



**WICHITA STATE  
UNIVERSITY**

**UNIVERSITY LIBRARIES**

## **Application of graphical models to inference and analysis of biological networks**

Item Type	Dissertation
Authors	Kotiang, Stephen
Publisher	Wichita State University
Rights	© Copyright 2022 by Stephen Kotiang All Rights Reserved
Download date	2026-05-20 12:40:58
Link to Item	<a href="https://soar.wichita.edu/handle/10057/24977">https://soar.wichita.edu/handle/10057/24977</a>

APPLICATION OF GRAPHICAL MODELS TO INFERENCE AND ANALYSIS OF  
BIOLOGICAL NETWORKS

A Dissertation by

Stephen Kotiang

Master of Engineering, Chonbuk National University, South Korea, 2015

Bachelor of Engineering, Moi University, Kenya, 2010

Submitted to the Department of Electrical and Computer Engineering  
and the faculty of the Graduate School of  
Wichita State University  
in partial fulfillment of  
the requirements for the degree of  
Doctor of Philosophy

December 2022

© Copyright 2022 by Stephen Kotiang  
All Rights Reserved

APPLICATION OF GRAPHICAL MODELS TO INFERENCE AND ANALYSIS OF  
BIOLOGICAL NETWORKS

The following faculty members have examined the final copy of this dissertation for form and content, and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Electrical Engineering and Computer Science.

---

Ali Eslami, Committee Chair

---

Hyuck Kwon, Committee Member

---

Yanwu Ding, Committee Member

---

KC Dukka, Committee Member

---

Bin Shuai, Committee Member

Accepted for the College of Engineering

---

Anthony Muscat, Dean

Accepted for the Graduate School

---

Coleen Pugh, Dean

## DEDICATION

To God, for the gift of life and salvation,  
and  
to my wife Venny, and daughter Kaitlyn who continue to inspire and encourage me in love  
and joyful enthusiasm,  
and  
to my mother and father Anna & Samwel Kotiang; without their love, wisdom, and  
guidance I would not be here today.

## ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor Professor Ali Eslami for his constant guidance and technical insight throughout my Ph.D. studies. His patience, encouragement, and especially his confidence in me were invaluable as he pushed me to perform to the best of my abilities. It has been an honor working under his supervision. Also, I thank my dissertation committee, Professor Kwon, Professor Ding, Professor Shuai, and Professor Dukka for their valuable feedback and insights that have influenced the quality of this research. In addition, I want to thank Ms. Kristie Bixby for her extensive and intensive editorial assistance throughout my research work. I extend my gratitude and best wishes to the past and current members of the Information Systems Laboratory at WSU, including Dr. Ali Behfarnia and Dr. Michael Okwori for their support and friendship. Special thanks to my family and church members at Wichita Three Angels Seventh-Day Adventist Church for the many prayers and encouragement throughout this journey. Finally, I acknowledge the support of this work in part by the National Aeronautics and Space Administration (NASA) under Award 80NSSC20M0133, in part by the National Science Foundation under Award OIA-1656006, and in part by the State of Kansas through the Kansas Board of Regents.

## ABSTRACT

The inference of gene regulatory networks (GRNs) from gene-expression measurements, and the desire to understand genomic functions and the behavior of these complex networks form a core element of systems biology-based phenotyping. This inference, also known as reverse engineering, is one of the most challenging tasks in systems biology and bioinformatics, mainly due to the complex nature of biological systems, which involve many factors and uncertainties. However, with rapid biotechnological advancements, large-scale high-throughput biological data have become available. These data have enabled researchers to deduce and understand how interactions among the vast array of components in biological systems relate and affect each other. Nevertheless, an up-to-date full understanding of such interactions has not been possible. In the recent past, numerous computational methodologies have been formalized to enable the deduction of reliable and testable predictions in today's biology. However, little research focus has been aimed at quantifying how well existing state-of-the-art GRNs correspond to measured gene-expression profiles. Furthermore, knowledge on how biological noise or error propagate up the development ladder of biological systems is currently lacking.

This dissertation presents computational frameworks to explore the global behavior of biological systems and the consistency between experimentally verified GRNs against corresponding gene-expression dataset. Also considered is the general question of the effect of perturbation on the dynamical network behavior, as well as developing an analytical technique to capture and characterize error evolution in biological networks. The developed computational tools are applied to assess network steady states, network state progression, and impact of gene deletion in *Escherichia coli* and *Budding yeast* models. The computational frameworks explained here provide useful graphical models and analytical tools to study biological networks. Moreover, the error propagation technique provides a step towards accurate prediction of noise propagation in biology, a key to understanding faithful signal propagation in gene networks as well as designing noise-tolerant artificial gene circuits.

# TABLE OF CONTENTS

Chapter	Page
1 INTRODUCTION . . . . .	1
1.1 Objective . . . . .	3
1.2 Thesis Organization . . . . .	3
1.3 Publications . . . . .	4
2 BACKGROUND INFORMATION . . . . .	5
2.1 Introduction . . . . .	5
2.2 Bayesian Networks . . . . .	6
2.2.1 Representation . . . . .	6
2.2.2 The Learning Problem . . . . .	7
2.2.3 Maximum Likelihood Estimator . . . . .	8
2.3 Factor Graphs and Message-Passing Algorithm . . . . .	10
2.3.1 Sum-Product Algorithm . . . . .	11
2.4 Density Evolution in Factor Graphs . . . . .	14
2.5 Discretization of Gene Expression Data . . . . .	15
3 PROBABILISTIC MODEL ANALYSIS OF BIOLOGICAL NETWORKS . . . . .	18
3.1 Introduction . . . . .	18
3.1.1 Related Work . . . . .	19
3.2 Model and Methods . . . . .	20
3.2.1 Gene Network Representation . . . . .	20
3.2.2 Inference in FGN Model . . . . .	21
3.2.3 Network Models and Data . . . . .	24
3.2.4 Discretization of Data . . . . .	26
3.3 Results and Analysis . . . . .	27
3.3.1 SOS Response Model . . . . .	28
3.3.2 Acid-Resistance Model . . . . .	34
3.3.3 Sparse and Dense Networks . . . . .	34
3.3.4 Impact of Discretization Levels . . . . .	38
3.3.5 Computational Complexity . . . . .	39
3.4 Summary . . . . .	41
4 BOOLEAN FACTOR GRAPH MODEL OF BIOLOGICAL NETWORKS . . . . .	42
4.1 Introduction . . . . .	42
4.1.1 Related Work . . . . .	44
4.2 Model and Methods . . . . .	45
4.2.1 Boolean Networks . . . . .	45
4.2.2 Boolean Factor Graph Model . . . . .	47

## TABLE OF CONTENTS (continued)

Chapter	Page
4.2.3 Model Network: Yeast Cell Cycle . . . . .	51
4.3 Applications of Boolean Factor Graph Model . . . . .	53
4.3.1 Gene-Deletion Analysis . . . . .	53
4.3.2 Gene Network Consistency Analysis . . . . .	58
4.3.3 Node Connectivity Analysis . . . . .	61
4.4 Performance Analysis . . . . .	63
4.4.1 Models Comparison . . . . .	63
4.4.2 Computation Cost . . . . .	65
4.5 Discussion . . . . .	67
4.6 Summary . . . . .	68
5 ERROR PROPAGATION ANALYSIS IN BIOLOGICAL NETWORKS . . . . .	70
5.1 Introduction . . . . .	70
5.1.1 Related Work . . . . .	71
5.2 Density Evolution Analysis on Gene Networks . . . . .	72
5.2.1 Network Model . . . . .	73
5.2.2 Error Analysis I . . . . .	74
5.2.3 Error Analysis II . . . . .	77
5.3 Discussion and Results . . . . .	81
5.3.1 Numerical Evaluations . . . . .	81
5.3.2 Computational Complexity . . . . .	85
5.3.3 Models Comparison . . . . .	86
5.3.4 Application to Hub-Like GRNs . . . . .	87
5.4 Summary . . . . .	87
6 CONCLUSION AND FUTURE WORK . . . . .	88
6.1 Conclusion . . . . .	88
6.2 Future Work . . . . .	88
6.2.1 Signal Processing on Graphs . . . . .	88
6.2.2 Error Propagation in Gene Networks . . . . .	89
REFERENCES . . . . .	91
APPENDIX . . . . .	102
A. Error Propagation Analysis . . . . .	103

## LIST OF TABLES

Table	Page
3.1	Two-logical states distribution of SOS response network's predicted marginal posteriors versus observed experimental states . . . . . 29
3.2	Three-logical states distribution of SOS response network's predicted marginal posteriors versus observed experimental states . . . . . 29
3.3	FGN model predictions on increasing number of quantization levels for SOS response network and AR regulatory network, and corresponding CPU runtimes. 39
4.1	Boolean truth tables for both activating and inhibiting gene interactions . . . . 48
4.2	Attractors of budding yeast cell cycle [23] . . . . . 53
4.3	Temporal evolution of protein states in Sic1 gene deletion . . . . . 55
4.4	Yeast cell cycle progression in Cln3 and Cln1,2 deletion . . . . . 56
4.5	Temporal evolution of protein states in Clb1,2 gene deletion . . . . . 56
4.6	Temporal evolution of protein states in SBF and MBF TFs deletion . . . . . 57
4.7	Temporal evolution of protein states in MBF gene deletion . . . . . 58
4.8	Bayes information criterion (BIC) measure of Gaussian mixture model discretization used to fit gene-expression data for different k number of components. . . . . 59
4.9	Attractors of cell cycle on real biological data . . . . . 60
5.1	Boolean truth tables for implementing logical cooperative rule of equation (5.9) 79

