., 2014), we propose to encode all nodes with their Boolean functions into one chromosome.

In this chapter, we design MGANI, which is a novel genetic algorithm with the marker-based encoding scheme to infer the Boolean network. The marker-based encoding scheme is initially proposed by Fullmer and Miikkulainen (1992) to evolve the topology of neural networks. Although the marker-based encoding scheme has been widely applied to evolving neural networks Hutt and Warwick (2007), no marker-based evolutionary algorithm exists to

infer Boolean networks. In MGANI, the marker-based encoding scheme represents a Boolean network as a chromosome. The chromosome consists of multiple sequences in which each represents a Boolean function of a target node. The marker bit is used to separate each sequence.

To summarize, the proposed algorithm has the following three main contributions:

1. We propose a novel Boolean network inference algorithm called MGANI (Marker-based Genetic Algorithm Network Inference), which is capable of inferring both the network topology and its dynamics. To the best of our knowledge, we are the first to apply the marker-based encoding scheme to Boolean network inference.
2. By adopting the markers to distinguish sequences with different lengths and introducing more logical operators in the encoding scheme, the proposed algorithm is able to efficiently infer more diverse Boolean functions.
3. The proposed algorithm is validated on both artificial Boolean networks and real-world Boolean networks. The overall results show that MGANI outperforms Best-Fit (Lähdesmäki et al., 2003), GABNI (Barman and Kwon, 2018), and ATEN (Shi et al., 2020).

4.2 Methods

In this section, we propose MGANI to infer Boolean networks using the genetic algorithm. The key features are that we encode all input nodes' Boolean functions into the chromosome based on the marker bit and introduce more logic operators into the encoding. For a Boolean network including n nodes, firstly, we initialize the population by randomly generating chromosomes, where each chromosome includes n sequences. Each sequence encodes the Boolean function associated with input nodes for a target node. Then, we adopt the tour-

nament selection scheme to choose better chromosomes as parents. After that, we conduct the crossover operator as well as mutation operator on these chromosomes. In addition, we utilize the heuristic sequence assembling scheme to find and assemble the best sequences as a new chromosome for the purpose of accelerating the convergence. The maximum number of iterations is predefined to stop the searching scheme. We demonstrate the details and features of our proposed algorithm in the following subsections.

4.2.1 Marker-based encoding scheme

Biological significance In DNA, the biological information is embodied in a sequence of nucleotide triplets. These triplets specify strings of amino-acids that make up a protein. Typically, a single strand of DNA specifies multiple proteins in this way. Certain nucleotide triplets serve as markers rather than being part of amino-acid definitions, which distinguishes the specification of different proteins. Each protein specification consists of a start marker and an end marker with the triplets in between defining the composition of the protein (Fullmer and Miikkulainen, 1992). This inspires us to adopt the marker-based encoding scheme in which each target node with its Boolean function and input nodes encoded in a chromosome is distinguished by the marker bit.

Chromosome representation with diverse logical operators A chromosome is composed of n sequences, representing a Boolean network consisting of n nodes with their Boolean functions. The sequences are separated by the marker bit *Start*. A sequence S_i denotes the Boolean function of a target node x_i . The Boolean function corresponds to four components in the sequence. The first component is the marker bit *Start*. The second component encodes the target nodes x_i . The third component corresponds to the input nodes of x_i . The last component includes logical operators. The length of each sequence in a chromosome is determined by the number of inferred input nodes. And the maximum length of each sequence is determined by x_{i_K} where K denotes the maximum number of input genes of x_i .

All bits in the sequence are encoded using integers. Since there are different components in a chromosome, their values have different ranges in order to be easily distinguished in the chromosome. In our encoding scheme, the nodes from x_1 to x_n are encoded from 1 to n . The logical operators range are encoded using the integers between $5n$ to $5n + 4$. Particularly, the negation of an input node x_i is denoted by the opposite of i (i.e., $-i$).

We present a chromosome example consisting of three sequences in Figure 4.1 and its encoding details in Table 4.1. The first sequence begins with $Start = 10$, then $(1, 2, -3, 1)$ denotes the node x_1 is regulated by three nodes (i.e., x_2, x_3 and x_1). $(16, 15)$ represents that two predefined logical operators are included in the Boolean function. Thus the first sequence presents the Boolean function $x_1(t+1) = x_2(t) \vee \neg x_3(t) \wedge x_1(t)$. The second sequence presents the Boolean function $x_2(t+1) = \neg x_1(t)$ and the third sequence denotes the Boolean function $x_3(t+1) = x_1(t) \oplus x_2(t)$. The lengths of the three sequences (i.e., the number of bits) are 7, 3, and 5, respectively. And the length of each sequence can be different with each other since the number of inferred input nodes is different.

Notice that instead of only containing three basic logical operators (i.e., *AND*, *OR*, *NOT*) in the Boolean function, we introduce *XOR*, *NAND*, *NOR* in the marker-based encoding scheme for efficiently representing complex Boolean functions. For example, $x_3(t+1) = (x_1(t) \wedge \neg x_2(t)) \vee (\neg x_1(t) \wedge x_2(t))$ can be represented by $x_3(t+1) = x_1(t) \oplus x_2(t)$.

4.2.2 Algorithm

Population Initialization We summarize the population initialization in Algorithm 4.1. Here, PS is the population size (i.e., the number of chromosomes), n is the number of nodes in the Boolean network, and K is the maximum number of input nodes of each target node in the Boolean network.

A Boolean network with n nodes is encoded as a chromosome consisting of n sequences

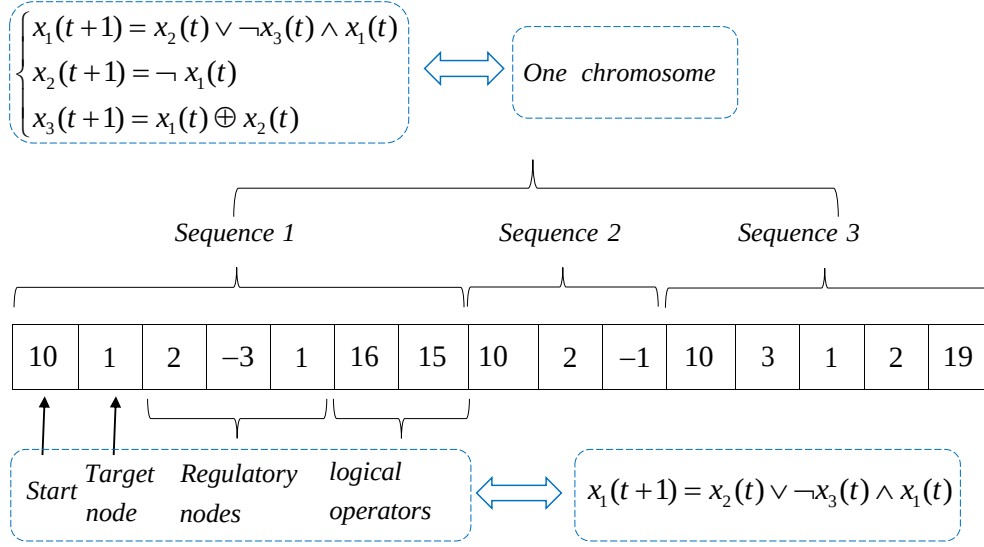


Figure 4.1: An example of a chromosome with specified values. According to specific definitions in Table 1, the first *Start* (i.e., 10) marks the beginning of sequence 1. Next to *Start* is 1 and it denotes the target node x_1 whose state is determined by the input nodes x_2 , x_3 , and x_1 with the logical operators *OR* and *AND*. Similarly, the sequence 2 and sequence 3 correspond to the second and the third Boolean functions, respectively.

Table 4.1: The encoding details for the components in the chromosome.

Symbol	Meaning	Value in the chromosome
<i>Start</i>	the start marker	$3n + 1$
x_i	the i th node	i
$\neg x_i$	<i>NOT</i> x_i	$-i$
$\wedge$	<i>AND</i>	$5n$
$\vee$	<i>OR</i>	$5n + 1$
$\bar{\vee}$	<i>NOR</i>	$5n + 2$
$\bar{\wedge}$	<i>NAND</i>	$5n + 3$
$\oplus$	<i>XOR</i>	$5n + 4$

(i.e., n Boolean functions). The first step in Algorithm 4.1 is to randomly assign a $k_{max} \in \{1, 2, \dots, K\}$ to $i_{k_{max}}$ to limit the length of each sequence. The second step is to generate a sequence of integers according to Table 4.1. After generating all the n sequences, then we combine the n sequences to form a chromosome CH_r (i.e., a Boolean network). We repeat this process until the number of chromosomes reaches the population size PS .

Algorithm 4.1: Population Initialization

Input : The maximum number of input nodes K , the number of nodes n in the Boolean network, the population size PS

Output: The population PN

```

1 for  $r \leftarrow 1$  to  $PS$  do
2   for  $i \leftarrow 1$  to  $n$  do
3      $k_{max} \leftarrow$  Randomly generate an integer ranging from 1 to  $K$ ;
4      $SE_i \leftarrow$  Randomly generate the  $i$ th sequence with length  $(2k_{max} + 1)$ ;
5    $CH_r \leftarrow$  Combine all  $SE_i$  as a chromosome;

```

Fitness Evaluation As we include all targets nodes in the chromosome, we first propose the inconsistency I_i to measure the quality of each sequence SE_i for a time series:

$$I_i = \sum_{t=2}^N (y_i(t) - x_i(t))^2 \quad (4.1)$$

where N is the number of time points in the time series data, $y_i(t)$ is the output state of x_i predicted by the Boolean function encoded in the sequence SE_i , $x_i(t)$ denotes the true state of the node x_i at time point t . Thus, given a dataset D with J time series, the fitness score of a chromosome can be evaluated by computing

$$fitness = \sum_{j=1}^J \sum_{i=1}^n I_i^j \quad (4.2)$$

n refers the number of nodes in the Boolean network.

Selection The selection aims to choose the parent chromosomes with good fitness values from the population for generating new better chromosomes. As one of the popular selection method, the binary tournament selection strategy is adopted to select chromosomes from the current generation as parent chromosomes. Each time, two chromosomes are selected from the current population and the fittest chromosome among the two chromosomes is chosen and passed to the next generation. We repeat this process until the population size is reached.

Crossover We summarize the crossover subroutine in Algorithm 4.2 with the purpose of generating new chromosomes. Firstly, we find all the sequences based on the *Start* markers and define $Pr_{crossover}$ as the probability of conducting the crossover operator. For each pair of chromosomes, we generate a random number u uniformly distributed in $[0,1]$. If $u \leq Pr_{crossover}$, then we exchange the random selected i th sequence in CH_r with the i th sequence in CH_{r+1} to obtain two new chromosomes. We repeat this process until a new population of PN with the population size PS is achieved.

Algorithm 4.2: Crossover

Input : $PN, n, Pr_{crossover}$

Output: The new population PN

```

1 for  $r \leftarrow 1$  to  $PS$  step 2 do
2   select two chromosomes  $CH_r, CH_{r+1}$  from  $PN$ ;
3    $u \leftarrow U(0, 1)$ ;
4   if  $(u \leq Pr_{crossover})$  then
5      $i \leftarrow$  Randomly generate an integer ranging from 1 to  $n$ ;
6     Exchange the  $i$ th sequence in  $CH_r$  with the  $i$ th sequence in  $CH_{r+1}$ ;

```

Mutation We summarize the mutation operator in Algorithm 4.3 for generating new chromosomes that may not present in the population. We define $Pr_{lmutation}$ as the probability of mutating logical operators and $Pr_{imutation}$ as the probability of mutating the input nodes.

Specifically, the first step is to find all the sequences in the r th chromosome CH_r . Then, we identify every logical operator o in the sequence SE . After that, we generate a random number u_1 uniformly distributed in $[0,1]$. If $u_1 \leq Pr_{lmutation}$ then we change the logical operator o to another operator. The way for mutating the input nodes are similar to the way for mutating the logical operators. For each input node v in a sequence in a chromosome, we generate a random number u_2 uniformly distributed in $[0,1]$. If $u_2 \leq Pr_{imutation}$, then we assign v with a new value. An example of the encoding details of the operators are predefined in Table 4.1.

Algorithm 4.3: Mutation

Input : $PN, Pr_{lmutation}, Pr_{imutation}$

Output: The mutated population PN

```

1 foreach chromosome  $CH_r$  in  $PN$  do
2   foreach sequence  $SE$  in  $CH_r$  do
3     foreach operator  $o$  in  $SE$  do
4        $u_1 \leftarrow U(0, 1)$  ;
5       if ( $u_1 \leq Pr_{lmutation}$ ) then
6          $o \leftarrow$  randomly generate an integer from the set
           $\{5n, 5n + 1, \dots, 5n + 4\} \setminus \{o\}$  ;
7     foreach input node  $v$  in  $SE$  do
8        $u_2 \leftarrow U(0, 1)$  ;
9       if ( $u_2 \leq Pr_{imutation}$ ) then
10         $v \leftarrow$  randomly generate an integer from the set
           $\{-n, -n + 1, \dots, n - 1, n\} \setminus \{v\}$  ;
11 return the mutated chromosome  $CH_r$ 

```

Heuristic sequence assembling After mutation, we utilize a heuristic sequence assembling for accelerating the convergence. The fitness of the sequence SE_i is calculated as the average

inconsistency (see Equation 4.1) over all time series: $fitSE_i = \frac{1}{J \times N} \sum_{j=1}^J I_i^j$ where J is the number of time series in a dataset and N is the length (i.e., the number of time points) of a time series. By utilizing $fitSE_i$ to evaluate the i th sequence, we find all the best $SE_i, i \in \{1, 2, \dots, n\}$ across all chromosomes, then we assemble these sequences as a new chromosome. This new chromosome would replace the chromosome whose fitness is the worst in the population.

4.3 Computational experiments

4.3.1 Datasets

4.3.1.1 Artificial networks

To validate the performance of MGANI, we introduce two artificial networks generated by BoolNet (Müssel et al., 2010), where there are 20 nodes and 30 nodes, respectively. The maximum in-degree of each network is set to 3. For each network, we generate a dataset including 20 time series and each time series contains 6 time points. Then we introduce the noise into the time series data by randomly flipping the state of each node with the probability of 1% or 5%.

4.3.1.2 Real-world network

We also applied MGANI to infer three small-scale with the number of nodes ≤ 16 (Kim et al., 2013)) real-world gene regulatory networks. The networks include the idealized protein-DNA interaction network, the cAMP receptor of *D. discoideum* network (Gao, Li, et al., 2013) and the *Drosophila* segment polarity gene regulatory network (Albert and Othmer, 2003). The first two networks include 6 nodes and 8 nodes, respectively. As shown in Table 4.2 and

Table 4.2: The Boolean functions of the idealized protein-DNA interaction network.

Target Node	Boolean functions
x_1	x_3
x_2	x_1
x_3	$\neg((x_2 \wedge x_5) \oplus x_2 \oplus x_5)$
x_4	$\neg((x_2 \wedge x_5) \oplus x_2 \oplus x_5)$
x_5	x_6
x_6	x_4

Table 4.3: The Boolean functions of the cAMP receptor of *D. discoideum* network.

Target Node	Boolean functions
x_1	$\neg x_3$
x_2	$\neg x_1$
x_3	$(x_2 \wedge x_8) \oplus x_8$
x_4	x_3
x_5	x_4
x_6	$\neg x_5$
x_7	$(x_4 \wedge x_5) \oplus x_4 \oplus x_5$
x_8	$(x_6 \wedge x_7) \oplus x_7$

Table 4.3, we present their Boolean functions. And in Figure 4.2 and Figure 4.3, we show their topological structure. For each network, we generate a dataset including 20 time series and each time series contains 6 time points. For the third network, we have described it in Section 3.3.1.2 and present its topology and Boolean functions in Figure 3.3 and Table 3.1. We used a dataset including 10 time series in which each covers 20 time points. The 1% and 5% noises were also introduced into this dataset.

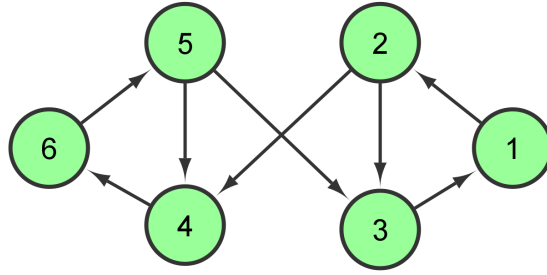


Figure 4.2: The topology of the idealized protein-DNA interaction network. The directed edges represent the existence of regulation without implying the activation or inhibition.

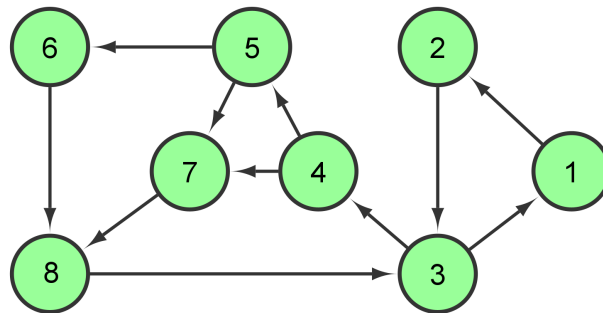


Figure 4.3: The topology of the cAMP receptor of *D.discoideum* network. The directed edges represent the existence of regulation without implying the activation or inhibition.

Table 4.4: The parameters in the proposed algorithm.

Symbol	Meaning	Value
$Pr_{crossover}$	the crossover probability	0.5
$Pr_{lmutation}$	the mutation probability of logical operators	0.1
$Pr_{imutation}$	the mutation probability of input nodes	0.1
PS	the population size	300
$Iter$	the number of iterations	200

4.3.2 Experimental design

We compared our algorithm MGANI with three existing network inference algorithms i.e., Best-Fit (Lähdesmäki et al., 2003), GABNI (Barman and Kwon, 2018) and ATEN (Shi et al., 2020) which can both infer the network topology and its dynamics. Table 4.1 illustrates the encoding details we used in MGANI. Table 4.4 presents the parameters used in our experiments. In the experiment, all algorithms were set to infer up to 3 input nodes for each target node (i.e., $K = 3$) when inferring two artificial networks and two smaller real-world networks. We limited $K = 4$ for inferring the *Drosophila* segment polarity gene network. Since that ATEN, GABNI, and MGANI are heuristic algorithms, we executed them 30 runs on every experiment and present their average values as well as standard deviation values in the following sections.

To evaluate the quality of the inferred network topology, we used the performance metrics Precision, False Positive Rate (FPR), Accuracy and F-score described in Section 2.1.3. To measure the quality of the dynamic model of the inferred network, we used the second method (i.e., Equation 2.8) defined in Section 2.1.3. Since the network states in an attractor can be seen as a time series, then the dynamical changes of the whole network over time starting from the first network state in the attractor are used.

Table 4.5: Experimental results of Best-Fit, ATEN, GABNI and MGANI for inferring the network topology which includes 20 nodes. The results of ATEN, GABNI and MGANI are presented using average and standard deviation values. The best result among four algorithms is highlighted in boldface.

		Noise		
		0%	1%	5%
Accuracy	Best-Fit	0.978	0.968	0.943
	ATEN	0.977±0.009	0.970±0.008	0.963±0.008
	GABNI	0.974±0.011	0.967±0.009	0.943±0.011
	MGANI	0.983±0.013	0.975±0.014	0.965±0.008
Precision	Best-Fit	0.833	0.791	0.559
	ATEN	0.829±0.034	0.772±0.045	0.692±0.042
	GABNI	0.790±0.041	0.745±0.031	0.534±0.033
	MGANI	0.868±0.029	0.823±0.032	0.795±0.026
FPR	Best-Fit	0.010	0.016	0.037
	ATEN	0.011±0.007	0.013±0.005	0.021±0.005
	GABNI	0.013±0.005	0.017±0.004	0.040±0.007
	MGANI	0.012±0.009	0.015±0.006	0.020±0.008
F-score	Best-Fit	0.816	0.760	0.654
	ATEN	0.811±0.019	0.779±0.022	0.706±0.019
	GABNI	0.781±0.028	0.735±0.021	0.611±0.017
	MGANI	0.857±0.019	0.821±0.014	0.795±0.022

4.3.3 Results

4.3.3.1 Performance on artificial networks

Inference of network topology We first used all algorithms to infer the network including 20 nodes. The corresponding experimental results are demonstrated in Table 4.5. Overall,

we can find that MGANI outperformed three other algorithms with respect to Precision, F-score, and Accuracy. As the level of noise increase, we can see that the performance of the four algorithms decreases. Although Best-Fit uses the exhaustive search, we can find that it cannot get the best result even when the data is noise-free. That is because there are more than one Boolean functions fit the data equally, and we choose a random one as the final solution.

In Table 4.6, we present the experimental results of Best-Fit, ATEN, GABNI and MGANI for inferring the network topology which includes 30 nodes. We can find that MGANI obtained the best results when the data is noisy. When the data is noisy, MGANI gained the best Precision and F-score. In terms of Accuracy and FPR, Best-Fit outperforms MGANI, ATEN and MGANI. And MGANI was still better than ATEN and GABNI. Overall, we can conclude that MGANI shows better performance on inferring the topology of the artificial Boolean network than the other three algorithms.

Inference of dynamic model Table 4.7 shows the dynamics accuracy of the inferred network with 20 nodes. When the data is noise-free, we can see all MGANI and ATEN obtained the best result where all network states in the attractor were retrieved. When data includes 1% noise, both MGANI and ATEN got 100% dynamic accuracy and Best-Fit completely inferred three attractors. When the data includes 5% noise, MGANI achieved the highest dynamic accuracy for four attractors. For the fourth attractor, MGANI performed slightly worse than ATEN but was still better than Best-Fit.

As shown in Table 4.8, there exist 2 attractors in the state space of the network with 30 nodes. When the data is noise-free or includes 1% noise, Best-Fit, ATEN, and MGANI can infer all network states in two attractors correctly. When the ratio of noise increases to 5%, the dynamics accuracy obtained from MGANI is higher than the other two algorithms.

Table 4.6: Experimental results of Best-Fit, ATEN, GABNI and MGANI for inferring the network topology which includes 30 nodes. The results of ATEN, GABNI and MGANI are presented using average and standard deviation values. The best result among four algorithms is highlighted in boldface.

		Noise		
		0%	1%	5%
Accuracy	Best-Fit	0.988	0.958	0.923
	ATEN	0.970±0.005	0.967±0.004	0.955±0.009
	GABNI	0.968±0.015	0.918±0.009	0.831±0.011
	MGANI	0.977±0.013	0.972±0.012	0.969±0.010
Precision	Best-Fit	0.735	0.463	0.313
	ATEN	0.738±0.039	0.711±0.045	0.687±0.036
	GABNI	0.596±0.052	0.406±0.073	0.247±0.010
	MGANI	0.837±0.043	0.789±0.038	0.727±0.037
FPR	Best-Fit	0.010	0.034	0.066
	ATEN	0.018±0.005	0.021±0.009	0.042±0.008
	GABNI	0.021±0.003	0.083±0.006	0.206±0.009
	MGANI	0.013±0.006	0.015±0.007	0.026±0.009
F-score	Best-Fit	0.733	0.568	0.444
	ATEN	0.761±0.047	0.739±0.051	0.723±0.059
	GABNI	0.622±0.054	0.478±0.049	0.358±0.017
	MGANI	0.832±0.053	0.797±0.051	0.756±0.045

4.3.3.2 Performance on real-world network

We included three real-world gene regulatory networks to evaluate MGANI's performance on the inference of both network topology and its dynamics, in comparison with the other three algorithms (i.e., Best-Fit, ATEN and GABNI). Notice that as GABNI cannot predict

Table 4.7: Experimental results of Best-Fit, ATEN and MGANI for inferring the dynamic model of the network with 20 nodes. The results of ATEN and MGANI are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.

		Dynamic Accuracy		
		0%	1%	5%
Attractor1	Best-Fit	1	1	0.900
	ATEN	1±0.000	1±0.000	0.933±0.022
	MGANI	1±0.000	1±0.000	0.954±0.019
Attractor2	Best-Fit	1	1	0.950
	ATEN	1±0.000	1±0.000	0.934±0.011
	MGANI	1±0.000	1±0.000	1±0.000
Attractor3	Best-Fit	1	1	0.925
	ATEN	1±0.000	1±0.000	0.916±0.036
	MGANI	1±0.000	1±0.000	0.931±0.041
Attractor4	Best-Fit	1	0.963	0.838
	ATEN	1±0.000	1±0.000	0.962±0.025
	MGANI	1±0.000	1±0.000	0.957±0.038
Attractor5	Best-Fit	1	0.950	0.775
	ATEN	1±0.000	1±0.000	0.933±0.022
	MGANI	1±0.000	1±0.000	0.857±0.031

the network states based on new input data that did not appear in the training data, we only present its results of inferring network topology.

Inference of network topology In Table 4.9, the results of inferring the idealized protein-DNA interaction network are summarized. We can see that MGANI always achieved the best Accuracy, Precision, FPR and F-score regardless of the level of noise.

Table 4.8: Experimental results of Best-Fit, ATEN and MGANI for inferring the dynamic model of the network with 30 nodes. The results of ATEN and MGANI are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.

		Dynamic Accuracy		
		0%	1%	5%
Attractor1	Best-Fit	1	1	0.973
	ATEN	1±0.000	1±0.000	0.977±0.025
	MGANI	1±0.000	1±0.000	0.982±0.029
Attractor2	Best-Fit	1	1	0.971
	ATEN	1±0.000	1±0.000	0.917±0.011
	MGANI	1±0.000	1±0.000	0.976±0.035

Table 4.10 demonstrates the performance on inferring the topology of the cAMP receptor of *D. discoideum* network. We can observe that the similar results where MGANI outperformed the other three algorithms in terms of all performance matrices. When the data includes 5% noise, MGANI more significantly outperforms other algorithms in terms of Precision and F-score.

Table 4.11 presents the experimental results of inferring the topological structure of the *Drosophila* segment polarity network. When the data is noise-free, Best-Fit gained better Accuracy, FPR and F-score and worse Precision than MGANI. ATEN also got better FPR than MGANI. When the level of noise increases, MGANI got the best Accuracy Precision and F-score for the dataset includes 1%. When the data contains 5% noise, MGANI achieved the best performance among all algorithms in terms of Accuracy, Precision, FPR and F-score. On the other hand, compared with the other two real-world networks, we can find the performance turned to be worse as the size of the network (i.e, the number of nodes) increase.

Table 4.9: Experimental results of Best-Fit, ATEN, GABNI and MGANI for inferring the the idealized protein-DNA network topology from the experimental data. The results of ATEN, GABNI and MGANI are presented using average and standard deviation values. The best result among four algorithms is highlighted in boldface.

		Noise		
		0%	1%	5%
Accuracy	Best-Fit	0.974	0.944	0.917
	ATEN	0.930±0.027	0.922±0.025	0.900±0.022
	GABNI	0.855±0.022	0.798±0.016	0.744±0.019
	MGANI	0.983±0.018	0.966±0.019	0.930±0.025
Precision	Best-Fit	0.887	0.875	0.727
	ATEN	0.876±0.027	0.851±0.031	0.782±0.033
	GABNI	0.631±0.017	0.522±0.046	0.464±0.016
	MGANI	0.903±0.032	0.867±0.038	0.809±0.052
FPR	Best-Fit	0.035	0.037	0.107
	ATEN	0.050±0.011	0.063±0.014	0.092±0.016
	GABNI	0.146±0.008	0.241±0.016	0.294±0.019
	MGANI	0.047±0.013	0.061±0.022	0.076±0.025
F-score	Best-Fit	0.875	0.870	0.842
	ATEN	0.873±0.027	0.852±0.024	0.832±0.033
	GABNI	0.722±0.036	0.650±0.032	0.605±0.019
	MGANI	0.969±0.027	0.939±0.029	0.882±0.032

Table 4.10: Experimental results of Best-Fit, ATEN, GABNI and MGANI for inferring the cAMP receptor of *D. discoideum* network topology from the experimental data. The results of ATEN, GABNI and MGANI are presented using average and standard deviation values. The best result among four algorithms is highlighted in boldface.

		Noise		
		0%	1%	5%
Accuracy	Best-Fit	0.922	0.859	0.813
	ATEN	0.930±0.023	0.888±0.021	0.826±0.025
	GABNI	0.926±0.021	0.842±0.028	0.710±0.027
	MGANI	0.955±0.026	0.940±0.034	0.913±0.029
Precision	Best-Fit	0.800	0.571	0.476
	ATEN	0.869±0.009	0.751±0.019	0.579±0.014
	GABNI	0.806±0.032	0.533±0.031	0.372±0.027
	MGANI	0.921±0.038	0.894±0.034	0.821±0.055
FPR	Best-Fit	0.038	0.113	0.208
	ATEN	0.038±0.008	0.086±0.010	0.127±0.016
	GABNI	0.057±0.012	0.124±0.021	0.344±0.045
	MGANI	0.032±0.013	0.041±0.022	0.094 ± 0.031
F-score	Best-Fit	0.762	0.640	0.625
	ATEN	0.846±0.011	0.772±0.024	0.635±0.017
	GABNI	0.776±0.039	0.613±0.032	0.537±0.028
	MGANI	0.904±0.034	0.880±0.043	0.826±0.037

Inference of dynamic model We included all two attractors of the protein-DNA interaction network for computing the dynamic accuracy where the first attractor contains two states and the second attractor includes six states. As shown in Table 4.12, all algorithms correctly predicted all states in the attractor and achieved 100% dynamic accuracy regardless of the noise in the time series data.