## LIST OF FIGURES

Figure	Page
2.1 Simple four-node Bayesian network with sample conditional probability tables. . . . .	7
2.2 Factor graph that expresses joint distribution as product of local functions $f_1(x_1)f_2(x_1, x_2, x_4)f_3(x_1, x_3)f_4(x_2, x_4)$ . . . . .	11
2.3 Factor-graph fragment depicting update rules of sum-product message-passing algorithm. . . . .	12
2.4 Message-passing in factor graph with erroneous messages, where $\phi_{(\cdot)}$ repre- sents probability of error message from variable node to factor node, and $\gamma_{(\cdot)}$ denotes probability of error message from factor node to variable node. . . . .	14
2.5 Discretization process workflow with two discrete states $A = \{0, 1\}$ for gene $x_i$ . GED $D'$ and discretized $D$ are composed by $n$ genes and four experimen- tal conditions. Discretization algorithm $\Sigma_I$ takes $D'$ and $x_i$ and infers the cut point $p = \{6\}$ and discretization scheme $I = \{[0.4, 6], (6, 9.5]\}$ to obtain dis- cretized expression profile of $x_i$ in the GED $D$ . Discretized GED $D$ is the in- put of an inference algorithm that will extract some kind of information of interest. . . . .	17
3.1 Graphical probability models: (a) Bayesian gene network, and (b) factor graph equivalent. . . . .	20
3.2 Messages sent in belief update of factor graph network example shown in Figure 3.1(b). Arrows indicate flow of belief messages sent between variable nodes (i.e., dashed ellipses corresponding to nodes in Bayesian gene network), $\mu$ denotes message sent from parent gene to child gene in FGN model, and conversely, $\lambda$ represents message from child to parent. . . . .	23
3.3 Graphical representation of <i>E. coli</i> SOS DNA repair true pathway [17]. . . . .	25
3.4 Graphical representation of acid-resistant gene regulatory network [49]. Over- expressed EvgA response regulator indirectly activates acid resistance through transcription of <i>ydeO</i> (i.e., $\text{EvgA} \rightarrow \text{YdeO} \rightarrow \text{GadE} \rightarrow \text{AR}$ ). However, with- out overexpression (under normal inducing conditions), EvgA can directly activate <i>gadE</i> transcription. . . . .	26
3.5 Pearson correlation plots between proportions of observed states and FGN in- ferred marginal posteriors for SOS response network: (a) 2-states discretiza- tion, p-value = $1.6858 \times 10^{-14}$ , and (b) 3-states discretization, p-value = $1.7540 \times 10^{-15}$ . Correlation coefficient $\rho$ is given in each plot. . . . .	31

LIST OF FIGURES (continued)

Figure	Page	
3.6	Average Pearson correlation coefficient plots of $n$ -randomized nodes or edges for SOS response network. Model predictions are repeatedly evaluated, and averages of correlation coefficients computed in $10 \cdot  E $ random attempts: (a) randomly shuffled datasets where states are swapped between different variable nodes, and (b) randomly shuffled $n$ -edges over network structure. . . . .	32
3.7	Pearson correlation plots between proportions of gene experimental observations and FGN inferred beliefs in <i>recA</i> gene KO model: (a) 2-states discretization, and (b) 3-states discretization. Correlation coefficient $\rho$ is given in each plot. . . . .	33
3.8	Pearson correlation plots between observed states node proportions and FGN inferred marginal posteriors for AR response network: (a) 2-states discretization, p-value = $8.5579 \times 10^{-13}$ , and (b) 3-states discretization, p-value = $2.920 \times 10^{-20}$ . . . . .	35
3.9	Average Pearson correlation coefficient plots of $n$ -randomized nodes or edges for AR response network. Model predictions are repeatedly evaluated, and averages of correlation coefficients are computed in $10 \cdot  E $ random attempts: (a) randomly shuffled datasets where states are swapped between different variable nodes, and (b) randomly shuffled $n$ -edges over network structure. . . . .	36
3.10	Evaluation of probabilistic FGN model on GRNs with different sparsity in: (a) SOS response network, and (b) AR regulatory network. Performance is compared to very sparse and dense networks for both 2-states (yellow-fill blue boxplots) and 3-states (red boxplots) discretization levels. Sparse and dense networks are obtained by deletion of edges from and addition of edges to true (original) GRNs, respectively. . . . .	37
3.11	Pearson correlation plots of LBP message-passing convergence with increasing iteration for both 2- and 3-states discretization levels in: (a) SOS response network, and (b) AR regulatory network. CPU runtime until convergence is on average 0.26s and 0.43s in SOS network for 2- and 3-states discretization, respectively. Accordingly, CPU runtime for AR network is on average 0.15s and 0.26s. . . . .	40
4.1	(a) Simple directed gene graph with $n = 4$ . (b) Equivalent undirected factor graph representation of (a). The state of gene $x_i$ at time $t + 1$ is governed by Boolean function $f_i$ , given a set of interacting genes. For instance, $x_3(t + 1) = f_3(x_2(t), x_3(t), x_4(t))$ . Red blunt edges indicate inhibition links, whereas black arrowheads represent activation interactions. . . . .	46

LIST OF FIGURES (continued)

Figure	Page
4.2	Message passing update in a sample Boolean factor graph: (a) messages sent from variable nodes to factor nodes denoted by $\lambda$ values, and (b) messages sent from factor nodes to variable nodes given by $\mu$ values. Solid (dashed) edges denote activation (inhibition) interactions. . . . . 50
4.3	Simplified yeast cell-cycle network adapted from [23]. Solid green edges denote activation links, whereas dashed red edges represent inhibition links. Nodes Cln3, Cln1,2, Swi5, Cdc20,14, and Mcm1,SFF have self degradation. Cln3 is a “stater kinase” used to trigger the G <sub>1</sub> - S phase transition when the cell grows sufficiently large. . . . . 51
4.4	Boxplots of the number of attractors of random biological networks with increasing average connectivity. Each network has ten nodes with irregular factor node degree distribution, and $\lfloor j \rfloor$ denotes the largest integer that is less than or equal to $j$ . The median of the boxplots follows a power-law distribution. 62
4.5	Time to evaluate the global attractors in random Boolean networks of ten nodes with respect to: (a) average connectivity of factor nodes, and (b) total number of edges in the network. The running cost depicts a linear performance with the graph connectivity. . . . . 66
5.1	(a) Simple directed gene graph, and (b) equivalent undirected factor graph representation of (a). Red blunt edges in (a) and black dashed edges in (b) indicate inhibition links, whereas black arrowheads represent activation interactions. . . . . 74
5.2	Message passing in Boolean factor graph with state perturbation probability $\epsilon$ . Gene $x_1$ ( $x_3$ ) activates (inhibits) gene $x_2$ . $\beta$ is an event that a gene is perturbed. The dashed edge denotes inhibition interaction. Messages $\lambda$ are passed between variable nodes and factor nodes. . . . . 75
5.3	Factor graph with initial disturbance error probability $\epsilon$ . Genes $x_1$ and $x_2$ activate gene $x$ , whereas genes $x_3$ and $x_4$ inhibit gene $x$ . Dashed edges denote inhibition interactions. . . . . 78
5.4	Error probability under density evolution for networks with different $\rho(x)$ as $l \rightarrow \infty$ . Initial $\epsilon_0 = 0.25$ . . . . . 82
5.5	Propagated error in biological network with $\beta_{u,v} = 0.6651$ in different iterations w.r.t. the initial disturbance. As $l \rightarrow \infty$ , the evolved error vanishes in $\beta_{u,v} < 1$ . . . . . 83

LIST OF FIGURES (continued)

Figure	Page
5.6	84

Probability of error for networks with different polynomial degree distributions, i.e.,  $\beta_{u,v}$  values (shown in the plot) versus the number of iterations,  $l \rightarrow \infty$ . Initial disturbance  $\epsilon_0 = 0.25$ . Networks  $\rho(u, v) = 0.9u^3v + 0.1u^3v^2$  and  $\rho(u, v) = 0.4u^2v + 0.6u^3v$  both have  $\beta_{u,v} \geq 1$ . . . . .

## LIST OF ABBREVIATIONS

AR	Acid Resistant
BIC	Bayes Information Criterion
BN	Boolean Network
COPASI	Complex Pathway Simulator
CPU	Central Processing Unit
DE	Density Evolution
DAGs	Directed Acyclic Graphs
DNA	Deoxyribonucleic Acid
<i>E. coli</i>	<i>Escherichia coli</i>
EM	Expectation-Maximization
FGN	Factor Graph Network
GED	Gene Expression Data
GMM	Gaussian Mixture Model
GRN	Gene Regulatory Network
KO	Knockout
LBP	Loopy Belief Propagation
miRNA	MicroRNA
ODE	Ordinary Differential Equation
RNA	Ribonucleic Acid

# CHAPTER 1

## INTRODUCTION

Understanding how biological organisms coordinate the systems-level response of a cell is of primary concern in systems biology. The cellular response is controlled by multiple biochemical or regulatory mechanisms that can be encapsulated by large network models. These networks aim at capturing dependencies among the interacting biological entities, such as messengerRNAs, proteins, and other metabolites. Also, modeling the coupled dynamics of gene or protein expression patterns in accordance with changing internal and environmental conditions, as well as characterization of the long-run behavior of such networks, is perhaps the most important task in genomic signal processing. In the literature, long-run behavior or the distribution of network states has been conjectured to correspond to the phenotype of a cell [1]. As high-throughput multi-level and multi-scale biological datasets are becoming more available, the inference of large-scale molecular networks has become an active area of research [2, 3].

In the past decade, numerous computational methods and mathematical frameworks have been developed to infer and analyze molecular networks. These methodologies have been used to rigorously integrate prior biological knowledge and high-throughput measurements. Such methods, often referred to as reverse engineering [4, 5, 6, 7], have been used to fit discrete models of gene regulatory networks (GRNs) to high-throughput experimental biological data. In the literature, gene expression-based inference approaches have shown modest performance when applied to real data compared to *in silico* expression data [4, 8]. In addition, predictive performance over a purely microarray expression-based approach can be improved by incorporating multiple types of data such as gene set enrichment [9], sequence information [10], and network topology [11].

On the other hand, GRNs have commonly been modeled using ordinary differential equations [12, 13, 14], Boolean networks (BNs) [15, 16], and probabilistic graphical models including Bayesian networks [17, 18]. For the reconstructed GRN model reassessment in light

of additional evidence, computational methodologies have been developed and formalized mathematically in the recent past, in order to rigorously integrate prior biological knowledge and high-throughput measurements [19, 20]. Furthermore, such methodologies have been formalized in a manner that allows for good predictive descriptions of experimental data.

Nevertheless, regardless of the modeling or computational approach applied, little emphasis has been put on assessing the validity of the inferred gene networks. Given the topology of a biological network and a partial set of gene expression profiles for all genes in the network, a reverse engineering algorithm must infer a probabilistic dynamical system that best *explains* the observed experimental data. This reverse engineering problem is considered in this dissertation. The *dynamics* of a network as trajectories of gene expression levels at steady state, given experimental conditions, are described. Subsequently, I formulated a computational framework that combines the formulation of Bayesian networks and factor graphs [21] to investigate the global dynamical property and impact of gene knockout in regulatory networks of gene interactions. The probabilistic model was applied to *Escherichia coli* DNA expression data, where it is successfully used to predict the global allowable steady state of genes in the respective extracted sub-networks. The analyses were performed on real gene expression data and expertly curated networks. Then, the model was validated using network perturbation techniques [22], as well as gene deletion experiments.

Similarly, I developed a Boolean factor graph computational framework to identify relevant asymptotical behaviors such as state transition cycles and steady states. The framework and structure of the proposed model can allow us to track the progress of network states. The methodology was applied to study a sample network regulating the cell cycle of the budding yeast [23]. Furthermore, the proposed framework allows us to derive an analytic closed-form recursive equation that captures the behavior and propagation of errors introduced by random perturbations in gene networks through the “density evolution” (DE) analysis [24, 25]. For the first time, DE analysis is introduced to study perturbations in biological networks. From the simulation and analyses conducted, I deduced that error characterization is not only important in its own right but also forms a basis that allows us

to quantify network parameters in designing models for inferring gene regulatory networks from gene-expression profiles. Finally, the application of the derived recursive equation elucidates what properties of gene networks are directly responsible for their robustness. All simulations and algorithms in this dissertation were implemented in the MATLAB software and environment.

## 1.1 Objective

In this dissertation, I seek to advance the body of knowledge for developing computational frameworks, tools, and algorithms needed for the accurate inference of gene regulatory networks. Developing accurate predictive tools needed to characterize and understand dynamical systems such as gene networks, is a major concern of genomic signal processing. The outcome of this work has merit in both natural and synthetic biology applications, for example, understanding genomic functions and the behavior of gene networks for the development of drugs and gene therapies; designing artificial gene circuits for medical, biotechnological, and industrial applications.

Consequently, tools developed in this study have the potential to deliver fundamental network theories, novel algorithms/protocols, and performance analysis for dealing with cascade noise or perturbation, and devising cross-network design solutions to improve the resilience against a cascade in communication networks.

## 1.2 Thesis Organization

The rest of the dissertation is organized as follows:

- Chapter 2 outlines the background on general tools and techniques used to design and build computational tools in the rest of the chapters. A brief but necessary knowledge on Bayesian network modeling, factor graphs, and inferences algorithms employed on graphical models. Finally, the concept of density evolution is introduced as a technique to analyze error propagation in biological networks.
- In Chapter 3, I introduce the first computational framework formalized as a probabilistic graphical model to statistically assess the consistency between biological

networks and their corresponding experimental gene-expression data.

- In Chapter 4 , a factor graph model representation of biological networks is formalized based on Boolean network models. The proposed model is employed to study the dynamic behavior of gene networks and computationally predict their response to random perturbations such as gene deletion or knockout.
- Chapter 5 treats the problem of error propagation in biological networks. A recursive formula is established and derived to quantify and characterize error propagation based on the density evolution analysis technique.
- Finally, the dissertation is concluded with a summary and propose future research in Chapter 6.

### 1.3 Publications

Below is a list of publications from my research that form the chapters of this dissertation.

Journal Paper(s):

1. Kotiang, S. and A. Eslami, “Density evolution for noise propagation analysis in biological networks,” *IEEE Access*, vol. 10, pp 4261 – 4270, April 2022.
2. Kotiang, S. and A. Eslami, “Boolean factor graph model for biological systems: The yeast cell-cycle network,” *BMC Bioinformatics*, vol. 22 no. 1, pp 1–27, May 2021.
3. Kotiang, S. and A. Eslami, “A probabilistic graphical model for system-wide analysis of gene regulatory networks,” *Bioinformatics*, vol. 36 no. 10, pp 3192–3199, Dec 2020.

Conference Paper(s):

1. Kotiang, S. and A. Eslami, Probabilistic factor graph modeling and analysis of biological networks, the 28<sup>th</sup> *Intelligent Systems for Molecular Biology (ISMB 2020)* Virtual Conference, July 13–16, 2020.

## CHAPTER 2

### BACKGROUND INFORMATION

#### 2.1 Introduction

Modeling the coupled dynamics of gene or protein expression patterns in accordance with changing internal and environmental conditions is an important task in systems biology. Thus, advances in computational and modeling tools in systems biology have provided an unprecedented opportunity to identify, infer, and understand the complex interactions of biological entities. To characterize and uncover the exact dynamics of genome-wide gene regulatory networks, significant research effort has been devoted to continuously refining computational methods.

In the literature, gene expression-based inference approaches have shown modest performance when applied to real data compared to *in silico* expression data [4, 8]. In addition, predictive performance over a purely gene expression-based approach can be improved by incorporating multiple types of data such as gene set enrichment [9], sequence information [10], and network topology [11]. For the reconstructed GRN model reassessment in light of additional evidence, in the recent past, computational methodologies have been developed and formalized mathematically, in order to rigorously integrate prior biological knowledge and high-throughput measurements [19, 20]. Furthermore, such methodologies have been formalized in a manner that allows for good predictive descriptions of experimental data.

This dissertation takes advantage and exploits techniques from probabilistic methods, graphical modeling, machine learning, communication theory, and systems biology to design, develop, build, and evaluate computational tools and algorithms for the analysis of complex biological networks.

## 2.2 Bayesian Networks

A graphical model is a family of probability distributions defined in terms of a *directed* or *undirected* graph. The nodes in the graph are identified with random variables, and joint probability distributions are defined by taking products over functions defined on connected subsets of nodes. Bayesian networks are probabilistic graphical models that compactly express the joint probability distribution among a set of variables [26]. They are a form of *directed graphical models* based on directed acyclic graphs (DAGs). In addition, Bayesian networks are useful for describing complex systems composed of locally interacting entities, where an entity only interacts with a small number of other entities. As such, Bayesian networks are well studied and have been successfully employed in numerous applications. These applications include monitoring systems, speech-recognition systems, medical and fault diagnosis expert systems, systems biology, classification and analysis for biological sequencing, etc. The interest in Bayesian networks stems from the fact that they can appropriately model GRNs [17, 18], whereby genes are modeled as nodes on the graph and causal dependencies between genes are modeled as edges.

### 2.2.1 Representation

In general, a Bayesian network can be formulated by a set  $\{G, \theta\}$ , where  $G = (X, E)$  is the model structure denoting the interaction dependencies, and  $\theta$  is the model parameter. In simple terms,  $\theta$  is referred to as the conditional probability distribution, and for discrete variables,  $\theta$  constitutes the conditional probability tables.  $X$  is a finite, nonempty set whose elements are called nodes (or vertices), and  $E$  is a set of ordered pairs of *distinct* elements of  $X$ . Elements of  $E$  are called edges (or arcs), and if  $(x_i, x_j) \in E$ , we say there is an edge from  $x_i$  to  $x_j$ .

Suppose we are given a set of random variables  $X = \{x_1, x_2, \dots, x_n\}$  describing nodes on  $G$ , and let a discrete conditional probability distribution of each node given values of its parents in  $G$  be specified. Then the product of these conditional distributions yields a joint probability distribution  $p$  of variables. The joint probability distribution can be decomposed

and formulated as

$$\begin{aligned}
 p(x_1, x_2, \dots, x_n) &= p(x_1|\Pi(x_1))p(x_2|\Pi(x_2)) \cdots p(x_{n-1}|\Pi(x_{n-1}))p(x_n|\Pi(x_n)) \\
 &= \prod_{i=1}^n p(x_i|\Pi(x_i)),
 \end{aligned}
 \tag{2.1}$$

where  $\Pi(x_i)$  or  $\Pi_i$  is the set of variables on which  $x_i$  depends, that is, the parent set of  $x_i$ . In large networks, this property allows us to greatly reduce the amount of required computation, since generally, most nodes will have few parents relative to the overall size of the network. Figure 2.1 shows an example Bayesian network, where  $G$  represents the decomposition of the joint distribution  $p(x_1, x_2, x_3, x_4) = p(x_1|x_2)p(x_2|x_3x_4)p(x_3)p(x_4)$ , and  $\theta$  constitutes the conditional probability tables. In this context,  $x_i$  is a discrete variable assuming values in  $\{1, 2\}$ .

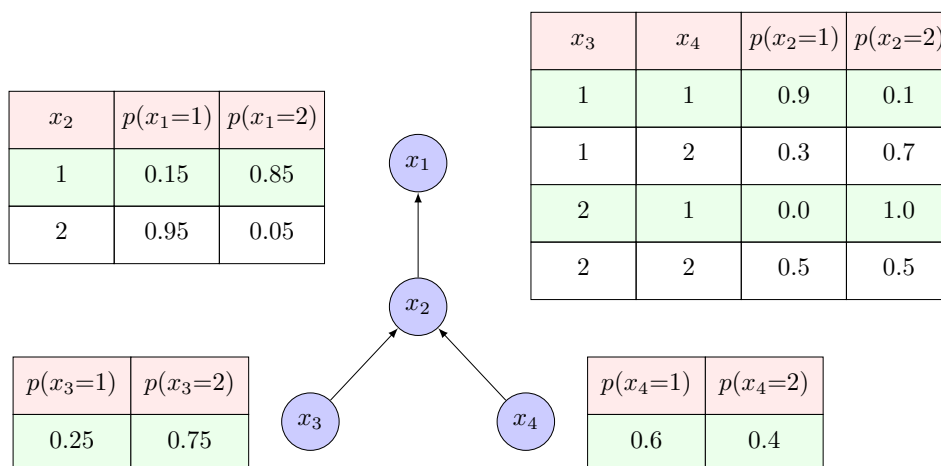


Figure 2.1: Simple four-node Bayesian network with sample conditional probability tables.