Table 4.11: Experimental results of Best-Fit, ATEN, GABNI and MGANI for inferring the *Drosophila* segment network topology from the experimental data. The results of ATEN, GABNI and MGANI are presented using average and standard deviation values. The best result among four algorithms is highlighted in boldface.

		Noise		
		0%	1%	5%
Accuracy	Best-Fit	0.987	0.951	0.928
	ATEN	0.979±0.005	0.968±0.004	0.953±0.002
	GABNI	0.954±0.004	0.946±0.007	0.937±0.007
	MGANI	0.982±0.002	0.973±0.003	0.959±0.005
Precision	Best-Fit	0.813	0.498	0.327
	ATEN	0.795±0.036	0.706±0.025	0.521±0.034
	GABNI	0.530±0.031	0.446±0.042	0.322±0.038
	MGANI	0.839±0.037	0.712±0.035	0.581±0.046
FPR	Best-Fit	0.006	0.045	0.059
	ATEN	0.011±0.003	0.012±0.002	0.023±0.003
	GABNI	0.022±0.005	0.029±0.004	0.052±0.004
	MGANI	0.014±0.002	0.015±0.002	0.021±0.003
F-score	Best-Fit	0.806	0.576	0.414
	ATEN	0.748±0.030	0.665±0.026	0.505±0.020
	GABNI	0.574±0.031	0.457±0.040	0.311±0.038
	MGANI	0.776±0.023	0.698±0.031	0.578±0.033

For the cAMP receptor of *D. discoideum* network, all three attractors were used for evaluating the quality of the inferred dynamic model. In Table 4.13, for the first attractor, MGANI achieved the same results with Best-Fit and ATEN where all algorithms got 100% dynamic accuracy under 0%, 1%, and 5% noise. For the second and third attractors, all algorithms correctly predict all network states when data is noise-free and includes 1%

Table 4.12: Experimental results of Best-Fit, ATEN and MGANI for inferring the dynamic model of the protein-DNA interaction network. The results of ATEN and MGANI are presented using average and standard deviation values.

	Algorithm	Dynamic Accuracy		
		Noise 0%	1%	5%
Attractor1	Best-Fit	1	1	1
	ATEN	1±0.000	1±0.000	1±0.000
	MGANI	1±0.000	1±0.000	1±0.000
Attractor2	Best-Fit	1	1	1
	ATEN	1±0.000	1±0.000	1±0.000
	MGANI	1±0.000	1±0.000	1±0.000

noise. When the data contains 5% noise, MGANI and Best-Fit still achieved 100% dynamic accuracy

We also used MGANI to infer the *Drosophila* segment polarity network. The six attractors that have been used here are described in Table 3.2 in Section 3.3.1.2. In Table 4.14, we present the experimental results of Best-Fit, ATEN and MGANI for inferring the dynamic model. We can find that Best-Fit, ATEN, and MGANI got 100% dynamic accuracy in six attractors correctly under 0% noise. When the data contains 1% noise, ATEN and MGANI got the best results when inferring the first five attractors. MGANI performed better than ATEN for inferring the sixth attractor. Compared with Best-Fit, the overall performance of MGANI was better than Best-Fit. When the data includes 5% noise, all algorithms fully retrieve all network states in the first attractor. And MGANI achieved the best performance on the inference of the other five attractors.

Table 4.13: Experimental results of Best-Fit, ATEN and MGANI for inferring the dynamic model of the cAMP receptor of D. discoideum network. The results of ATEN and MGANI are presented using average and standard deviation values.

	Algorithm	Dynamic Accuracy		
		Noise 0%	1%	5%
Attractor1	Best-Fit	1	1	1
	ATEN	1±0.000	1±0.000	1±0.000
	MGANI	1±0.000	1±0.000	1±0.000
Attractor2	Best-Fit	1	1	1
	ATEN	1±0.000	1±0.000	0.967±0.023
	MGANI	1±0.000	1±0.000	1±0.000
Attractor3	Best-Fit	1	1	1
	ATEN	1±0.000	1±0.000	0.929±0.033
	MGANI	1±0.000	1±0.000	1±0.000

4.4 Summary

This chapter describes MGANI, a genetic algorithm with mark-based encoding for answering the second research question, i.e., how to improve the scalability and accuracy of gene regulatory network inference.

Compared with the existing network inference algorithms using genetic algorithms, we introduce more logical operators (i.e., *XOR*, *XAND* and *NOR*) so that our algorithm is more efficient to infer more complex networks. And to avoid encoding each target node using a long string, we encode the marker bit, the target node, the input nodes and logical operators into the chromosome. The maximum number of bits required for encoding a Boolean function for a target node is determined by the maximum number of input nodes. Additionally, we also encode all nodes into one chromosome and apply a genetic algorithm for

Table 4.14: Experimental results of Best-Fit, ATEN and MGANI for inferring the dynamic model of the Drosophila segment polarity network. The results of ATEN and MGANI are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.

	Algorithm	Dynamic Accuracy		
		Noise 0%	1%	5%
Attractor1	Best-Fit	1	1	1
	ATEN	1±0.000	1±0.000	1.000±0.000
	MGANI	1±0.000	1±0.000	1.000±0.000
Attractor2	Best-Fit	1	1	0.967
	ATEN	1±0.000	1±0.000	0.984±0.023
	MGANI	1±0.000	1±0.000	1.000±0.000
Attractor3	Best-Fit	1	1	0.925
	ATEN	1±0.000	1±0.000	0.916±0.024
	MGANI	1±0.000	1±0.000	0.981±0.031
Attractor4	Best-Fit	1	0.963	0.907
	ATEN	1±0.000	1±0.000	0.931±0.019
	MGANI	1±0.000	1±0.000	0.965±0.022
Attractor5	Best-Fit	1	0.944	0.873
	ATEN	1±0.000	1±0.000	0.927±0.031
	MGANI	1±0.000	1±0.000	0.962±0.029
Attractor6	Best-Fit	1	0.939	0.775
	ATEN	1±0.000	0.945±0.018	0.901±0.021
	MGANI	1±0.000	0.961±0.031	0.923±0.016

finding the best Boolean network directly. We apply MGANI to infer both artificial networks and real-world networks, the experimental results demonstrate that the overall performance of MGANI is better than other algorithms when data is noisy.

Chapter Five

Finding the control kernel for controlling gene regulatory networks

In this chapter, we focus on the gene regulatory network control problem based on the Boolean network model. Based on the state transition diagram, we introduce the importance and motivation of our algorithm in Section 5.1. Section 5.1 also explains the intuition of our algorithm. Section 5.2 describes how we identify the control kernel. Section 5.3 evaluates the proposed algorithm on six real-world networks. Finally, Section 5.4 summarizes the chapter.

5.1 Introduction

Controlling gene regulatory networks is a crucial topic for understanding the regulation of biological processes. There are many existing approaches have been proposed for finding the control strategies of Boolean networks to lead an initial state to a target state over finite time steps. Datta et al. (2003) have proposed a method using dynamic programming for finding a control strategy for probabilistic Boolean networks. It can also be used to Boolean networks where the control strategy is denoted as a sequence of states of control genes (Akutsu, Hayashida, Ching, et al., 2007). Akutsu, Hayashida, Ching, et al. (2007) have shown that

finding control strategies is NP-hard problem. They also proposed a dynamic programming based algorithm for computing a control strategy given a Boolean network with a tree structure. Other techniques have also been developed, e.g., integer linear programming-based method (Hou, Tamura, et al., 2016), Model Checking techniques (Langmead and Jha, 2009), for identifying the genes and control strategies for driving a Boolean network from an initial state to a desired state.

Recently, a few studies have focused on finding the control kernel (i.e., the minimum set of genes) for controlling the Boolean network where the regulation of the control kernel can drive all initial states to a desired attractor. The control problem was first studied in terms of structural controllability of a linear system (Lin, 1974; Liu et al., 2011). And the recent work proposed by (Wu et al., 2016) defined the minimum driver node sets (MSSs) by analyzing the network structural controllability. The genes in the MSS can be actuated to make the network completely controllable in finite time by suitable inputs. However, the purely structure-method may not be able to be directly applied to all types of Boolean networks (Hou, Tamura, et al., 2016; Gates and Rocha, 2016). Rafimanzelat and Bahrami (2018b) proposed a method based on the semi-tensor product for finding such control kernels. But their method is only suitable for small-scale Boolean networks as it requires a high computational cost to handle exponential size matrices (Hou, Tamura, et al., 2016). Kim et al. (2013) developed an algorithm based on a genetic algorithm for discovering the control kernel. But their algorithm cannot always ensure all possible states converge to the desired attractor. As Boolean networks are non-linear and have many feed-forward and feedback loops, there is a handful of work on controlling Boolean networks (Baudin et al., 2019).

In this paper, we present an algorithm based on a genetic algorithm for finding the control kernel of a Boolean network. We decompose the control problem into two subproblems. Based on the concept of the state transition diagram, the first (global) subproblem aims at defining a global control strategy encoded in a directed tree spanning each basin, where the tree determines the global path leading each basin to the desired attractor. Then

the second (local) subproblem is to identify the control kernel with a sequence of control actions so that by changing the state transitions of the selected genes, any possible initial states in a basin can flow into the desired attractor through above mentioned global path.

To summarize, our main contributions are:

1. We propose a new coding scheme for directed ordered tree in the genetic algorithm. We also design the initialization, crossover and mutation operators for this coding.
2. The control kernel identified by our algorithm guarantees the convergence of any initial network states into the desired state.
3. We applied our algorithm to find the control kernels of six real-world networks. Our experimental results show that our algorithm finds smaller control kernels than other algorithms.

5.1.1 Problem definition and intuition

Our problem is to identify a minimum set of genes as control kernel where by regulating those genes, all possible initial states can flow into the desired attractor through a trajectory. Figure 5.1 presents the original state transition diagram of a Boolean network including three genes where each node is a network state. The desired state is marked as a read node and it is not reachable from any states in basin 1 and basin 2.

Let we denote the hamming distance between two states s_j and s_k as $d(s_j, s_k)$ and the set of genes realise this Hamming distance as $\text{argd}(s_j, s_k)$. Then the minimum set of genes required for a control action forcing s_j to s_k is $\text{argd}(s_j, s_k)$. To make all possible initial states in basin 1 flow into the desired attractor, as we described in the previous section, we can take a control action to flip the state of gene x_2 (i.e., $\text{argd}((0, 0, 0), (0, 1, 0))$) when the network reaches the state $(0, 0, 0)$ such that $(0, 0, 0) \xrightarrow{x_2=1} (0, 1, 0)$. Similarly to basin 2, we can make

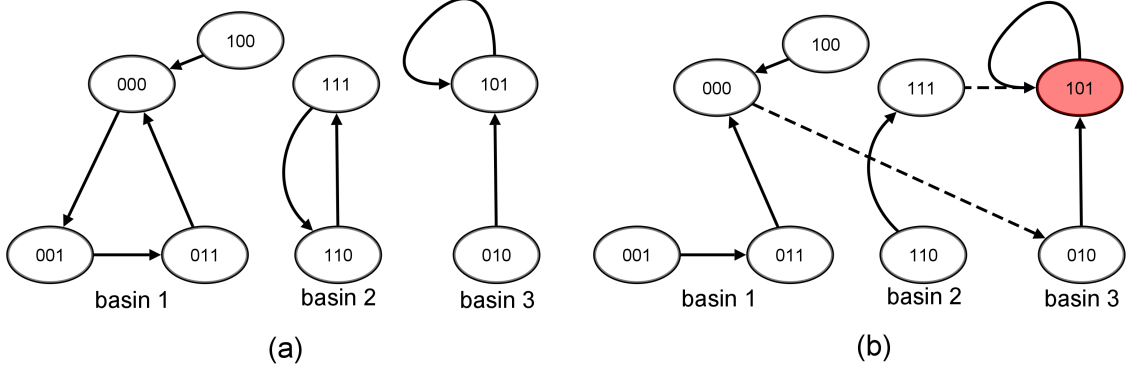


Figure 5.1: An example for identifying the control kernel. (a) The original state transition diagram of a Boolean network. (b) By controlling the second gene when the network reaches particular state, the desired attractor (marked as a red node) is reachable from all possible initial states.

a new trajectory $(1, 1, 1) \xrightarrow{x_2=0} (1, 0, 1)$. As shown in Figure 5.1, we use dashed line to show new trajectories made by our control actions where we need to control only one gene (i.e., x_2) to make the desired attractor be reachable from any initial states, thus $\{x_2\}$ is a control kernel of this Boolean network. We can also take a sequence of control actions (not shown in Figure 5.1) to achieve $(0, 0, 0) \xrightarrow{x_1=1, x_2=1} (1, 1, 0) \rightarrow (1, 1, 1) \xrightarrow{x_1=0, x_3=0} (0, 1, 0) \rightarrow (1, 0, 1)$ by controlling genes $\text{argd}((0, 0, 0), (1, 1, 0)) \cup \text{argd}((1, 1, 1), (0, 1, 0)) = \{x_1, x_2, x_3\}$. But this control strategy needs 3 genes so it is not the solution to our problem. To avoid performing redundant control actions on the initial states in the same basin, it is obvious to see that taking a control action to lead an attractor state to a target state, then all states in the same can transit to the target state.

Therefore we propose a genetic algorithm mainly to find the control kernel by formulating it into two subproblems. The first problem aims at determining global paths along the basin-basin connection towards the desired attractor, where we can take a sequence of control actions to achieve the global paths. The goal of the second problem is to define

control action that should lead any possible initial states to the desired attractor using as few genes as possible. Then the union of sets of genes used in all control actions compose the control kernel.

5.2 Method

In order to identify the control kernel of a Boolean network, we propose a genetic algorithm to solve the global subproblem by defining the global paths connecting each basin. The paths are presented as a directed ordered tree where the root is the basin of the desired attractor and other nodes refer to other basins. The paths in a tree also correspond to a sequence of control actions with a set of controlled genes. Then the local subproblem is to identify the best tree with the minimum number of controlled genes (i.e., control kernel).

The outline of our GA is described as follows: We first initialize a parent population of candidate individuals (i.e., trees). We randomly select two individuals as parents from the population and recombine them to produce new offspring. We then mutate the offspring with a small probability. Finally we select new parent population from the offspring and parent population according to the fitness score. The fitness score depends on the number of controlled genes in an individual and thus determines the best tree. We continue all procedures until we find a control kernel which includes only one gene or a predefined maximum number of iterations is reached.