## 2.2.2 The Learning Problem

In Bayesian networks, the learning problem can be described in two ways: (1) learning the model structure  $G$ , and (2) learning the model parameter  $\theta$ . Learning  $G$  is a model

selection problem given a set of training data and prior information [27, 28]. Also, learning can be from a complete or incomplete data. In the case of incomplete data, there are instances of missing values in the dataset, whereas in the complete data case, all parameter values are known. Given a complete data training set  $D$  over  $X$ , the task is to select a model structure  $G$  that maximizes the posterior probability, i.e.,  $G = \arg \max p(G|D)$ . Following the Bayesian theorem, the posterior can be formulated as

$$p(G|D) = \frac{p(D|G)p(G)}{p(D)}, \quad (2.2)$$

where  $p(G)$  is the structure prior,  $p(D|G)$  is the marginal likelihood, and  $p(D)$  is a constant that is independent of the structure  $G$  given by

$$p(D) = \sum_{G_i} p(D|G_i)p(G_i). \quad (2.3)$$

In the literature, the difficulty of learning Bayesian networks lies in the computation of the marginal likelihood because it involves integration, which is often intractable [28]. Given a model structure, the marginal likelihood is given by

$$p(D|G) = \int_{\theta} p(D|G, \theta)p(\theta|G) d\theta. \quad (2.4)$$

Chickering [29] reported that finding the optimal directed acyclic graph with respect to best posterior probability is NP-complete. In this work the main concern was learning  $\theta$  (an estimation problem), rather than learning  $G$ . Given a sequence of instances of the variable nodes  $x_i$ 's, the interest is to find an estimate for the parameter's value that most *likely* seems to have caused this particular set of sequences. That is,  $\theta$ , which gives a sample the highest probability compared to other possible values of the parameter, namely the optimal  $\theta$  value that best explains the given sample sequences.

### 2.2.3 Maximum Likelihood Estimator

One possible way to find an estimator for a parameter is the likelihood method. Suppose  $x_i[1], x_i[2], \dots, x_i[m]$  are samples from a population with a certain distribution and

unknown parameter  $\theta$ , where  $m$  are instances of the variable  $x_i$ , and  $i = 1, 2, \dots, n$ . In the simple case, let us consider a single variable  $x$ . Thus, the likelihood function is defined as

$$L(\theta : D) = p(D|\theta) = p(x[1], x[2], \dots, x[m] | \theta) = \prod_{j=1}^m p(x[j] | \theta), \quad (2.5)$$

where  $\theta$  gets all the possible values of the desired parameter. An outcome  $\hat{\theta}$  is the maximum likelihood estimator for  $\theta$ , if it holds that for all  $\theta$ ,  $L(\hat{\theta}) \geq L(\theta)$ . That is,  $\hat{\theta}$  is the best  $\theta$  value that maximizes the likelihood function.

Given the network structure shown in Figure 2.1 and a training data sample  $D$  of the form

$$D = \begin{pmatrix} x_1[1] & x_2[1] & \dots & \dots & x_n[1] \\ x_1[2] & x_2[2] & \dots & \dots & x_n[2] \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ x_1[m] & x_2[m] & \dots & \dots & x_n[m] \end{pmatrix} = \begin{pmatrix} x_1[1] & x_2[1] & x_3[1] & x_4[1] \\ x_1[2] & x_2[2] & x_3[2] & x_4[2] \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ x_1[m] & x_2[m] & x_3[m] & x_4[m] \end{pmatrix},$$

where we assume the samples are independent and identically distributed (i.i.d), the likelihood function is computed as

$$\begin{aligned} L(\theta : D) &= \prod_m p(x_1[m], x_2[m], x_3[m], x_4[m] : \theta) \\ &= \prod_m \begin{pmatrix} p(x_1[m] | x_2[m] : \theta) \\ \times p(x_2[m] | x_3[m], x_4[m] : \theta) \\ \times p(x_3[m] : \theta) \\ \times p(x_4[m] : \theta) \end{pmatrix}. \end{aligned} \quad (2.6)$$

The second part of equation (2.6) has been rewritten as a factorial representation of the network structure in Figure 2.1. That is, looking at columns of the training data instead of rows, the computation of the likelihood is now a product of the likelihood of each variable given its parents. For any Bayesian network, we can generalize the likelihood function as

$$\begin{aligned}
L(\theta : D) &= \prod_m p(x_1[m], x_2[m], \dots, x_n[m] : \theta) \\
&= \prod_i \prod_m p(x_i[m] \mid \Pi_i[m] : \theta) \\
&= \prod_i L_i(\theta_i : D) .
\end{aligned} \tag{2.7}$$

From equation (2.7), we can observe that the likelihood decomposes according to the structure of the network. Therefore, we can compute each node's log-likelihood separately and then sum them together to obtain the log-likelihood of the entire network. This decomposition principle enables us to perform independent estimations, since the parameters for each variable are independent [26].

### 2.3 Factor Graphs and Message-Passing Algorithm

In general, a factor graph [21] for variables  $x_1, \dots, x_n$  and functions  $f_1, \dots, f_K$  can be defined to be a bipartite<sup>1</sup> graph associating a set of nodes known as *variable nodes* corresponding to the variables, and a set of *factor nodes* corresponding to the functions as shown in Figure 2.2. Each factor node depends on a subset of variable nodes, i.e., an edge exists between variable node  $x_i$  and factor node  $f_k$ , if and only if  $x_i$  is an argument of  $f_k$ . Let a real-valued  $g(X)$  be a global function over the variables in the set  $X = \{x_i : i \in N\}$ , then  $g$  can be written as

$$g(x_1, \dots, x_n) = \prod_{k \in K} f_k(x_{C_k}) , \tag{2.8}$$

where  $K$  is a discrete index set,  $C_k$  is the index set of variables that are connected to the function  $f_k$ , and  $x_{C_k}$  denotes this set of variables. Also, each factor  $f_k(x_{C_k})$  in equation (2.8) is referred to as a *local function*. Suppose a real-valued function  $g(x_1, x_2, x_3, x_4)$  is defined on the bipartite graph shown in Figure 2.2, then  $g$  can be expressed as a product

$$g(x_1, x_2, x_3, x_4) = f_1(x_1) f_2(x_1, x_2, x_4) f_3(x_1, x_3) f_4(x_2, x_4) . \tag{2.9}$$

---

<sup>1</sup>A bipartite graph (or bigraph) is a graph whose vertices can be divided into two disjoint sets  $U$  and  $V$  such that every edge connects a vertex in  $U$  to one in  $V$ .

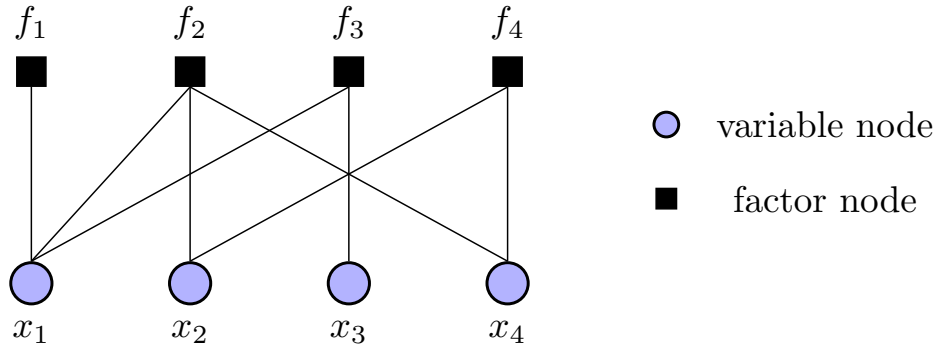


Figure 2.2: Factor graph that expresses joint distribution as product of local functions  $f_1(x_1)f_2(x_1, x_2, x_4)f_3(x_1, x_3)f_4(x_2, x_4)$ .

Factor graphs encompass many graphical models including Markov random fields [30], Bayesian networks [26], and Tanner graphs [25, 31]. It is plausible that many algorithms and mathematical models in these fields are naturally expressed in terms of factor graphs. One such algorithm is the *sum-product algorithm*, also known as *belief-propagation* [21], which operates in a factor graph by passing “messages” along edges of the graph, following a single, simple computational rule.

### 2.3.1 Sum-Product Algorithm

In many circumstances, we seek to compute the *posterior* distributions, also referred to as *marginal* functions,  $g_i(x_i)$ , for more than one value of  $i$ . For each  $i \in N$  the variable  $x_i$  takes on values in the finite set  $A_i$ . For the moment, take  $g$  as being real-valued. This section describes a general algorithm that can be used to compute the marginal functions

$$g_i(x_i) \triangleq \sum_{x_1 \in A_1, \dots, x_{i-1} \in A_{i-1}, x_{i+1} \in A_{i+1}, \dots, x_n \in A_n} g(x_1, \dots, x_n) \quad (2.10)$$

for all variables  $x_i$ . Adopting the convention that  $\sum_{x_i} f(x_i) = \sum_{x_i \in A_i} f(x_i)$ , and similarly, for a subset  $J \subset N$ , the notation  $\sum_{x_i: i \in J} f(X_J)$  means that we sum over all possible configurations of the variables indexed by  $J$ . Thus  $g_i(x_i) = \sum_{x_j: j \in N \setminus \{i\}} g(X)$ . The definition of marginal function can be extended to an arbitrary subset  $J$  of  $N$  by expressing [21]

$$g_J(X_J) \triangleq \sum_{x_i: i \in N \setminus J} g(X). \quad (2.11)$$

The computation of  $g_i(x_i)$  begins at the leaves (i.e., nodes without descendants) of a factor graph. Each leaf factor node and each leaf variable node send “belief” messages to their parent nodes. A factor node  $f$  with parent  $x$  computes the product of all received messages from its children and then operates on the result with a summary operator  $\sum_{\sim\{x\}}$  over all its variables except  $x$  to send its belief. On the other hand, a variable node  $x$  simply sends the product of all received messages as its belief. It is worth noting that there is no fixed parent/child relationship among neighboring nodes. As such, each two neighboring nodes are regarded as a parent of the other at some point. In *summary* or “not-sum” operation, instead of indicating the variables being summed over, those variables *not* being summed over are indicated. Figure 2.3 shows a sum-product message passing procedure.