5.2.1 Encoding scheme for directed ordered tree

Each individual is represented as a directed ordered tree so that we can design a sequence of control actions for achieving each edge in order. The tree describes the state transition diagram after the control. Each node in the tree denotes a basin and each edge denotes a

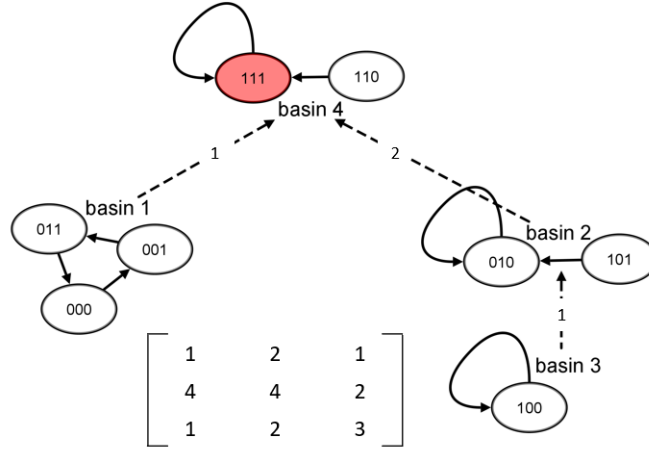


Figure 5.2: An example of a tree and an array to represent it. In the tree, each basin corresponds to a node and each edge between two basins is presented by the dashed arrow. The number attached in an edge specifies the order for the children of a basin. The lower part presents this tree as an array consisting of three strings.

trajectory from a child node to its parent.

Inspired by the edge sets encoding (Raidl and Julstrom, 2003) in which a tree can be represented by two strings, we extend their work and propose a new efficient coding scheme with an additional string to encode the ordering of each node. Given a Boolean network including m basins, we label each basin as $B_1, B_2, \dots, B_{m-1}$ and particularly the basin of the desired attractor as B_m . We then introduce an array of three strings $O = \{o_1, o_2, \dots, o_{m-1}\}$, $P = \{p_1, p_2, \dots, p_{m-1}\}$ and $C = \{c_1, c_2, \dots, c_{m-1}\}$ to represent a directed ordered tree. Each string consists of exactly $m - 1$ integers from $\{1, 2, \dots, m - 1\}$. c_i and p_i denote a directed edge pointing from the B_{c_i} to B_{p_i} . And o_i represents B_{c_i} is the o_i th child of B_{p_i} .

To illustrate our representation, we present a tree with 4 nodes (i.e., 4 basins) in Figure 5.2, where the tree edge is represented by the dashed arrow. We can form an array $T(O, P, C)$

shown in the lower part of Figure 5.2 to present the directed ordered tree: $\begin{bmatrix} 1 & 2 & 1 \\ 4 & 4 & 2 \\ 1 & 2 & 3 \end{bmatrix}$ where

the top row is $O = \{o_1, o_2, o_3\} = \{1, 2, 1\}$, the middle row is $P = \{p_1, p_2, p_3\} = \{4, 4, 2\}$ and

the bottom row is $C = \{c_1, c_2, c_3\} = \{1, 2, 3\}$. Each edge is interpreted by a column in T . In this example, o_1 and o_2 specify that B_1 and B_2 are the first and the second child of B_4 , respectively. o_3 denotes that B_3 is the first child of B_2 .

5.2.2 Algorithm

Initial population We start our algorithm by initializing the population with a collection of trees, where each tree determines the global control strategy (i.e., the solutions to the global subproblem). We describe the procedure of initialization as described in Algorithm 5.1 based on randomized version of Prim's algorithm (Prim, 1957). We first set the desired basin as the root of the tree. To make the tree grow without cycles and duplicates, we repeat to randomly select a node (i.e., a basin) which is not in the tree and append it to a random node in the tree, while P and C are updated correspondingly. In each iteration, we also remark the ordering of each child node and save it in O . We stop the repetition when the tree includes all nodes.

Crossover The crossover operator is used to recombine two parents from the population for generating new offspring. The basic idea of the crossover operator is to create new global paths among basins, where we preserve common tree edges as well as random tree edges from two parents.

We describe the crossover operators in Algorithm 5.2 and present an example in Figure 5.3 with two parents $T^1 = \begin{bmatrix} 2 & 1 & 1 & 1 & 2 \\ 6 & 3 & 6 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}$ and $T^2 = \begin{bmatrix} 1 & 1 & 3 & 1 & 2 \\ 2 & 6 & 6 & 3 & 6 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}$. Similar to Algorithm 5.1, we start to generate new offspring from the tree rooted with the desired basin. We first choose a parent B_i from the eligible parent set V' (i.e., $\{6\}$). Then we select its child from either T^1 or T^2 (i.e., $\{1, 3\} \cup \{2, 3, 5\}$) whose parent is v . The selected child must not appear in T . If we choose $u = 3$ to remark the first child, then we make $o_u = 1$,

Algorithm 5.1: Population Initialization

Input: A state transition diagram in which basins are labelled from 1 to m and the desired attractor is in the m th basin;

Output: An individual T of three vectors with length of $m-1$;

- 1 Initilize $O' = (o'_1, o'_2, \dots, o'_m) = (0, \dots, 0)$, and $M = \{1, 2, \dots, m\}$;
 - 2 Initilize a tree $T(O, P, C)$ of three empty strings O, P, C ;
 - 3 $V \leftarrow \{m\}$;
 - 4 **while** $V \neq M$ **do**
 - 5 $v \leftarrow$ Randomly choose a value from V ;
 - 6 $o'_v \leftarrow o'_v + 1$;
 - 7 $u \leftarrow$ Randomly choose a value from $M \setminus V$;
 - 8 $c_u \leftarrow u$;
 - 9 $o_u \leftarrow o'_v$;
 - 10 $p_u \leftarrow v$;
 - 11 $V \leftarrow V \cup \{u\}$
-

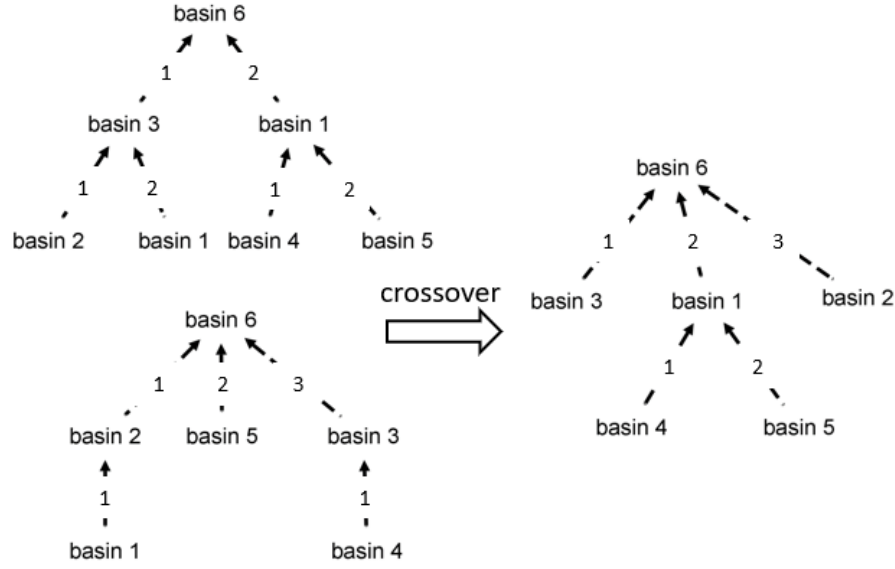


Figure 5.3: An example of crossover operation by recombining two parent trees to produce one offspring.

$p_u = 6$, and $c_u = 3$ to represent B_3 is the first child of B_6 . We then update V' to $V' \cup \{u\}$ where updated V' consists of the nodes in T and at least one child in T^1 or T^2 . We repeat to add more nodes into the tree until all nodes are included in the new tree and finally obtain

$$T = \begin{bmatrix} 2 & 3 & 1 & 1 & 2 \\ 6 & 6 & 6 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}.$$

Mutation We describe our mutation operator in Algorithm 5.3 for creating diversity that may not have been present in the population. We extract one random edge ($B_{c_i} \Rightarrow B_{p_i}$) from the tree after crossover. It makes the tree become two separated subtrees. One subtree is rooted with B_i and another one is created by removing the former subtree from the original tree. We then build a new edge between these two subtrees.

$$\text{Given a parent tree } T^1 = \begin{bmatrix} 2 & 1 & 3 & 1 & 1 & 2 & 1 & 2 \\ 2 & 9 & 9 & 3 & 2 & 9 & 4 & 3 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix} \text{ as shown in Figure 5.4, we first}$$

select a random eligible node $i = 4$ and then remove the corresponding edge ($B_4 \Rightarrow B_3$).

Algorithm 5.2: Crossover

Input: Parent trees $T^1(C^1, P^1, O^1)$ and tree $T^2(C^2, P^2, O^2)$;

Output: Offspring tree $T(C, P, O)$;

1 Initilize $O' = (o_1, o_2, \dots, o'_m) = (0, 0, \dots, 0)$; $M \leftarrow \{1, 2, \dots, m\}$; $V, V' \leftarrow \{m\}$;

2 Initilize a tree T with three empty strings O, P, C with length of $m - 1$;

3 **while** $V \neq M$ **do**

4 $v \leftarrow$ Randomly choose a value from V' ;

5 $o'_v \leftarrow o'_v + 1$;

6 $u \leftarrow$ Randomly choose a value from $U_1 = \{u \mid p_u^1 = v, u \in M \setminus V\} \cup$

$U_2 = \{u \mid p_u^2 = v, u \in M \setminus V\}$;

7 **if** $U_1 = \emptyset$ *and* $U_2 = \emptyset$ **then**

8 $V' \leftarrow V' \setminus \{v\}$;

9 **else**

10 $p_u \leftarrow v$;

11 $c_u \leftarrow u$;

12 $o_u \leftarrow o'_v$;

13 $V \leftarrow V \cup \{u\}$;

14 $V' \leftarrow V' \cup \{u\}$

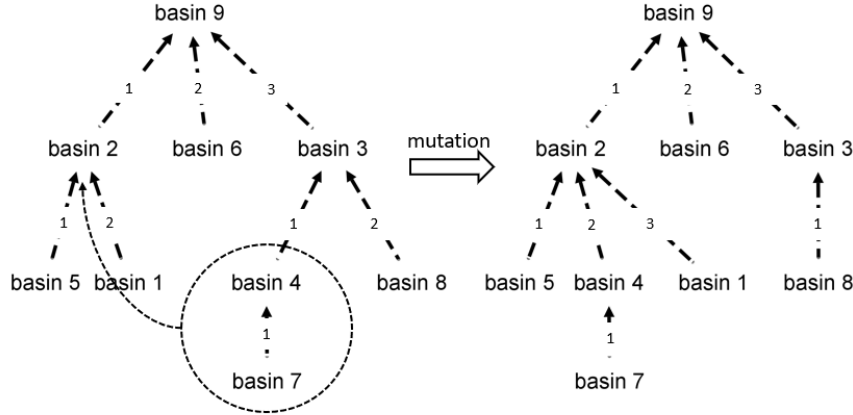


Figure 5.4: An example of mutation operator by removing a subtree and appending this subtree.

As a child node of node p_i has been removed, we then re-order the left children nodes by subtracting 1 from the order value which is greater than o_4 . After that, we randomly choose a new parent node $k = 2$ for node 4 and build a new edge ($B_4 \Rightarrow B_2$). Accordingly, we change p_4 to be 2. As a new child node 4 has been appended to node 2, we have to reorder the ordering of children of node 2. We first get eligible order values from $\{1, 2, \dots, L\}$ where L refers to the number of children nodes. Then we select a random value to be the ordering of node 4 and change other order values accordingly. Finally we obtain the offspring

$$T = \begin{bmatrix} 3 & 1 & 3 & 1 & 1 & 2 & 2 & 1 \\ 2 & 9 & 9 & 3 & 2 & 9 & 4 & 3 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}.$$

Fitness Evaluation The fitness evaluation identifies the beset tree in the local subproblem. The fitness evaluates the number of genes used for achieving the paths in an individual. Each individual (i.e., a tree) only determines a path from a basin towards the desired attractor. Thus we need to identify a sequence of control actions to lead any initial states in a basin to the desired attractor through the path. The genes used in control actions compose the control kernel.

As shown in Algorithm 5.4, we determine a sequence of control actions in level-order

Algorithm 5.3: Mutation

Input: Parent tree $T^1(C^1, P^1, O^1)$;

Output: Offspring tree $T(C, P, O)$;

```

1  $T \leftarrow T^1$ ;
2  $M \leftarrow \{1, \dots, m\}$ ;
3  $u \leftarrow$  select a random values from  $M \setminus \{m\}$ ;
4  $V \leftarrow \{v \mid p_v = p_u, v \in V\}$ ;
5 while  $p_u = \{m\}$  and  $|V| = 1$  do
6    $u \leftarrow$  select a random value from M;
7    $V \leftarrow \{v \mid p_v = p_u, v \in V\}$ ;
8 foreach  $v, v \in V$  do
9   if  $o_v > o_u$  then
10     $o_v \leftarrow o_v - 1$ ;
11  $M \leftarrow M \setminus V$ ;
12  $v \leftarrow$  select a random value from M;
13  $p_u \leftarrow v$ ;
14  $L \leftarrow |\{l \mid p_l = v\}|$ ;
15  $l \leftarrow$  select a random value from  $\{1, 2, \dots, L\}$ ;
16 foreach  $u, u \in \{u \mid p_u = v\}$  do
17   if  $o_u \geq l$  then
18     $o_u \leftarrow o_u + 1$ ;
19  $o_u \leftarrow l$ ;
```

and identify the control kernel recursively. For each edge leading B_u to B_v , our control action is to force a state in A_u to B_v with genes $\arg \min_{x_i \in X} |ck \cup \arg d(s_j, s_k)|$, $s_j \in A_u, s_k \in B_v$. Ideally, the number of genes in the control kernel is unchanged (i.e., $\arg d(s_j, s_k) \subseteq ck$) while we design new control action for the next pair of basins. In case that we cannot find a suitable control action to connect two basins by only using the genes in the existing ck , we consider an alternative control strategy with two control actions. Specifically, for an edge from B_u to B_v , the first control action is to force $s_{jj}, s_{jj} \in B_u \setminus A_u$ to B_v . It results in all states reach s_{jj} in the original state transition will fall into B_v eventually. Then the second control action is to force an attractor state s_j to s_{jjj} where s_{jjj} is a state can reach s_{jj} following the original state transition. This control strategy searches for a minimum set of controlled genes by computing $\arg \min_{x_i \in X} |ck \cup h(s_j, s_{jjj}) \cup h(s_{jj}, s_k)|$, $s_{jj}, s_{jjj} \in B_u \setminus A_u, s_k \in B_v$. The control strategy using fewer genes will be used. Finally, the union of sets of genes required in all control actions composed the control kernel.

Selection After the offspring population is generated, we perform the selection based on the $(\mu + \lambda)$ principle. In each generation, we generate λ offspring using crossover and mutation operators from the population of μ individuals. Then the selection is performed by discarding λ individuals with the worst fitness score out of total $(\mu + \lambda)$ individuals. Then the population size is constant as μ for the next generation. In this way, the best individuals can always survive.

5.3 Results

5.3.1 Experimental design

We have applied our algorithm to six real-world gene regulatory networks. These networks include the budding yeast cell-cycle regulatory network (Li et al., 2004), fission yeast cell-

Algorithm 5.4: Fitness evaluation

Input: A tree $T(O, P, C)$ of $G(X, F)$

Output: The number of genes in the ck

```

1  $ck \leftarrow \emptyset$ ;
2  $Q \leftarrow$  Create a queue;
3 Add  $m$  in  $Q$ ;
4  $pV \leftarrow 1$ ;
5 while  $Q$  is not empty do
6    $pV \leftarrow$  Gets the number of elements in  $Q$ ;
7   while  $pV > 0$  do
8      $u \leftarrow$  Get the head of  $Q$ ;
9     Remove the head of  $Q$ ;
10     $cV \leftarrow \{v | p_v = u\}$ ;
11    for  $r = 1$  to  $|cV|$  do
12       $v \leftarrow$  get  $v$  that  $v \in cV$  and  $o_v = r$ ;
13      Add  $v$  at the tail of  $Q$ ;
14       $ck \leftarrow$  determine the genes used in a control strategy for leading  $B_u$  to
15       $B_v$ ;
15     $pV \leftarrow pV - 1$ ;

```

Table 5.1: Gene regulatory networks used in our experiment

Benchmark Networks	Number of genes	Number of attractors
Budding yeast cell-cycle regulatory network	11	7
Fission yeast cell-cycle regulatory network	9	16
GRN underlying mammalian cortical area	10	3
GRN underlying mouse myeloid development	11	8
ERBB-receptor regulated network	20	-
CREB signalling network	64	-

cycle regulatory network (Wang et al., 2010), the gene regulatory network (GRN) underlying mammalian cortical area (Giacomantonio and Goodhill, 2010), the GRN underlying mouse myeloid development (Krumsiek et al., 2011), the ERBB-receptor regulated network (Sahin et al., 2009) and the CREB signalling network (Abdi et al., 2008). The details of these networks are present in Table 5.1. In our experiments, we set μ as the number of attractors and λ equals 2μ . The maximum number of iterations is set as 200 and the mutation probability is 0.05.

For small-scale networks with the number of genes no more than 16 (Kim et al., 2013), we used the whole state transition diagram. We also enumerate all possible trees for the network only includes 3 attractors. For larger-scale networks, we used the genetic algorithm to find the best tree based on a number of sampled state transitions from many different random initial states (10,000 initial states were randomly sampled in this study). The desired attractor was chosen according to (Kim et al., 2013). We compare our results with other three existing control kernel identification algorithms and we denote them as 'CKGA' (Kim et al., 2013), 'MSS' (Wu et al., 2016), 'CKSTP' (Rafimanzelat and Bahrami, 2018b) in this section. Due to the unavailable implementation of their algorithms, we collected the results from the literature for comparison.

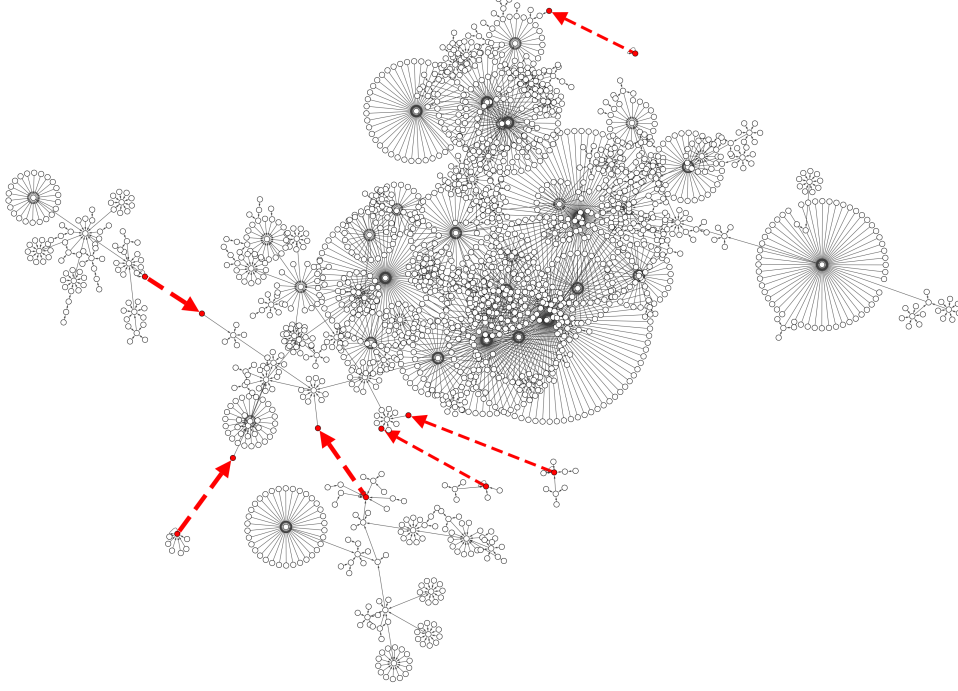


Figure 5.5: The state transition diagram of the budding yeast cell-cycle regulatory network after regulating the genes of the control kernel. Each node denotes a network state and each black arrow denotes the original state transition between two states. The attractor with the largest basin is used as the desired attractor. All red arrows constitute the global paths defined by our algorithm.

5.3.2 Performance on control kernel identification

Our experiment results are shown in Table 5.2. We observed that the number of genes in the control kernel is much smaller than the total number of genes in the networks. By controlling at most 11% genes in a network can result in a state transition diagram in which all possible initial states can flow into the desired attractor.

For the budding yeast cell-cycle regulatory network, our method found three control kernels where each includes only one gene. MSS and CKSTP also discovered control kernels with the same size. But CKGA identified larger control kernels of 4 genes. In Figure 5.5, we present the state transition diagram by taking control actions with gene *Cln3*. We can see

Table 5.2: The control kernel found for real-world gene regulatory networks. Our method were compared with CKGA, MSS and CKSTP. The symbol ‘-’ indicates that the corresponding results cannot be collected.

Networks	Our Method	CKGA	MSS	CKSTP
Budding yeast	{Cln3} or {Clb5,6} or {Clb1,2}	{Cdh1, MBF, SBF, Sic1}	{Cln3}	{Cln3} or {Clb5,6} or {Clb1,2}
Fission yeast	{Cdc2/13}	{Cdc25, Rum1, Ste9, Wee1}	{SK, PP} or ..	-
Mammalian	{Fgf8_g} or {Emx2_g} or {Sp8_g}	{Fgf8_g} or {Emx2_g} or {Sp8_g} or ...	-	-
Mouse myeloid	{GATA_1}	{GATA_1, EKLF, Fli_1} or {EKLF, Fli_1, PU1}	{CEBPa}	{GATA_1, GATA_2, CEBPa} or {GATA_1, GATA_2, PU1} or ...
ERBB-receptor	{ErbB2_3}	{MEK1} or {IGF1R}	-	-
CREB	{Calmodulin, PKA, cAMP} or {Calmodulin, Ca2plus, cAMP} or {Calmodulin, AC2, cAMP}	{Calmodulin, AC2, AC5, M2R, PKA} or {Calmodulin, CREB, CaMKI, Galphai, PKA}	-	-

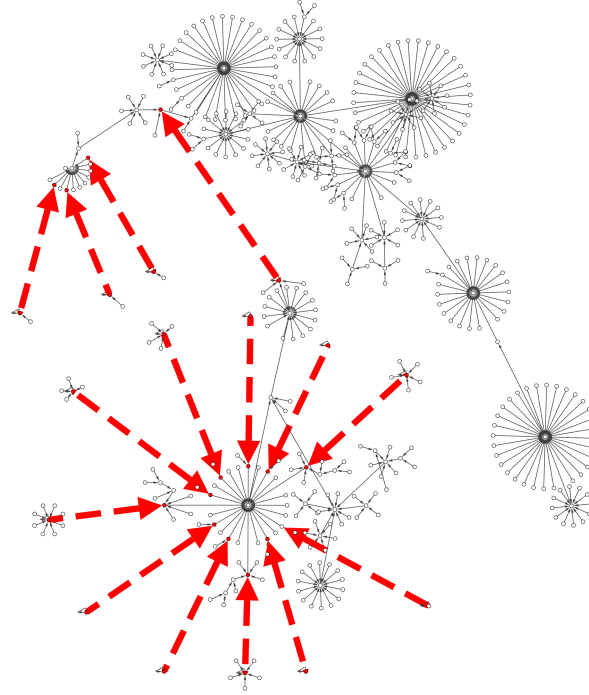


Figure 5.6: The state transition diagram of the fission yeast cell-cycle regulatory network after regulating the genes of the control kernel. Each node denotes a network state and each black arrow denotes the original state transition between two states. The attractor with the largest basin is used as the desired attractor. All red arrows constitute the global paths defined by our algorithm.

that the desired attractor is reachable from any possible initial states.

In Figure 5.6, we show that the state transition diagram of the fission yeast cell-cycle regulatory network after controlling the control kernel $\{Cdc2/13\}$. There are 11 basins connecting to the secondary largest basin and 4 basins directly connected to the desired attractor. Compared to CKGA and MSS, the control kernel identified by our algorithm includes only 1 gene, but CKGA and MSS identified 4 genes and 2 genes as the control kernel, respectively.

Figure 5.7 depicts all state transitions by regulating the state of gene $Fgf8_g$ of the GRN underlying mammalian cortical area. Figure 5.8 shows all state transitions by

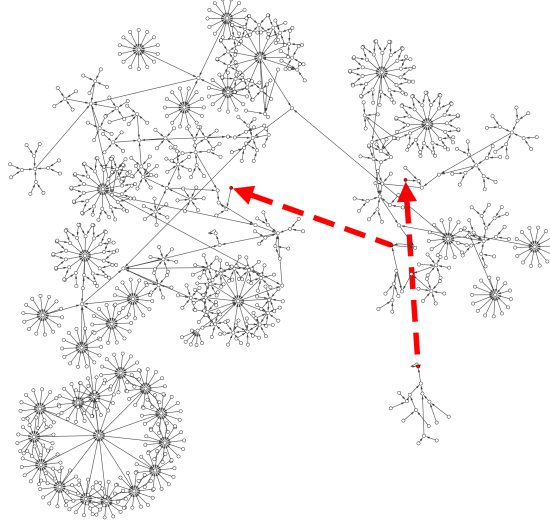


Figure 5.7: The state transition diagram of the gene regulatory network underlying mammalian cortical area after regulating the genes of the control kernel. Each node denotes a network state and each black arrow denotes the original state transition between two states. The attractor with the largest basin is used as the desired attractor. All red arrows constitute the global paths defined by our algorithm.

controlling GATA1, we can steer all states to the desired attractor. For this GRN underlying mouse myeloid development, MSS also defines a control kernel with the same size. But both CKGA and CKSTP found a control kernel including three genes.

Figure 5.9 presents the control strategy for the ERBB-receptor regulated network. Compared to CKGA, we defined a control kernel with the same size as CKGA. Figure 5.10 indicates the station transition diagram of CREB signalling network which includes 16 attractors/basins. We can also observe that this control strategy corresponds to a spanning tree with 4 levels. Compared to CKGA which defined two control kernels that each includes 5 genes, we found 3 control kernels that each contains 3 genes.

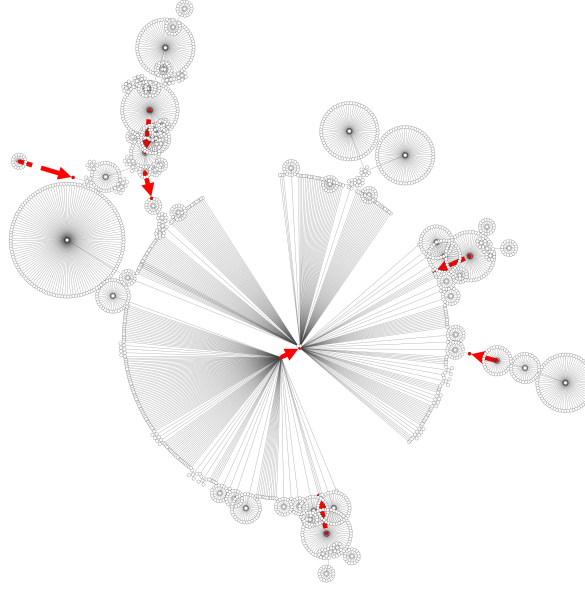


Figure 5.8: The state transition diagram of the gene regulatory network underlying mouse myeloid development after regulating the genes of the control kernel. Each node denotes a network state and each black arrow denotes the original state transition between two states. The attractor with the largest basin is used as the desired attractor. All red arrows constitute the global paths defined by our algorithm.

5.4 Summary

In this chapter, we propose a novel algorithm to find the control kernel of a gene regulatory network. Our proposed algorithm decomposes the control problem into two subproblems. The first problem is to determine the global paths spanning the attractors using a genetic algorithm with a novel encoding of directed ordered tree, while the second subproblem aims at identifying the specific control actions to find the best global paths which are achieved by regulating the minimum number of genes.

Based on the concept of state transition diagram, we present a new coding of directed ordered tree in a genetic algorithm, where the node denotes the basin and the edge refers to a trajectory from a child node to its parent. All edges correspond to a sequence of control

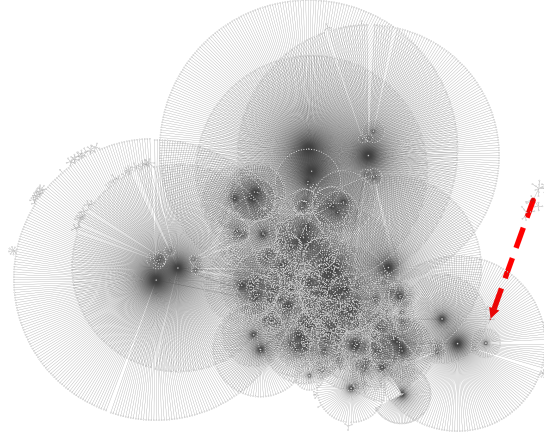


Figure 5.9: The state transition diagram of the ERBB-receptor regulated network after regulating the genes of the control kernel. Each node denotes a network state and each black arrow denotes the original state transition between two states. The attractor with the largest basin is used as the desired attractor. All red arrows constitute the global paths defined by our algorithm.

actions and a set of genes required for making the desired attractor reachable from any initial state. In the genetic algorithm, we introduce the initialization, selection, crossover and mutation operators to generate a collection of trees. Finally, the best tree corresponding to the minimum number of controlled genes determines the control kernel. We applied our algorithm to six real-world gene regulatory networks and found smaller control kernels than other existing control algorithms.