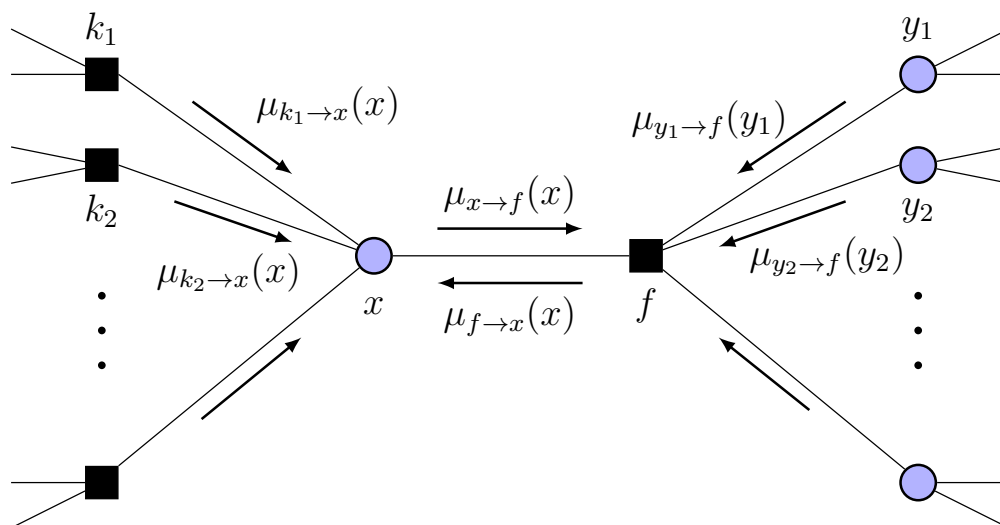


Figure 2.3: Factor-graph fragment depicting update rules of sum-product message-passing algorithm.

To obtain a mathematical expression for the belief-propagation, let  $n(v)$  denote a set of all neighbors of a node  $v$ ,  $\mu_{x \rightarrow f}(x)$  represent a message sent from node  $x$  to  $f$ , and let  $\mu_{f \rightarrow x}(x)$  denote the message sent from factor node  $f$  to variable node  $x$ . Therefore, as illustrated in Figure 2.3, the message computations are expressed as follows [21]

$$\mu_{x \rightarrow f}(x) = \prod_{k \in n(x) \setminus \{f\}} \mu_{k \rightarrow x}(x), \quad (2.12)$$

$$\mu_{f \rightarrow x}(x) = \sum_{\sim \{x\}} \left( f(x_C) \prod_{y \in n(f) \setminus \{x\}} \mu_{y \rightarrow f}(y) \right), \quad (2.13)$$

where  $x_C = n(f)$  is the set of arguments of the function  $f$ .

Considering  $g_3(x_3)$  in Figure 2.2 as an example,  $g_3(x_3)$  can be obtained as follows:

$$g_3(x_3) = \mu_{f_3 \rightarrow x_3}(x_3), \quad (2.14)$$

where

$$\mu_{f_3 \rightarrow x_3}(x_3) = \sum_{\{x_1\}} f_3(x_1, x_3) \mu_{x_1 \rightarrow f_3}(x_3), \quad (2.15)$$

and

$$\mu_{x_1 \rightarrow f_3}(x_3) = \mu_{f_1 \rightarrow x_1}(x_1) \mu_{f_2 \rightarrow x_1}(x_1). \quad (2.16)$$

At the end of each message-passing cycle, a variable node updates according to the rule given in equation (2.12), while the update rule at a local function node is given in equation (2.13). This iterative process continues until convergence, i.e., when no significant difference in belief update occurs. Then the algorithm terminates, and  $g_i(x_i)$  is computed as the product of all messages directed toward  $x_i$ .

The factor graph representation is convenient and is known for its many applications in probability theory, systems biology, cyber-physical systems, coding theory, and complex networks [19, 24, 25, 32, 33]. In systems biology, the factor graph representation has been successfully employed to predict the impact of single gene deletions on the global dynamic behavior of gene regulatory networks (GRNs) [32], refine regulatory networks of gene interactions with improved fit to experimental data and discover new regulatory relationships [19], optimize gene regulation functions [6], and even characterize the steady-state behavior of Bayesian gene networks [34].

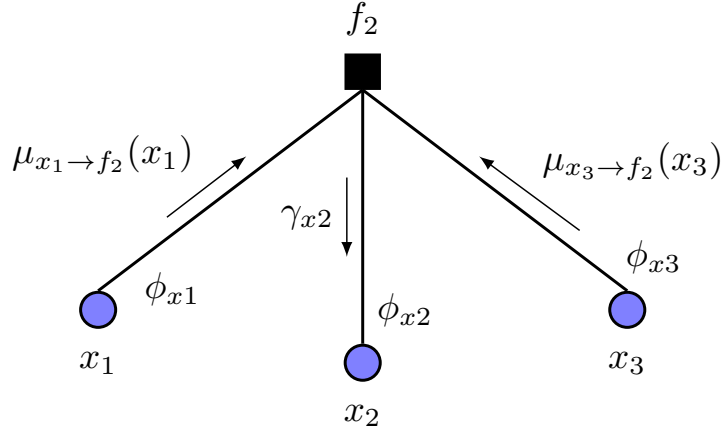


Figure 2.4: Message-passing in factor graph with erroneous messages, where  $\phi_{(\cdot)}$  represents probability of error message from variable node to factor node, and  $\gamma_{(\cdot)}$  denotes probability of error message from factor node to variable node.

## 2.4 Density Evolution in Factor Graphs

In the literature, density evolution refers to tracking the probability density function of error messages between variable nodes and factor nodes with the aim of studying the behavior and performance of a factor graph model [25, 35]. In DE algorithm, the distributions of messages from variable nodes to factor nodes at two consecutive iterations of belief propagation are connected by a recursive formula.

As an illustrative example, consider the factor graph shown in Figure 2.4 having a modulo 2 check-sum factor node functionality on the values of its neighboring variable nodes to zero. A belief-propagation algorithm is applied to the graph, where the messages passed between the variable and factor nodes are defined as log-likelihood ratios of probabilities that a given state is ‘0’ or ‘1’. Let  $\phi_l$  denote the probability of an error message from a variable node to a factor node, and let  $\gamma_l$  represent the probability of an error from a factor node to a variable node, both in the  $l$ -th iteration of a message-passing algorithm. Thus, conditioned on the event that the factor node has degree  $d$ , we have  $\gamma_l = 1 - (1 - \phi_l)^{d-1}$  under the assumption of independence [25]. Of note, the degree of a node in the factor graph denotes the number of edges incident to the node.

To obtain a closed-form expression for the total error probability in the graph, from

factor nodes to variable nodes, a generating function  $\rho(z) = \sum_d \rho_d z^{d-1}$  is defined, which represents the factor node degree distribution, where  $\rho_d$  is the fraction of nodes with degree  $d$ . Hence, we obtain the following:  $\gamma_l = 1 - \rho(1 - \phi_l)$ . For this example, the assumption is that a variable node performs no computation on the received messages but simply sends back the received messages on each edge. Then, for the next iteration,  $\phi_{l+1} = \gamma_l$ . Hence, the closed form recursive expression for the graph is written as  $\phi_{l+1} = 1 - \rho(1 - \phi_l)$ .

Density evolution analysis has been employed in other research fields such as multi-layered complex networks and coding theory. Behfarnia and Eslami [33] used DE analysis to study the dynamics of failure propagation and healing in networked cyber-physical systems. Likewise, Richardson and Urbanke employed density evolution to track the density of erasure messages in order to analyze the performance of a decoding algorithm for low-density parity-check (LDPC) codes and to find optimum degree distributions [24, 25]. In this work, I hypothesized that density evolution can be employed to provide an exact analytic characterization of the impact of random network perturbation on the steady-state behavior of biological networks.

## 2.5 Discretization of Gene Expression Data

Advances in RNA-seq technologies allow the simultaneous measurement of the expression of thousands of genes under different experimental conditions, enabling the unraveling and reverse-engineering of the interactions of the genes in an organism. Data discretization is frequently applied as a preprocessing step in the analysis of biological data. In several analyses, gene expression data (GED) discretization is required to allow the application of algorithms for the inference of biological knowledge that need discrete data as an input. Discretization maps real data into a typically small number of finite values. In this regard, the discretization of the data plays a key role in the outcomes of the gene expression analysis [36]. In general, discretization is carried out if prior biological knowledge suggests that the underlying variables are indeed discrete, or for computational efficiency. Since discretized data can be more stable with respect to random variations of gene-expression measurements [36], discretization can help improve the robustness of data and reduce noise in the continu-

ous variables. Also, discretization improves the efficiency of the inference algorithms owing to the reduced search space in which the extraction of knowledge is performed. Furthermore, in systems biology, discretization helps with the interpretation of the results because each discrete state has a direct meaning in its value.

The selection of a discretization technique is a non-trivial task. There are several factors to consider to make a suitable choice for a particular instance in the addressed biological problem. First of all, there is the issue related to the “level of discretization” used in the modeling, i.e. how many discrete states or quantization levels will be used to represent the gene expression levels. Two-states discretization, i.e., *upregulation* and *downregulation*, gives the simplest case for representation and the easiest way of interpretation of the results [6], although it does not allow any modeling of intermediate states. Another issue to consider is the inherent computational complexity in the discretization approach, as this may add an undesirable load in the computational time required to perform the expression analysis. Nevertheless, any discretization process implies loss of information.

This section describes a stricter formulation of a discretization problem. Let  $D'$  be a GED matrix of  $N$  rows and  $M$  columns, where  $d'_{ij}$  represents the expression level of the gene  $g_i$  under experimental condition  $j$ . A discretized matrix  $D$  results from the application of a discretization function,  $\Sigma_I$ , on  $D'$ , that maps each element  $d'_{ij}$  in  $D'$  to one of the elements in the set  $A = \{0, 1\}$ . Assume without loss of generality, that the discretization algorithm considers the values of each gene  $x_i$  of the GED  $D'$  independently. In this way, the discretization algorithm  $\Sigma_I(D', x_i)$  transforms the continuous gene expression values of  $x_i$  in the set  $D'$ , into  $k$  intervals  $I = \{[p_0, p_1], (p_1, p_2], \dots, (p_{k-1}, p_k]\}$ , where  $p_0$  and  $p_k$  are the minimal and maximal values in the gene expression profile of  $x_i$ , respectively. The algorithm thus transforms the matrix  $D'$  into a matrix  $D$  by mapping values using the learned intervals. Figure 2.5 shows an example of the workflow involved in the discretization process when dealing with two discrete gene states.