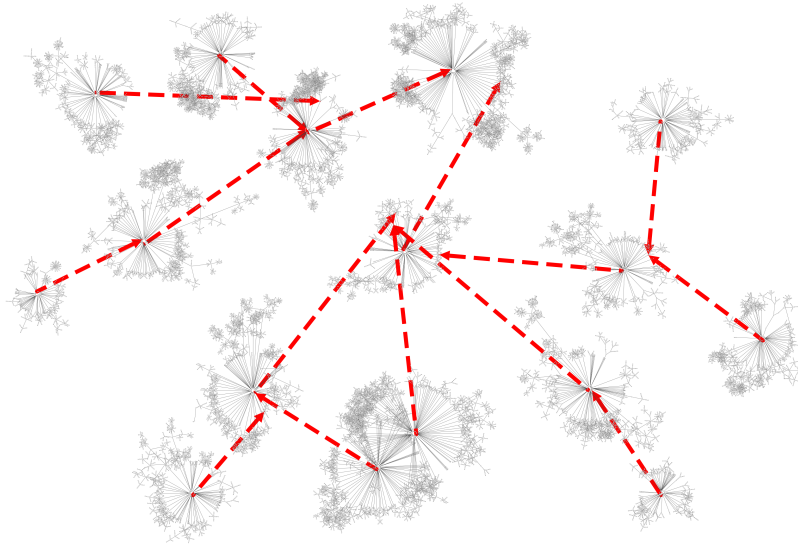


Figure 5.10: The state transition diagram of the CREB signalling network after regulating the genes of the control kernel. Each node denotes a network state and each black arrow denotes the original state transition between two states. The attractor with the largest basin is used as the desired attractor. All red arrows constitute the global paths defined by our algorithm.

Chapter Six

Conclusion and discussion

This chapter summarizes this thesis and presents the limitations of our proposed algorithm and points out potential directions related to the work presented in this thesis.

6.1 Conclusion

This thesis focuses on the problem of inferring and controlling gene regulatory networks based on the Boolean network model. Chapter 1 first introduces the background of gene regulatory networks and Boolean networks and presents three key research questions in Sections 1.4. Chapter 2 presents the literature review of existing works on Boolean network inference and control. Chapters 3, 4 and 5 are proposed for answering the three research questions, respectively.

Chapter 3 aims at addressing the first research question, i.e., how to infer both Boolean network topology and its dynamics from short and noisy gene expression data. In Chapter 3, we propose an And/Or tree ensemble algorithm including both optimization approach and feature selection approach. Due to a large number of genes with small datasets, we propose to infer the predictive prime implicants as features to reconstruct the Boolean function. We

introduce the sub-sampling and ensemble process to deal with small noisy datasets. The effectiveness of our algorithm has been validated on both artificial network and real-world network.

Chapter 4 proposes a genetic algorithm with marker-based encoding for answering the second research question, i.e., how to improve the scalability and accuracy of Boolean network inference. Instead of encoding each Boolean function with a chromosome, the proposed algorithm encodes the whole network with all Boolean functions into a chromosome, where the input genes and logical operators are represented by integers. Different target genes are distinguished by the markers. More logical operators are introduced to help the genetic algorithm efficiently infer the complex Boolean networks. The performance of the proposed algorithm has been validated on both artificial network and real-world network.

Chapter 5 aims to address the third research question, how to find the control kernel for a Boolean network. In Chapter 5, the control kernel identification problem has been decomposed into two subproblems. The global problem is to define the global paths along basin-basin connection leading each basin to the target attractor. To this end, we develop the genetic algorithm to provide a collection of directed ordered trees spanning each basin, where the tree is directed ordered tree and encoded by a novel and efficient encoding scheme. The local problem is to select the best tree that a set of controlled genes is required to achieve the global paths of the tree. We then propose to recursively identify the controlled genes until the target basin is reachable from any possible states. The proposed algorithm has been applied to control six real-world networks.

6.2 Discussion

This last section presents the limitations of our proposed algorithm and discusses potential research directions in the future.

In Chapter 3, we propose ATEN, an And/Or tree ensemble algorithm for inferring both network topology and its dynamics. ATEN uses simulated annealing for searching and reconstructing features (i.e., prime implicants) and the final Boolean functions. However, it is not efficient for ATEN to build a larger tree and find the complex Boolean function (e.g., NXOR). We have discussed the upper limit on tree size (i.e., the maximum number of input genes) in the supplementary, but have not investigated how many trees should be reconstructed in the ensemble. Since the feature is extracted from an ensemble of And/Or trees, we may miss the important features if only a few trees are reconstructed. On the other hand, inferring a large number of trees requires high computational cost.

In Chapter 4, we propose MGANI, a genetic algorithm with marker-based encoding for network inference. MGANI encodes an entire network in a candidate solution and is efficient for inferring complex Boolean function. However, how to select the parameters involved in the genetic algorithm is a problem for MGANI. And genetic algorithm cannot guarantee that the inferred network is optimal.

Chapter 5 proposes a genetic algorithm for identifying the control kernel for Boolean networks. Although our algorithm enables the inferred control kernel to make all possible initial states flow into the desired attractor, but cannot guarantee there exists no smaller control kernel. Another limitation is that our algorithm is not easy to scale up to large-scale networks, because the number of network states grows exponentially as the number of genes increases.

All our algorithm are designed based on the Boolean networks with synchronous updating scheme due to its computational efficiency. However, asynchronous updating scheme is more biological plausible but they are more complex to analyze and model (Garg et al., 2008). The most effective choice in asynchronous networks is to make most one gene update in a single time point (Saadatpour et al., 2010). Thus in the state transition diagram, each network state could flow into multiple network states. It is possible to utilize Boolean state

space scoring function (Lim et al., 2016) for inferring asynchronous Boolean networks in our future work. For the control problem, we need to consider that for an initial state, it is possible that there are many paths from it to other attractors. Therefore, we could classify the states into two subsets of states: one subset includes the states only flow into one attractor and one subset includes other states, and then propose new algorithms to handle the second subsets.

Another interesting direction is to extend our algorithms to infer and control probabilistic Boolean networks (PBNs) (Shmulevich, Dougherty, Kim, et al., 2002; Shmulevich, Dougherty, and Zhang, 2002), which improves Boolean networks for handling noise and uncertainty. The PBN allows each target gene to be associated with multiple Boolean functions with corresponding selection probabilities. Thus both the Boolean function and its selection probability needs to be reconstructed from the time series data. One way to address this problem is to identify the pseudo-time ordering of a time series so that our two network inference algorithms can be used to infer probabilistic Boolean networks. However, since the gene expression data is typically short and noisy, new theories and algorithms are required to be developed.

The scalability has been a major challenge for inferring and controlling large-scale gene regulatory networks. For both network inference and control problem, it would be interesting to investigate whether other alternative algorithms (e.g., swarm intelligence) could be more computationally efficient. We also consider formulating the problem as a multi-objective optimization problem so that cooperative co-evolutionary algorithm can be applied. For the network control problem, our algorithm relies on the simulations of state transition diagram where the size of the state space grows exponentially with the number of genes, thus how to develop efficient algorithms to control large-scale networks is important.

Appendix One

Supplementary to Chapter 3

A.1 Upper limit on tree size

The upper limit on tree size, which is defined as the maximum number of input genes in the tree, is an important control parameter of ATEN for inferring gene regulatory networks. To determine the value of this parameter, one can use prior knowledge, e.g., the maximum number of regulators (input genes) in the network to be inferred. However, this prior knowledge is usually not available and setting the value is challenging. If the value of the tree size is too large, we essentially increase the search space, making the inference problem more difficult which leads to large number of false positives or even infeasible computational time. On the other hand, if we set the value too small, then we essentially restrict our search space, which will introduce a lot of false negatives, i.e., missing many true interactions. We found that other control parameters do not affect the inference results significantly. These parameters, e.g. the cooling scheme in the SA algorithm, are detailed in ATEN package <https://github.com/ningshi/ATEN>.

To determine the tree size, we first used experiments to investigate how this parameter affects ATEN's performance. We used the largest artificial network with 150 nodes to

generate noisy data with different numbers of time points. We then inferred the network use ATEN with different upper limits on tree size, i.e., 3, 4, 5, 6, 8 and 10. The corresponding results are given in Figure A.1.

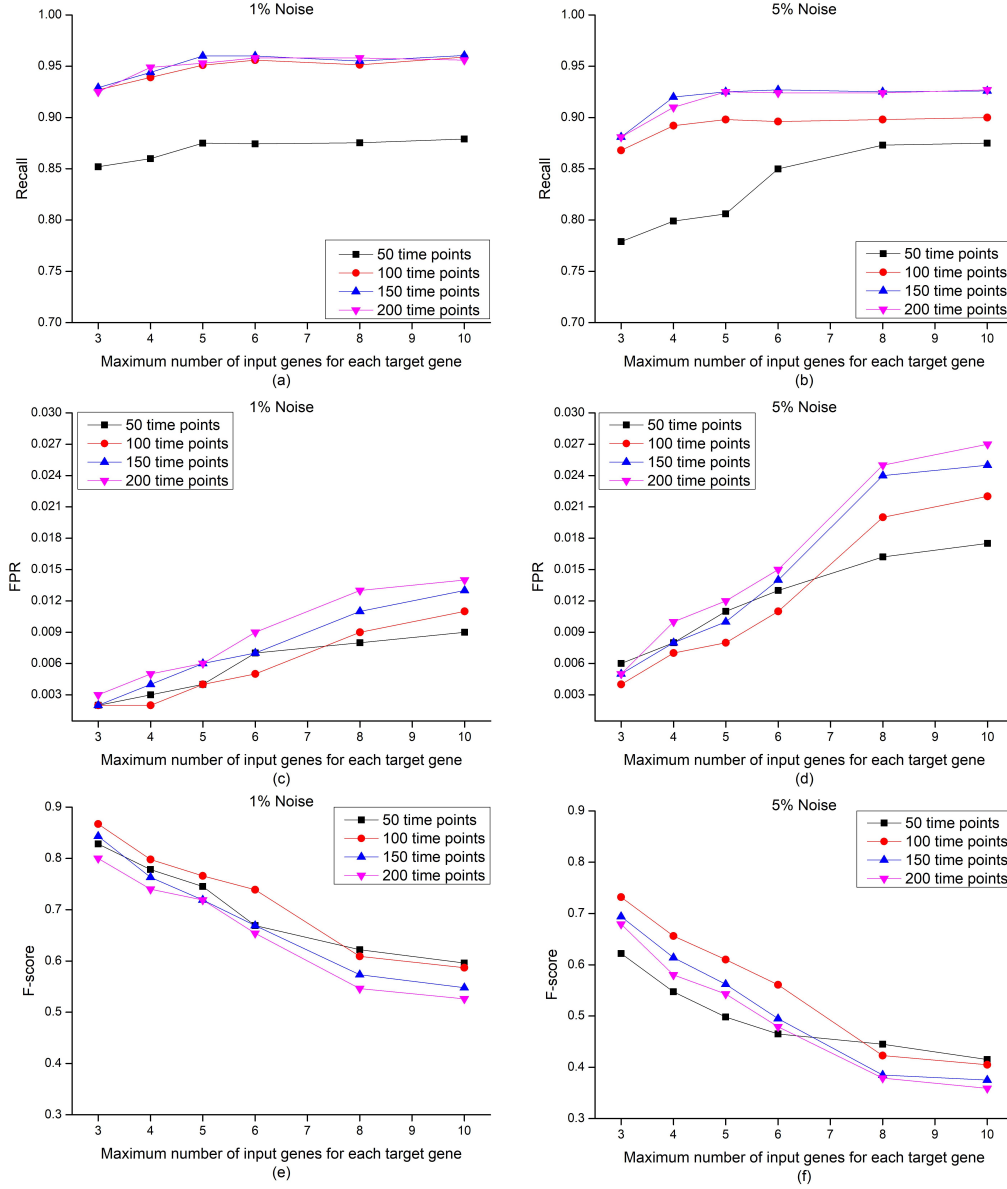


Figure A.1: Experimental results of ATEN for the inferred network with 150 nodes from the datasets including 1% noise and 5% noise. The results are presented using average values. The maximum number of input genes for each target gene was limited to 3, 4, 5, 6, 8 and 10, respectively.

Figure A.1 shows that if we increase the upper limit on tree size, (i.e., the maximum number of input genes), Recall (or known as true positive rate) and False Positive Rate (FPR) increased but F-score decreased. This is because large tree size will introduce more true positive interactions as well as more false positive interactions. However, when the upper limit is larger and equal to 8, Recall saturates (See Figure A.1 (a)-(b)), while FPR increases gradually and F-score decreases steadily (See Figure A.1 (c)-(f)). These results suggested that, when we do not have any prior knowledge about the network topology, setting the upper limit on the tree size to 8 can achieve the best trade-off between true positive and false positive interactions.

A.2 Experimental results

In this supplementary material, we present the performance of inferring networks with 50, 100 and 150 nodes where ran ATEN with the upper limit on tree size equal to the known maximum in-degree of the networks.

A.2.1 Performance on artificial network

We set the upper limit on tree size as 5. We summarize the experimental results in Tables A.1-A.6. Tables A.1, A.3 and A.5 show that Best-Fit achieved the best performance when the data is noise-free. From Tables A.2, A.4 and A.6, when the data is noisy, ATEN achieved the best overall performance.

Table A.1: Experimental results of ATEN, MIBNI and Best-Fit for the artificial networks with 50 nodes inferred from the noise-free data. The maximum in-degree of each target gene inferred by ATEN, MIBNI and Best-fit was limited to 5. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.

Number of Time Points	Algorithm	Noise 0%		
		Recall	FPR	F-score
50	ATEN	0.933±0.017	0.004 ±0.001	0.882 ±0.016
	MIBNI	0.787	0.083	0.309
	Best-Fit	0.983	0.000	0.992
100	ATEN	0.928±0.018	0.003 ±0.001	0.896±0.029
	MIBNI	0.803	0.082	0.315
	Best-Fit	0.983	0.000	0.984
150	ATEN	0.939±0.017	0.004±0.001	0.921±0.027
	MIBNI	0.803	0.082	0.315
	Best-Fit	0.967	0.001	0.967
200	ATEN	0.931 ±0.013	0.004±0.001	0.887±0.016
	MIBNI	0.787	0.083	0.309
	Best-Fit	0.967	0.001	0.967

A.2.2 Performance on real-world network

We also ran ATEN with the upper limit on tree size set to 4, i.e., the maximum in-degree of the real-world network. From Table A.7, when the data is noise-free, we can see that Best-Fit outperformed than other algorithms. From Table A.8, when the data includes noise, ATEN always obtained the best F-score values. Compared ATEN with Best-Fit, we find ATEN obtained worse Recall values when the datasets including 100 and 150 time points with 1% noise. However, ATEN achieved better FPR and F-score values than Best-Fit. Compared with the Recall and FPR values of REACT, we can see that ATEN achieved more stable

Table A.2: ATEN inferred more accurate artificial network topology from short and noisy data. Experimental results of ATEN, MIBNI and Best-Fit for the networks with 50 nodes inferred from noisy data. The maximum in-degree of each target gene inferred by ATEN, MIBNI and Best-fit was limited to 5. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.

Number of Time Points	Algorithm	Noise 1%			Noise 5%		
		Recall	FPR	F-score	Recall	FPR	F-score
50	ATEN	0.918 ±0.016	0.013±0.002	0.747±0.028	0.831±0.030	0.023±0.002	0.606±0.026
	MIBNI	0.786	0.083	0.309	0.787	0.083	0.309
	Best-Fit	0.770	0.032	0.503	0.623	0.063	0.287
100	ATEN	0.938±0.014	0.019±0.003	0.700±0.033	0.875±0.023	0.032±0.003	0.554±0.028
	MIBNI	0.803	0.082	0.315	0.820	0.082	0.322
	Best-Fit	0.950	0.050	0.483	0.820	0.081	0.325
150	ATEN	0.928 ±0.021	0.022±0.002	0.665±0.019	0.856±0.045	0.041 ±0.002	0.486±0.025
	MIBNI	0.803	0.082	0.315	0.803	0.082	0.315
	Best-Fit	0.918	0.055	0.446	0.820	0.082	0.321
200	ATEN	0.884±0.017	0.029±0.002	0.590±0.016	0.925±0.020	0.038±0.002	0.534±0.014
	MIBNI	0.770	0.083	0.302	0.787	0.083	0.309
	Best-Fit	0.902	0.067	0.393	0.934	0.078	0.369

results. Moreover, from Table A.8, it can be concluded that ATEN outperformed the other three algorithms when the data was shorter (i.e., 50 time points) with higher noise level (i.e., 5% noise). Comparing Table A.8 with Figure 3.4 in the main text, we can find that ATEN with a smaller upper limit on tree size (i.e., 4) achieved worse Recall values but better FPR and F-score values.

Furthermore, the overall performance of ATEN for predicting the dynamic model was better than MIBNI, Best-Fit and similar to REACT. From Table A.9, We can observe that ATEN always retrieved at least 40% steady states, even when the input data was short and noisy (i.e., 50 time points and 5% noise). Compared with REACT, ATEN correctly retrieved the same number of steady states. Comparing Table A.9 with Table 3.10 in the main text , we can find that ATEN with a larger upper limit on tree size (i.e., 8) retrieved more steady states.

Table A.3: Experimental results of ATEN, MIBNI and Best-Fit for the artificial networks with 100 nodes inferred from the noise-free data. The maximum in-degree of each target gene inferred by ATEN, MIBNI and Best-fit was limited to 5. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.

Number of Time Points	Algorithm	Noise 0%		
		Recall	FPR	F-score
50	ATEN	0.947±0.007	0.002 ±0.000	0.913±0.012
	MIBNI	0.828	0.040	0.338
	Best-Fit	0.945	0.001	0.953
100	ATEN	0.972±0.007	0.001 ±0.000	0.933±0.017
	MIBNI	0.852	0.040	0.347
	Best-Fit	1	0	1
150	ATEN	0.967±0.015	0.002±0.000	0.928±0.014
	MIBNI	0.844	0.040	0.344
	Best-Fit	0.992	0.000	0.992
200	ATEN	0.974 ±0.009	0.002±0.000	0.930±0.015
	MIBNI	0.844	0.040	0.344
	Best-Fit	1	0	1

Table A.4: ATEN inferred more accurate artificial network topology from short and noisy data. Experimental results of ATEN, MIBNI and Best-Fit for the networks with 100 nodes inferred from noisy data. The maximum in-degree of each target gene inferred by ATEN, MIBNI and Best-fit was limited to 5. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.

Number of Time Points	Algorithm	Noise 1%			Noise 5%		
		Recall	FPR	F-score	Recall	FPR	F-score
50	ATEN	0.899 ±0.010	0.007±0.001	0.734±0.017	0.813±0.018	0.015±0.001	0.542±0.022
	MIBNI	0.742	0.041	0.303	0.727	0.041	0.293
	Best-Fit	0.750	0.019	0.469	0.625	0.025	0.349
100	ATEN	0.922 ±0.012	0.005±0.001	0.801±0.019	0.874±0.016	0.014±0.001	0.595±0.013
	MIBNI	0.750	0.041	0.306	0.750	0.041	0.306
	Best-Fit	0.891	0.026	0.454	0.836	0.027	0.424
150	ATEN	0.905±0.006	0.006±0.001	0.785±0.016	0.890±0.013	0.011 ±0.001	0.660±0.011
	MIBNI	0.742	0.041	0.303	0.719	0.041	0.293
	Best-Fit	0.891	0.025	0.463	0.875	0.027	0.439
200	ATEN	0.891±0.012	0.006±0.001	0.753±0.019	0.883±0.021	0.012±0.001	0.637±0.011
	MIBNI	0.742	0.041	0.303	0.727	0.041	0.293
	Best-Fit	0.891	0.026	0.459	0.844	0.028	0.419

Table A.5: Experimental results of ATEN, MIBNI and Best-Fit for the artificial networks with 150 nodes inferred from the noise-free data. The maximum in-degree of each target gene inferred by ATEN, MIBNI and Best-fit was limited to 5. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.

Number of Time Points	Algorithm	Noise 0%		
		Recall	FPR	F-score
50	ATEN	0.920±0.022	0.002 ±0.000	0.842±0.020
	MIBNI	0.508	0.018	0.268
	Best-Fit	0.939	0.000	0.941
100	ATEN	0.963±0.009	0.001 ±0.000	0.904±0.001
	MIBNI	0.547	0.018	0.289
	Best-Fit	0.989	0.000	0.989
150	ATEN	0.972±0.008	0.001±0.000	0.911±0.013
	MIBNI	0.547	0.018	0.289
	Best-Fit	0.994	0.000	0.994
200	ATEN	0.976 ±0.004	0.001±0.000	0.914±0.011
	MIBNI	0.5536	0.018	0.292
	Best-Fit	0.994	0.000	0.994

Table A.6: ATEN inferred more accurate artificial network topology from short and noisy data. Experimental results of ATEN, MIBNI and Best-Fit for the networks with 150 nodes inferred from noisy data. The maximum in-degree of each target gene inferred by ATEN, MIBNI and Best-fit was limited to 5. The results of ATEN are presented using average and standard deviation values. The best result among three algorithms is highlighted in boldface.

Number of Time Points	Algorithm	Noise 1%			Noise 5%		
		Recall	FPR	F-score	Recall	FPR	F-score
50	ATEN	0.875±0.011	0.004±0.000	0.745±0.011	0.806±0.013	0.011±0.000	0.498±0.007
	MIBNI	0.525	0.018	0.277	0.497	0.018	0.262
	Best-Fit	0.882	0.008	0.623	0.709	0.019	0.346
100	ATEN	0.951±0.009	0.004±0.000	0.766±0.016	0.898±0.017	0.008±0.000	0.610±0.012
	MIBNI	0.536	0.018	0.283	0.536	0.018	0.283
	Best-Fit	0.916	0.016	0.475	0.866	0.026	0.336
150	ATEN	0.960±0.009	0.006±0.000	0.719±0.010	0.925±0.009	0.010±0.000	0.562±0.010
	MIBNI	0.547	0.018	0.289	0.536	0.018	0.283
	Best-Fit	0.922	0.021	0.403	0.894	0.026	0.344
200	ATEN	0.953±0.004	0.006±0.000	0.686±0.012	0.925±0.007	0.012±0.001	0.543±0.011
	MIBNI	0.553	0.018	0.292	0.542	0.018	0.286
	Best-Fit	0.883	0.024	0.364	0.905	0.026	0.349

Table A.7: Experimental results of ATEN, MIBNI and Best-Fit for the real-world networks inferred from the noise-free data. The maximum in-degree of each target gene inferred by ATEN, MIBNI and Best-fit was limited to 4. The results of ATEN are presented using average and standard deviation values. The experimental results of REACT, i.e., the best Recall and FPR values which are presented outside brackets and the worst Recall and FPR values which are showed in brackets, were collected from Vera-Licona et al., 2014. The best result among three algorithms is highlighted in boldface. The symbol ‘*’ indicates that the corresponding results cannot be collected.

Number of Time Points	Algorithm	Noise 0%		
		Recall	FPR	F-score
50	ATEN	0.622±0.015	0.013±0.001	0.646±0.017
	MIBNI	0.382	0.054	0.286
	Best-Fit	0.673	0.008	0.721
	REACT	*	*	*
100	ATEN	0.806 ±0.013	0.006±0.001	0.823±0.014
	MIBNI	0.465	0.050	0.349
	Best-Fit	0.917	0.002	0.930
	REACT	*	*	*
150	ATEN	0.830±0.016	0.005±0.001	0.845±0.018
	MIBNI	0.444	0.051	0.333
	Best-Fit	0.903	0.003	0.910
	REACT	*	*	*
202	ATEN	0.801±0.024	0.008±0.001	0.807±0.020
	MIBNI	0.361	0.054	0.271
	Best-Fit	0.812	0.005	0.839
	REACT	0.778(0.61)	0.0824(0.2)	*

Table A.8: ATEN inferred more accurate real-world network topology from short and noisy data. Experimental results of ATEN, MIBNI, Best-Fit and REACT for the inferred network from noisy data. The maximum in-degree of each target gene inferred by ATEN, MIBNI and Best-fit was limited to 4. The results of ATEN are presented using average and standard deviation values. The experimental results of REACT, i.e., the best Recall and FPR values which are presented outside brackets and the worst Recall and FPR values which are showed in brackets, were collected from Vera-Licona et al., 2014. The symbol “*” indicates that the corresponding results cannot be collected.

Number of Time Points	Algorithm	Noise 1%			Noise 5%		
		Recall	FPR	F-score	Recall	FPR	F-score
50	ATEN	0.556±0.012	0.015±0.001	0.576±0.014	0.500±0.022	0.021±0.002	0.499±0.018
	MIBNI	0.396	0.053	0.297	0.354	0.055	0.266
	Best-Fit	0.549	0.024	0.516	0.410	0.047	0.322
	REACT	*	*	*	*	*	*
100	ATEN	0.736±0.020	0.012±0.001	0.731±0.018	0.604±0.020	0.019±0.001	0.586±0.021
	MIBNI	0.486	0.050	0.365	0.451	0.051	0.339
	Best-Fit	0.757	0.031	0.604	0.604	0.044	0.454
	REACT	*	*	*	*	*	*
150	ATEN	0.778±0.013	0.010±0.001	0.767±0.016	0.653±0.025	0.021±0.001	0.604±0.024
	MIBNI	0.431	0.052	0.323	0.410	0.052	0.307
	Best-Fit	0.826	0.027	0.670	0.667	0.041	0.501
	REACT	*	*	*	*	*	*
202	ATEN	0.765±0.017	0.008±0.001	0.742±0.014	0.563 ±0.015	0.020±0.001	0.547±0.017
	MIBNI	0.431	0.052	0.323	0.306	0.057	0.229
	Best-Fit	0.847	0.030	0.661	0.625	0.043	0.469
	REACT	0.916(0.61)	0.00824(0.08)	*	0.833(0.61)	0.0896(0.38)	*

Table A.9: ATEN inferred more accurate real-world dynamic model from short and noisy data. The maximum in-degree of each target gene inferred by ATEN, MIBNI and Best-fit was limited to 4. The numbers indicate the number of steady states that was successfully retrieved. The best result among three algorithms is highlighted in boldface. The experimental results of REACT was collected from Vera-Licona et al., 2014 and the symbol ‘*’ indicates that the corresponding results cannot be collected.

Number of Time Points	50			100			150			202		
Algorithms	Noise 0%	1%	5%	Noise 0%	1%	5%	Noise 0%	1%	5%	Noise 0%	1%	5%
ATEN	4	4	2	3	5	5	3	5	4	4	5	4
MIBNI	0	0	0	0	0	0	0	0	0	0	0	0
Best-Fit	4	1	0	5	4	2	5	5	2	5	3	1
REACT	*	*	*	*	*	*	*	*	*	5	3	5

References

- Aalto, Atte et al. (2020). “Gene regulatory network inference from sparsely sampled noisy data”. In: *Nature communications* 11.1, pp. 1–9.
- Abdi, Ali, Mehdi Baradaran Tahoori, and Effat S Emamian (2008). “Fault diagnosis engineering of digital circuits can identify vulnerable molecules in complex cellular pathways”. In: *Science Signaling* 1.42, ra10–ra10.
- Akutsu, Tatsuya, Morihiro Hayashida, Wai-Ki Ching, et al. (2007). “Control of Boolean networks: Hardness results and algorithms for tree structured networks”. In: *Journal of theoretical biology* 244.4, pp. 670–679.
- Akutsu, Tatsuya, Morihiro Hayashida, and Takeyuki Tamura (2008). “Algorithms for inference, analysis and control of Boolean networks”. In: *International Conference on Algebraic Biology*. Springer, pp. 1–15.
- Akutsu, Tatsuya, Satoru Kuhara, et al. (1998). “Identification of gene regulatory networks by strategic gene disruptions and gene overexpressions”. In: *SODA*. Vol. 98. Citeseer, pp. 695–702.
- Akutsu, Tatsuya, Satoru Miyano, and Satoru Kuhara (1999). “Identification of genetic networks from a small number of gene expression patterns under the Boolean network model”. In: *Biocomputing’99*. World Scientific, pp. 17–28.
- Albert, Réka (2004). “Boolean modeling of genetic regulatory networks”. In: *Complex networks*. Springer, pp. 459–481.

- Albert, Reka (2005). “Scale-free networks in cell biology”. In: *Journal of cell science* 118.21, pp. 4947–4957.
- Albert, Réka and Hans G Othmer (2003). “The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in *Drosophila melanogaster*”. In: *Journal of theoretical biology* 223.1, pp. 1–18.
- Arnone, Maria I and Eric H Davidson (1997). “The hardwiring of development: organization and function of genomic regulatory systems”. In: *Development* 124.10, pp. 1851–1864.
- Bansal, Mukesh, Giusy Della Gatta, and Diego Di Bernardo (2006). “Inference of gene regulatory networks and compound mode of action from time course gene expression profiles”. In: *Bioinformatics* 22.7, pp. 815–822.
- Bar-Joseph, Ziv (2004). “Analyzing time series gene expression data”. In: *Bioinformatics* 20.16, pp. 2493–2503.
- Barabasi, Albert-Laszlo and Zoltan N Oltvai (2004). “Network biology: understanding the cell’s functional organization”. In: *Nature reviews genetics* 5.2, pp. 101–113.
- Barabási, Albert-László, Natali Gulbahce, and Joseph Loscalzo (2011). “Network medicine: a network-based approach to human disease”. In: *Nature reviews genetics* 12.1, pp. 56–68.
- Barman, Shohag and Yung-Keun Kwon (2017). “A novel mutual information-based Boolean network inference method from time-series gene expression data”. In: *PloS one* 12.2, e0171097.
- (2018). “A Boolean network inference from time-series gene expression data using a genetic algorithm”. In: *Bioinformatics* 34.17, pp. i927–i933.
- Battiti, Roberto (1994). “Using mutual information for selecting features in supervised neural net learning”. In: *IEEE Transactions on neural networks* 5.4, pp. 537–550.
- Baudin, Alexis et al. (2019). “Controlling large Boolean networks with single-step perturbations”. In: *Bioinformatics* 35.14, pp. i558–i567.
- Blais, Alexandre and Brian David Dynlacht (2005). “Constructing transcriptional regulatory networks”. In: *Genes & development* 19.13, pp. 1499–1511.