In the literature, several methods have been proposed to discretize or cluster gene-expression data [19, 36]. According to Gat-Viks et al. [19], the variable-specific discretization

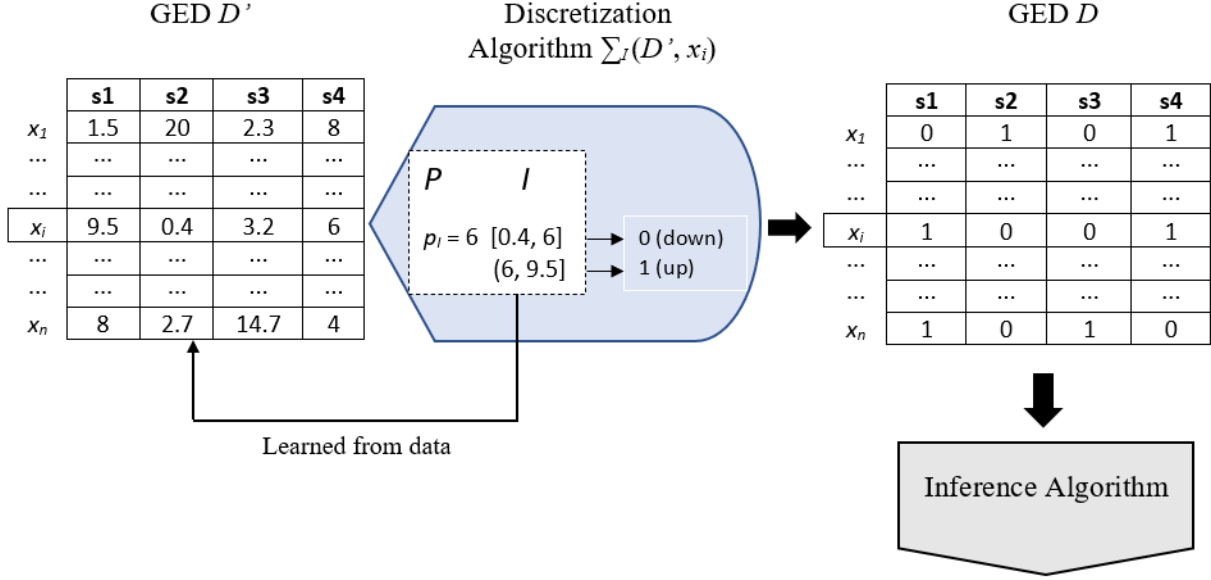


Figure 2.5: Discretization process workflow with two discrete states  $A = \{0, 1\}$  for gene  $x_i$ . GED  $D'$  and discretized  $D$  are composed by  $n$  genes and four experimental conditions. Discretization algorithm  $\Sigma_I$  takes  $D'$  and  $x_i$  and infers the cut point  $p = \{6\}$  and discretization scheme  $I = \{[0.4, 6], (6, 9.5]\}$  to obtain discretized expression profile of  $x_i$  in the GED  $D$ . Discretized GED  $D$  is the input of an inference algorithm that will extract some kind of information of interest.

method outperforms the global optimized single common discretization scheme. In addition, it is generally more accurate and flexible than standard clustering preprocessing methods used by Friedman et al. [18] for real gene-expression data. However, this flexibility may come at a cost of over-fitting and decreased learnability.

In this dissertation, I employed a gene-specific discretization scheme that optimizes the Gaussian mixture model likelihood using the iterative expectation-maximization (EM) algorithm [37] in the MATLAB environment. In each EM iteration, the posterior distributions of component memberships are inferred and then used to re-estimate the mixture proportions by computing the Gaussian sufficient statistics (component means, covariance matrices, and mixing proportions). The new discretization distributions are used in the next iteration, and the algorithm iterates until convergence.

## CHAPTER 3

### PROBABILISTIC MODEL ANALYSIS OF BIOLOGICAL NETWORKS

#### 3.1 Introduction

To analyze the behavior of gene networks, there is a need to study regulation functions and interactions among biological entities. Such relations, for instance, determine the level of gene expression for a particular gene from a set of transcription factors interacting with the gene. In practice, the information available for regulation mechanisms is incomplete and of variable certainty, which motivates the employment of a probabilistic framework for inferring such functions. In addition, due to the noisy nature of biological experiments, probabilistic models help integrate experimental data in a network context. Therefore, in this chapter, a probabilistic model is presented by defining variables and formulating prior biological knowledge on their relations. Each biological entity can be modeled as a discrete or continuous random variable. This random variable represents the level in which an entity is present in the cell. Taking partial information into account, I derived a distribution function for each variable that considers interaction relations among them as well as their level of uncertainty.

The model is formalized as a probabilistic factor graph [21], which can handle highly complex systems and extensive datasets. In addition, the model allows us to overcome the drawbacks of models that assume noiseless observations, because it is able to mix noisy continuous measurements with discrete regulatory relations among variables. Furthermore, it does not require the explicit determination of network kinetic parameters. This probabilistic framework can enable us to model uncertainty in cellular networks through integration of prior biological knowledge and high-throughput experimental data.

Then, I employed the proposed probabilistic model to characterize the steady-state behavior of two experimentally verified molecular networks in *Escherichia coli*, namely “DNA damage response network” and “acid-resistance regulatory network.” In addition, the model

was applied to statistically assess the global consistency between networks and their corresponding gene expression profiles from diverse experimental conditions. The simulation results showed a high correlation between the observed experimental states and the proposed model’s inferred system behavior under various experimental conditions. Likewise, the example networks studied are strongly supported by their corresponding gene-expression measurements. Furthermore, the proposed model can discriminate good molecular networks from poor networks, which is important, particularly in assessing the quality of inferred biological networks from genomic data.

### 3.1.1 Related Work

In the literature, some methods that can take a biological network and simulate biological data of different genes as either time-series data or steady-state values have been proposed. One of these is *sgnesR* [38], an *R* package used to simulate a gene expression profile from a given gene network using the stochastic simulation algorithm (SSA), for which the reaction parameters are specified under defined constraints. Similarly, a multi-view genomic data simulator (MVBioDataSim) proposed by Fratello et al. [39] can generate synthetic biological data from ordinary differential equation (ODE)-based network models with known parameters, constructed through an iterative procedure. Simulated datasets, although fully controlled, are often too simplistic to efficiently explain the complex regulatory interactions among biological entities compared to real gene expression data.

Another widely used simulation and modeling tool in systems biology is the complex pathway simulator (COPASI) [40, 41]. This stand-alone program specializes in setting up and analyzing biochemical and kinetic network models while also providing some basic stoichiometric analyses. It allows for more detailed and fine-grained analysis, but also demands more knowledge, namely about the kinetics of individual processes. An important factor in the simulation of these models is the knowledge of kinetic reaction parameters. This information can be extracted from the literature; however, it is difficult to find [40]. Lack of kinetic constants stem from the difficulty in measurements and uncertainties in the function of many proteins and their interactions, thus limiting the application of some of

these approaches. However, these simulators provide valuable information that can be used to test network inference methods qualitatively, as well as to identify model parameters.

## 3.2 Model and Methods

### 3.2.1 Gene Network Representation

Figure 3.1(a) shows an example of a simple gene network structure having four genes. An unsigned directed edge from gene  $g_2$  to  $g_1$ , for instance, means gene  $g_2$  influences the action of  $g_1$ , through creation of some specific proteins [6]. Altering the state of one gene can cause other genes interacting with it to change their states, leading to a cascade of modifications in their alleles. This process is repeated iteratively in time until a global steady state is reached. In general, let  $g = \{g_1, g_2, \dots, g_n\}$  denote the set of biological random variables, i.e., genes in a network. Here, genes are considered as discrete random variables. Each node  $g_i$  may take a value from a range of (usually finite)  $k$  logical states  $s = \{0, 1, \dots, k - 1\}$  that a variable may attain. Also,  $Pa_i = \{Pa_{i,1}, Pa_{i,2}, \dots, Pa_{i,m}\} \subseteq g$  denotes the set of *parents* of  $g_i$  (i.e., set of variables that regulate  $g_i$ ).

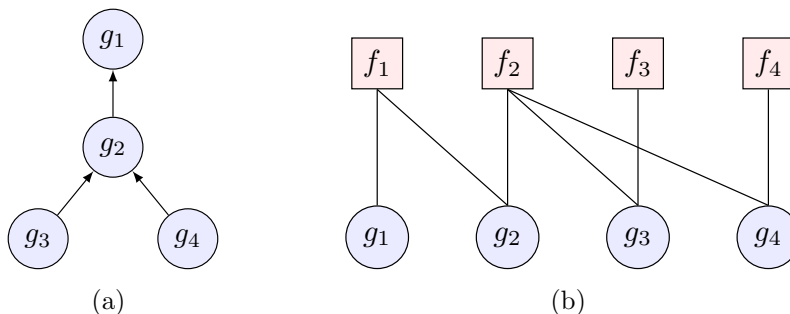


Figure 3.1: Graphical probability models: (a) Bayesian gene network, and (b) factor graph equivalent.

By definition, the regulation function  $f_i$  for a variable  $g_i$  is formulated by the conditional probabilities as

$$f_i \triangleq p(g_i | Pa_i). \quad (3.1)$$

This function  $f_i$  is referred to as the belief that  $g_i$  takes a certain state with respect to an assignment from its parents. If  $Pa_i = \emptyset$  (i.e.,  $g_i$  has no parents), then  $p(g_i|\emptyset) = p(g_i)$ . The learning problem of  $f_i$  is defined as selecting the maximum likelihood of data given the model. Formally, the Bayesian network shown in Figure 3.1(a) implicitly encodes the joint probability distribution as a product of local conditional distributions:

$$p(g_1, g_2, \dots, g_n) = \frac{1}{Z} \prod_{i=1}^n p(g_i|Pa_i), \quad (3.2)$$

where  $Z$  is a normalization constant. In this chapter, I employed a probabilistic factor graph [21], which explicitly expresses the structure of the joint distribution's factorization in equation (3.2). For example, Figure 3.1(b) expresses the factorization

$$p(g_1, g_2, g_3, g_4) \propto p(g_1|g_2)p(g_2|g_3, g_4)p(g_3)p(g_4). \quad (3.3)$$

A factor graph visualized as an undirected graph associating variable nodes and factor nodes is defined as a bipartite graph (see Figure 3.1(b)), for instance, of the simple network Figure 3.1(a). In this context, a *variable node* denotes each random variable  $g_i$ , and a *factor node* denotes a local function  $f_i$ . To convert a Bayesian network to a bipartite graph, an edge is simply drawn between a variable  $g$  and a factor  $f_j$ , if the scope of  $f_j$  contains  $g$ . This representation is convenient and has been successfully utilized in the literature to discover new regulatory relationships and even optimize regulation functions [6, 19]. However, in practice, biological dependency graph models contain loops, and the situation becomes more complex. In this case, the probabilistic factor graph network (FGN) model approximates rather than giving the true belief functions [42]. Furthermore, it is noteworthy that, given experimental measurements at the single-cell level, the regulation functions may suffice to describe the inherent stochasticity of the underlying biochemical reactions.

### 3.2.2 Inference in FGN Model

This section discusses the problem of probabilistic inference in the proposed FGN model. Typically, each experiment provides partial information on the state of model vari-

ables. Therefore, given experimental evidence, for example, the gene expression profile  $D$  specifying real-value sensor variables and the graphical model, the problem of probabilistic inference seeks computation of the *posterior* distributions for a single hidden (unmeasured) variable. This is referred to as *marginal inference*. Of particular interest, for instance, would be to compute  $p(g_i|D)$ . Another problem would be to compute the *likelihood*  $p(D)$  of the evidence. The probabilistic inference is an NP-hard problem [43] because it can involve summing an exponentially large number of terms.

In directed acyclic graphs, many exact inference techniques exist in the literature, such as variable elimination, naive brute force marginalization, and the family of message-passing algorithms such as junction tree, sum-product, and belief propagation, which perform very well [21, 26, 44]. However, for loopy graphs (graphs that contain cycles such as GRNs), messages may circulate indefinitely in loops and hence do not guarantee convergence. Moreover, even if they do, the steady state may not represent marginals of the nodes. Accordingly, sufficient conditions to guarantee uniqueness of fixed points and/or convergence have been studied in the literature [45]. In this work, I explored the loopy belief propagation (LBP) algorithm, a popular message-passing algorithm on loopy graphs. This algorithm belongs to a class of *variational* algorithms, which approximate the marginal distribution functions, assuming certain decomposition over a cluster of variables or independent variables [42, 45, 46]. Essentially, it is a fixed-point iterative procedure that tries to minimize the *Bethe* Free Energy,  $F_{\text{bethe}}$  [42]. Still it is a well-defined algorithm and empirically often achieves a good approximation if the solution converges [46].

In the proposed FGN model, belief propagation is used as the message-passing protocol. Messages sent between “variable nodes” corresponding to dashed ellipses for the equivalent FGN as shown in Figure 3.2 are computed as functions of the parent  $p$ . A message sent from  $p$  to child  $c$  is denoted as  $\mu_c(p)$ , while a message sent from  $c$  to  $p$  is denoted as  $\lambda_c(p)$ . Note that within an ellipse, the message sent from  $g_i$  to the local function  $f_i$  is given by the product of all incoming  $\lambda$  messages. Similarly, a message from  $f_i$  to  $g_i$  is the product of  $f_i$  with other  $\mu$  messages received at  $f_i$  summarized for  $g_i$ .

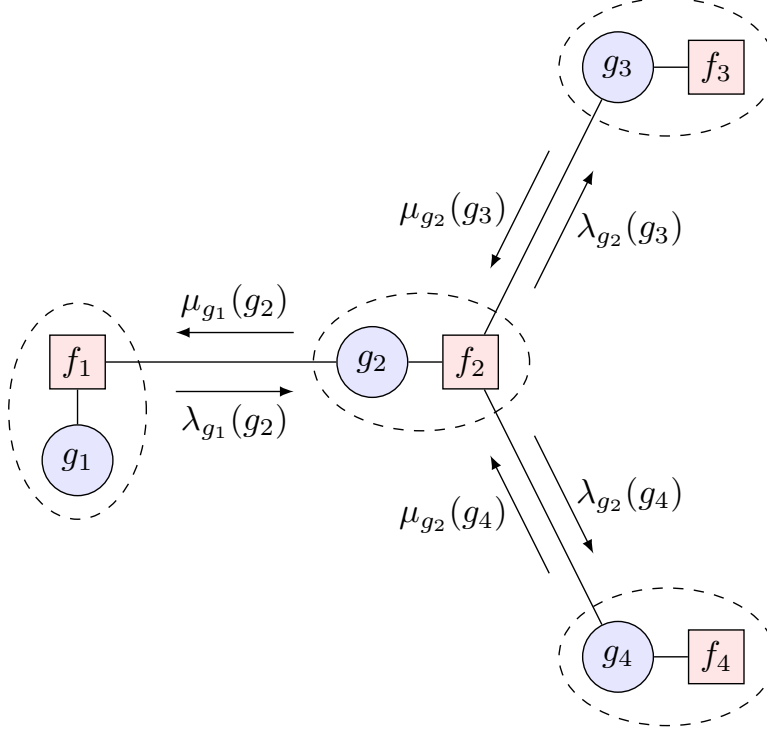


Figure 3.2: Messages sent in belief update of factor graph network example shown in Figure 3.1(b). Arrows indicate flow of belief messages sent between variable nodes (i.e., dashed ellipses corresponding to nodes in Bayesian gene network),  $\mu$  denotes message sent from parent gene to child gene in FGN model, and conversely,  $\lambda$  represents message from child to parent.

Formally, denote the set of parents of a gene variable  $g_i$  by  $Pa_i$ , and the set of children of  $g_i$  by  $C_i$ . Therefore, for every  $a \in Pa_i$ ,

$$\lambda_{g_i}(a) = \sum_{\sim\{a\}} \left( \prod_{d \in C_i} \lambda_d(g_i) p(g_i | Pa_i) \prod_{h \in Pa_i \setminus \{a\}} \mu_{g_i}(h) \right), \quad (3.4)$$

and for every  $d \in C_i$ ,

$$\mu_d(g_i) = \prod_{c \in C_i \setminus \{d\}} \lambda_c(g_i) \sum_{\sim\{g_i\}} \left( p(g_i | Pa_i) \prod_{a \in Pa_i} \mu_{g_i}(a) \right), \quad (3.5)$$

where  $\sum_{\sim\{x\}}$  is the summary operator over all variables except  $x$ . Nodes are updated in parallel, and both  $\lambda$  and  $\mu$  messages are normalized at each iteration. Computation of a marginal function of an individual variable, referred to as the *belief* of a node in the factor graph and denoted by  $b_i(\cdot)$ , is given by the product of all messages received by the node:

$$b_i(g_i) = \prod_{d \in C_i} \lambda_d(g_i) \sum_{\sim\{g_i\}} \left( p(g_i | Pa_i) \prod_{a \in Pa_i} \mu_{g_i}(a) \right). \quad (3.6)$$

The message-passing algorithm continues until the solution converges or no significant difference in belief update occurs in the order of  $10^{-4}$ . In all inferences done here, the initial messages were set to a uniform vector of ones. However, in certain cases, random initialization yielded similar results since the initial conditions rapidly “fade out.”

### 3.2.3 Network Models and Data

In this chapter, I tested the applicability of the proposed model on two experimentally verified reference networks in *Escherichia Coli* [47] with real gene expression data. Of particular interest in *E. coli* are the cyclic gene regulatory small sub-networks known as the “SOS DNA repair network” [17, 48] and the “acid resistance regulatory network” [49, 50].

The dataset of expression profiles used to model and evaluate the probabilistic framework consists of uniformly normalized gene expression data in the M<sup>3D</sup> database [51]. This compendium of data facilitates large-scale computational analyses by providing a bulk download of human-curated, computable experimental metadata, and computer-validated data for integrity. The compendium data used on *E. coli* (version 4 build 6) contain 466 microarray expression profiles of 4,297 genes collected under a wide range of experimental conditions, including wild type, pH changes, varying oxygen concentrations, antibiotics, heat shock, growth phases, different media, environmental perturbations, gene knockout/knockdown, and time series at the steady-state level. In addition, the expert knowledge in RegulonDB [47] enabled us to assess and validate the network models.

#### ***Escherichia Coli*: SOS Response Network Model**

Figure 3.3 shows an SOS response network that includes 9 genes with 24 edges. The pathway regulates and coordinates cell survival and repair after extensive DNA damage, which involves *lexA* and *recA* genes as principal mediators. LexA is a repressor protein dimer for the majority of DNA repair genes while the cell is healthy. On the other hand,

RecA protein acts as a sensor of DNA damage that induces the response by inhibiting LexA [17].

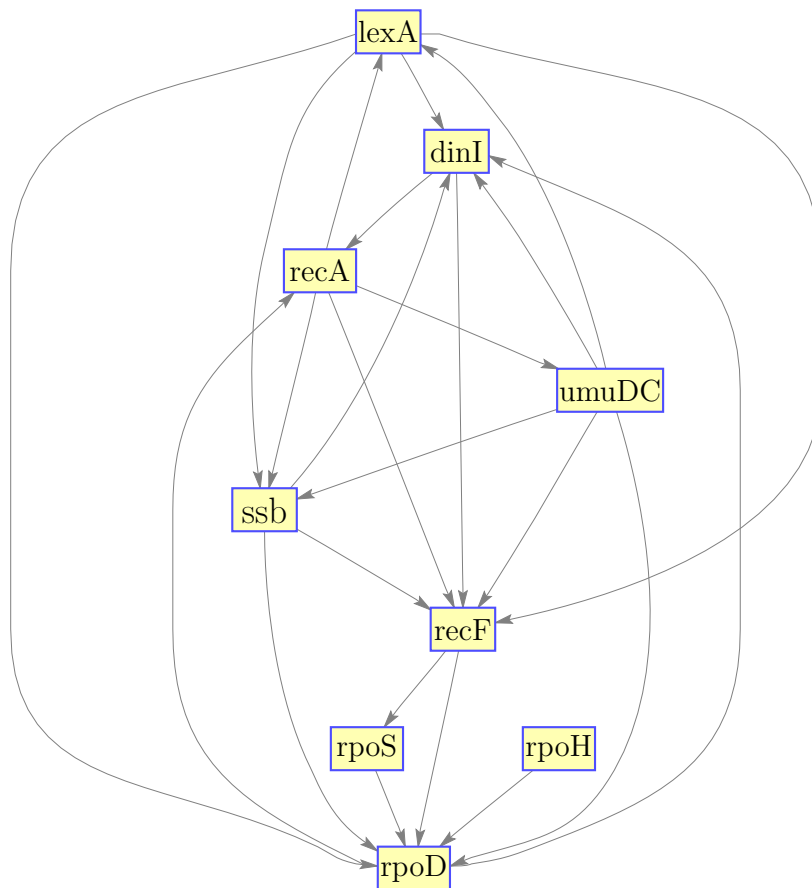


Figure 3.3: Graphical representation of *E. coli* SOS DNA repair true pathway [17].