- Bloomingdale, Peter, Jin Niu, Donald E Mager, et al. (2018). “Boolean network modeling in systems pharmacology”. In: *Journal of pharmacokinetics and pharmacodynamics* 45.1, pp. 159–180.
- Brazma, Alvis et al. (2003). “ArrayExpress—a public repository for microarray gene expression data at the EBI”. In: *Nucleic acids research* 31.1, pp. 68–71.
- Breiman, Leo (2001). “Random forests”. In: *Machine learning* 45.1, pp. 5–32.
- Butte, Atul J and Isaac S Kohane (1999). “Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements”. In: *Biocomputing 2000*. World Scientific, pp. 418–429.
- Chai, Lian En et al. (2014). “A review on the computational approaches for gene regulatory network construction”. In: *Computers in biology and medicine* 48, pp. 55–65.
- Chou, Hsinghua, G Premkumar, and Chao-Hsien Chu (2001). “Genetic algorithms for communications network design-an empirical study of the factors that influence performance”. In: *IEEE Transactions on Evolutionary Computation* 5.3, pp. 236–249.
- Datta, Aniruddha et al. (2003). “External control in Markovian genetic regulatory networks”. In: *Machine learning* 52.1-2, pp. 169–191.
- De Jong, Hidde (2002). “Modeling and simulation of genetic regulatory systems: a literature review”. In: *Journal of computational biology* 9.1, pp. 67–103.
- Dorier, Julien et al. (2016). “Boolean regulatory network reconstruction using literature based knowledge with a genetic algorithm optimization method”. In: *BMC bioinformatics* 17.1, p. 410.
- Edgar, Ron, Michael Domrachev, and Alex E Lash (2002). “Gene Expression Omnibus: NCBI gene expression and hybridization array data repository”. In: *Nucleic acids research* 30.1, pp. 207–210.
- Emmert-Streib, Frank, Matthias Dehmer, and Benjamin Haibe-Kains (2014). “Gene regulatory networks and their applications: understanding biological and medical problems in terms of networks”. In: *Frontiers in cell and developmental biology* 2, p. 38.

- Fogel, Gary B and David W Corne (2002). *Evolutionary computation in bioinformatics*. Elsevier.
- Ford Jr, Lester Randolph and Delbert Ray Fulkerson (2015). *Flows in networks*. Princeton university press.
- Friedman, Nir et al. (2000). “Using Bayesian networks to analyze expression data”. In: *Journal of computational biology* 7.3-4, pp. 601–620.
- Fullmer, Brad and Risto Miikkulainen (1992). “Using marker-based genetic encoding of neural networks to evolve finite-state behaviour”. In: *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pp. 255–262.
- Gao, Bo, Lixiang Li, et al. (2013). “Principle for performing attractor transits with single control in Boolean networks”. In: *Physical Review E* 88.6, p. 062706.
- Gao, Jianxi, Yang-Yu Liu, et al. (2014). “Target control of complex networks”. In: *Nature communications* 5.1, pp. 1–8.
- Garg, Abhishek et al. (2008). “Synchronous versus asynchronous modeling of gene regulatory networks”. In: *Bioinformatics* 24.17, pp. 1917–1925.
- Gates, Alexander J and Luis M Rocha (2016). “Control of complex networks requires both structure and dynamics”. In: *Scientific reports* 6.1, pp. 1–11.
- Genuer, Robin, Jean-Michel Poggi, and Christine Tuleau-Malot (2010). “Variable selection using random forests”. In: *Pattern recognition letters* 31.14, pp. 2225–2236.
- Geurts, Pierre, Damien Ernst, and Louis Wehenkel (2006). “Extremely randomized trees”. In: *Machine learning* 63.1, pp. 3–42.
- Giacomantonio, Clare E and Geoffrey J Goodhill (2010). “A Boolean model of the gene regulatory network underlying Mammalian cortical area development”. In: *PLoS Comput Biol* 6.9, e1000936.
- Günther, Frauke, Nina Wawro, and Karin Bammann (2009). “Neural networks for modeling gene-gene interactions in association studies”. In: *BMC genetics* 10.1, p. 87.
- Hartwell, Leland H et al. (1999). “From molecular to modular cell biology”. In: *Nature* 402.6761, pp. C47–C52.

-
- Hecker, Michael et al. (2009). “Gene regulatory network inference: data integration in dynamic models—a review”. In: *Biosystems* 96.1, pp. 86–103.
- Hopcroft, John E and Richard M Karp (1973). “An $n^5/2$ algorithm for maximum matchings in bipartite graphs”. In: *SIAM Journal on computing* 2.4, pp. 225–231.
- Hou, Wenpin, Peiying Ruan, et al. (2018). “On the Number of Driver Nodes for Controlling a Boolean Network to Attractors”. In: *bioRxiv*, p. 395442.
- Hou, Wenpin, Takeyuki Tamura, et al. (2016). “Finding and analyzing the minimum set of driver nodes in control of boolean networks”. In: *Advances in Complex Systems* 19.03, p. 1650006.
- Huang, Sui (1999). “Gene expression profiling, genetic networks, and cellular states: an integrating concept for tumorigenesis and drug discovery”. In: *Journal of molecular medicine* 77.6, pp. 469–480.
- Hutt, Benjamin and Kevin Warwick (2007). “Synapsing variable-length crossover: Meaningful crossover for variable-length genomes”. In: *IEEE transactions on evolutionary computation* 11.1, pp. 118–131.
- Irrthum, Alexandre, Louis Wehenkel, Pierre Geurts, et al. (2010). “Inferring regulatory networks from expression data using tree-based methods”. In: *PloS one* 5.9, e12776.
- Karlebach, Guy and Ron Shamir (2008). “Modelling and analysis of gene regulatory networks”. In: *Nature Reviews Molecular Cell Biology* 9.10, pp. 770–780.
- Kauffman, Stuart A (1969). “Metabolic stability and epigenesis in randomly constructed genetic nets”. In: *Journal of theoretical biology* 22.3, pp. 437–467.
- Kim, Junil, Sang-Min Park, and Kwang-Hyun Cho (2013). “Discovery of a kernel for controlling biomolecular regulatory networks”. In: *Scientific reports* 3, p. 2223.
- Krumsiek, Jan et al. (2011). “Hierarchical differentiation of myeloid progenitors is encoded in the transcription factor network”. In: *PloS one* 6.8, e22649.
- Lähdesmäki, Harri, Ilya Shmulevich, and Olli Yli-Harja (2003). “On learning gene regulatory networks under the Boolean network model”. In: *Machine learning* 52.1-2, pp. 147–167.

- Langmead, Christopher James and Sumit Kumar Jha (2009). “Symbolic approaches for finding control strategies in Boolean networks”. In: *Journal of Bioinformatics and Computational Biology* 7.02, pp. 323–338.
- Laubenbacher, Reinhard and Brandilyn Stigler (2004). “A computational algebra approach to the reverse engineering of gene regulatory networks”. In: *Journal of theoretical biology* 229.4, pp. 523–537.
- Lee, Wei-Po and Wen-Shyong Tzou (2009). “Computational methods for discovering gene networks from expression data”. In: *Briefings in bioinformatics* 10.4, pp. 408–423.
- Li, Fangting et al. (2004). “The yeast cell-cycle network is robustly designed”. In: *Proceedings of the National Academy of Sciences* 101.14, pp. 4781–4786.
- Liang, Shoudan, Stefanie Fuhrman, Roland Somogyi, et al. (1998). “Reveal, a general reverse engineering algorithm for inference of genetic network architectures”. In: *Pacific symposium on biocomputing*. Vol. 3. 3, pp. 18–29.
- Lim, Chee Yee et al. (2016). “BTR: training asynchronous Boolean models using single-cell expression data”. In: *BMC bioinformatics* 17.1, p. 355.
- Lin, Ching-Tai (1974). “Structural controllability”. In: *IEEE Transactions on Automatic Control* 19.3, pp. 201–208.
- Liu, Yang-Yu, Jean-Jacques Slotine, and Albert-László Barabási (2011). “Controllability of complex networks”. In: *nature* 473.7346, pp. 167–173.
- Loh, Wei-Yin (2011). “Classification and regression trees”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1.1, pp. 14–23.
- Margolin, Adam A et al. (2006). “ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context”. In: *BMC bioinformatics*. Vol. 7. S1. Springer, S7.
- Marinescu, Radu and Rina Dechter (2009). “AND/OR branch-and-bound search for combinatorial optimization in graphical models”. In: *Artificial Intelligence* 173.16-17, pp. 1457–1491.

-
- Maucher, Markus et al. (2011). “Inferring Boolean network structure via correlation”. In: *Bioinformatics* 27.11, pp. 1529–1536.
- McCluskey, Edward J (1956). “Minimization of Boolean functions”. In: *The Bell System Technical Journal* 35.6, pp. 1417–1444.
- Mendoza, Mariana Recamonde and Ana Lúcia C Bazzan (2011). “Evolving random boolean networks with genetic algorithms for regulatory networks reconstruction”. In: *Proceedings of the 13th annual Conference on Genetic and Evolutionary Computation*, pp. 291–298.
- Meyer, Patrick E et al. (2007). “Information-theoretic inference of large transcriptional regulatory networks”. In: *EURASIP journal on bioinformatics and systems biology* 2007, pp. 1–9.
- Müssel, Christoph, Martin Hopfensitz, and Hans A Kestler (2010). “BoolNet—an R package for generation, reconstruction and analysis of Boolean networks”. In: *Bioinformatics* 26.10, pp. 1378–1380.
- Nacher, Jose C and Tatsuya Akutsu (2012). “Dominating scale-free networks with variable scaling exponent: heterogeneous networks are not difficult to control”. In: *New Journal of Physics* 14.7, p. 073005.
- Nacher, Jose C, Masayuki Ishitsuka, et al. (2019). “Finding and analysing the minimum set of driver nodes required to control multilayer networks”. In: *Scientific reports* 9.1, pp. 1–12.
- Nam, Dougu, Seunghyun Seo, and Sangsoo Kim (2006). “An efficient top-down search algorithm for learning boolean networks of gene expression”. In: *Machine Learning* 65.1, pp. 229–245.
- Pal, Sankar K, Sanghamitra Bandyopadhyay, and Shubhra Sankar Ray (2006). “Evolutionary computation in bioinformatics: A review”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 36.5, pp. 601–615.
- Prim, Robert Clay (1957). “Shortest connection networks and some generalizations”. In: *The Bell System Technical Journal* 36.6, pp. 1389–1401.

-
- Quine, Willard V (1955). “A way to simplify truth functions”. In: *The American mathematical monthly* 62.9, pp. 627–631.
- Rafimanzelat, Mohammad Reza and Fariba Bahrami (2018a). “Attractor controllability of Boolean networks by flipping a subset of their nodes”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 28.4, p. 043120.
- (2018b). “Attractor stabilizability of Boolean networks with application to biomolecular regulatory networks”. In: *IEEE Transactions on Control of Network Systems* 6.1, pp. 72–81.
- Raidl, Günther R and Bryant A Julstrom (2003). “Edge sets: an effective evolutionary coding of spanning trees”. In: *IEEE Transactions on evolutionary computation* 7.3, pp. 225–239.
- Recanatini, Maurizio and Chiara Cabrelle (2020). “Drug research meets network science: where are we?” In: *Journal of Medicinal Chemistry*.
- Ruczinski, Ingo, Charles Kooperberg, and Michael LeBlanc (2003). “Logic regression”. In: *Journal of Computational and graphical Statistics* 12.3, pp. 475–511.
- Saadatpour, Assieh, István Albert, and Réka Albert (2010). “Attractor analysis of asynchronous Boolean models of signal transduction networks”. In: *Journal of theoretical biology* 266.4, pp. 641–656.
- Sahin, Özgür et al. (2009). “Modeling ERBB receptor-regulated G1/S transition to find novel targets for de novo trastuzumab resistance”. In: *BMC systems biology* 3.1, p. 1.
- Sanchez-Castillo, Manuel et al. (2018). “A Bayesian framework for the inference of gene regulatory networks from time and pseudo-time series data”. In: *Bioinformatics* 34.6, pp. 964–970.
- Schwender, Holger and Katja Ickstadt (2008). “Identification of SNP interactions using logic regression”. In: *Biostatistics* 9.1, pp. 187–198.
- Shi, Ning et al. (2020). “ATEN: And/Or tree ensemble for inferring accurate Boolean network topology and dynamics”. In: *Bioinformatics* 36.2, pp. 578–585.

- Shmulevich, Ilya, Edward R Dougherty, Seungchan Kim, et al. (2002). “Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks”. In: *Bioinformatics* 18.2, pp. 261–274.
- Shmulevich, Ilya, Edward R Dougherty, and Wei Zhang (2002). “From Boolean to probabilistic Boolean networks as models of genetic regulatory networks”. In: *Proceedings of the IEEE* 90.11, pp. 1778–1792.
- Studham, Matthew E et al. (2014). “Functional association networks as priors for gene regulatory network inference”. In: *Bioinformatics* 30.12, pp. i130–i138.
- Van Laarhoven, Peter JM and Emile HL Aarts (1987). “Simulated annealing”. In: *Simulated annealing: Theory and applications*. Springer, pp. 7–15.
- Vera-Licona, Paola et al. (2014). “An algebra-based method for inferring gene regulatory networks”. In: *BMC systems biology* 8.1, p. 37.
- Wang, Guanyu et al. (2010). “Process-based network decomposition reveals backbone motif structure”. In: *Proceedings of the National Academy of Sciences* 107.23, pp. 10478–10483.
- Wu, Lin et al. (2016). “Minimum steering node set of complex networks and its applications to biomolecular networks”. In: *IET systems biology* 10.3, pp. 116–123.
- Xiao, Yufei and Edward R Dougherty (2007). “The impact of function perturbations in Boolean networks”. In: *Bioinformatics* 23.10, pp. 1265–1273.
- Yin, Hongli and Siying Zhang (2016). “Minimum structural controllability problems of complex networks”. In: *Physica A: Statistical Mechanics and its Applications* 443, pp. 467–476.