### ***Escherichia Coli*: Acid Resistance Network Model**

*E. coli* has a potent acid-resistant (AR) system enabling it to survive extreme acidic environments ( $\text{pH} < 2.5$ ). As such, from both medical and fermentation points of view, the physiological and molecular response to acid shock has been the subject of intense study. In this section, I considered an efficient glutamate-dependent AR regulatory network shown in Figure 3.4 and also reported in [49] and [50]. The pathway has LuxR-family member GadeE as the central activator. Moreover, it has at least 11 known regulatory proteins that affect the induction of the response (see Table 1 in [50]).

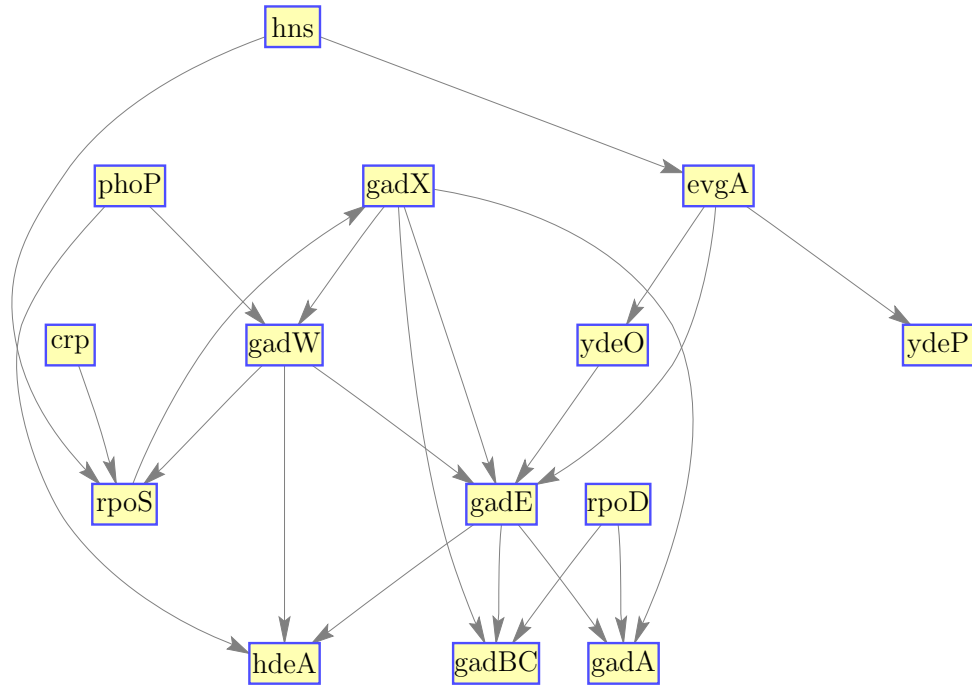


Figure 3.4: Graphical representation of acid-resistant gene regulatory network [49]. Overexpressed EvgA response regulator indirectly activates acid resistance through transcription of *ydeO* (i.e.,  $\text{EvgA} \rightarrow \text{YdeO} \rightarrow \text{GadE} \rightarrow \text{AR}$ ). However, without overexpression (under normal inducing conditions), EvgA can directly activate *gadE* transcription.

### 3.2.4 Discretization of Data

In probabilistic graphical models, discretizing a real-value sensor or node measurement is an integral part of the model, in order to fully account for dependencies between regulation function priors and the discretization scheme. Generally, this step is carried out if prior knowledge suggests that the underlying variables are indeed discrete, or for computational efficiency. In addition, it helps improve the robustness of data and reduce noise in the continuous variables, since discretized data can be more stable with respect to random variations of mRNA measurements [36]. However, in the biological field, discretization may result in loss of information.

Finding the optimal discretization is an NP-complete problem [52]. In the literature, several methods to discretize microarray data have been proposed [18, 19, 36]. According to

[19], for real biological gene expression data, the variable-specific discretization scheme outperforms the global optimized single common discretization method, and is generally more accurate and flexible than the standard preprocessing approaches used by [18]. However, this flexibility can lead to overfitting and may decrease learnability. I applied gene-specific discretization and employed mixtures of Gaussian distribution to model the relations between continuous observations on a gene variable and its discrete logical state, i.e., each Gaussian component corresponds to a specific state. Given the experimental evidence and a logical function prior for each variable, the discretization functions are optimized using the expectation-maximization (EM) learning algorithm.

In the analysis of the model, I used two- and three-quantization levels with values 0,1 and 0,1,2 states, respectively. In these contexts, each state (i.e., Gaussian component) may correspond to a different range of gene expression levels for different genes, defined by the estimated parameters of the Gaussian mixture model (GMM). Therefore, given an observation sample point for a particular gene, we can say that the sample point most likely belongs to a given state with a certain probability. For instance, the *lexA* gene in the SOS response network has a normalized gene expression data range between 7.60 and 12.70. According to the two-states discretization, the *lexA* GMM parameters were defined by a 0-state mixture proportion of 0.7939, mean of 9.341, and variance of 0.4389, and, similarly, a 1-state mixture proportion of 0.2061, mean of 11.51, and variance of 0.1016.

### 3.3 Results and Analysis

Here, I constructed a probabilistic FGN model representing the regulatory relations for each of the considered networks and applied the LBP inference algorithm to estimate the marginal posterior distributions on all gene logical variables. To test the prediction accuracy of the proposed model, I computed the statistical Pearson correlation,  $\rho$ , between the inferred marginals of each gene variable  $g_i$  and the probability of  $g_i$ 's observed states given by experimental observations. Furthermore, to verify results I compared the performance of the probabilistic model on the true networks against random networks. The randomization analysis is performed to provide a framework to test the consistency between an experimentally

verified gene network against its corresponding gene expression profiles. I implemented two network perturbation methods to obtain random networks. The first perturbation method produces a network with identical topology to the original regulatory model; however, the expression data are perturbed by uniformly redistributing the whole expression profiles of genes using the Fisher-Yates shuffle algorithm [53]. Entire gene profiles are swapped in order to keep each profile internally consistent, in order to be able to quantify how consistent a random set of interactions is with respect to the gene expression compendium data. This method is known as *randomized node* perturbation because the process depicts permutation of nodes on a graph.

The second method, referred to as *randomized edge* perturbation, perturbs the regulatory network topology while preserving the node degree distribution in all nodes as well as preserving the expression profiles of individual genes. To achieve this, I implemented a simple numerical algorithm, as proposed in the work of [54]. The algorithm works by selecting two existing edges,  $(i, j)$  and  $(k, l)$ , and reconfiguring their endpoints such that the new edges become  $(i, l)$  and  $(k, j)$ . If any of these new edges already exists, then the procedure is terminated and a different pair is selected instead. This process is repeated  $10 \cdot |E|$  times, where  $|E|$  is the cardinality of edges in the network. The resultant network is a randomized version of the original network. Both methods preserve the degree of all variable nodes in a given network to rule out the impact of node degree distribution on consistency; degree distribution is an important characteristic for nodes in biological regulatory networks [22]. Lastly, I performed multiple sampling of the resultant random networks and computed their average correlation.

### 3.3.1 SOS Response Model

Tables 3.1 and 3.2 summarize the model predictions (i.e., steady-state marginals) for the nine genes after convergence against the logical proportions of the discretized compendium data. The message updates converged at an average of 19.5 and 11.5 iterations for 2-states and 3-states discretization, respectively, using two different types of initialization, i.e., random and uniform.

Table 3.1: Two-logical states distribution of SOS response network’s predicted marginal posteriors versus observed experimental states

Gene	Predicted Marginals		Observed States	
	0	1	0	1
lexA	0.9213	0.0787	0.7876	0.2124
dinI	0.2643	0.7357	0.2918	0.7082
umuDC	0.7019	0.2981	0.7232	0.2768
recA	0.5695	0.4305	0.5687	0.4313
ssb	0.9892	0.0108	0.9292	0.0708
recF	0.1005	0.8995	0.2275	0.7725
rpoS	0.7722	0.2278	0.7296	0.2704
rpoH	0.7342	0.2658	0.7232	0.2768
rpoD	0.8670	0.1330	0.8648	0.1352

*Note:* At the 95% confidence interval,  $t$ -statistics mean difference between model-predicted marginals and observed states is expected to lie between  $-0.0664$  and  $0.0829$  (i.e., for each discretization state).

Table 3.2: Three-logical states distribution of SOS response network’s predicted marginal posteriors versus observed experimental states

Gene	Predicted Marginals			Observed States		
	0	1	2	0	1	2
lexA	0.9999	0.0	0.0001	0.7811	0.0944	0.1245
dinI	0.7112	0.0029	0.2859	0.4936	0.2553	0.2511
umuDC	0.9772	0.0	0.0228	0.7339	0.0386	0.2275
recA	0.6554	0.3429	0.0017	0.4807	0.2961	0.2232
ssb	0.0501	0.9261	0.0238	0.1567	0.7511	0.0922
recF	0.1104	0.5605	0.3291	0.1373	0.4850	0.3777
rpoS	0.8142	0.1676	0.0181	0.7704	0.1996	0.0300
rpoH	0.6626	0.3374	0.0	0.6524	0.3433	0.0043
rpoD	1.000	0.0	0.0	0.9678	0.0193	0.0129

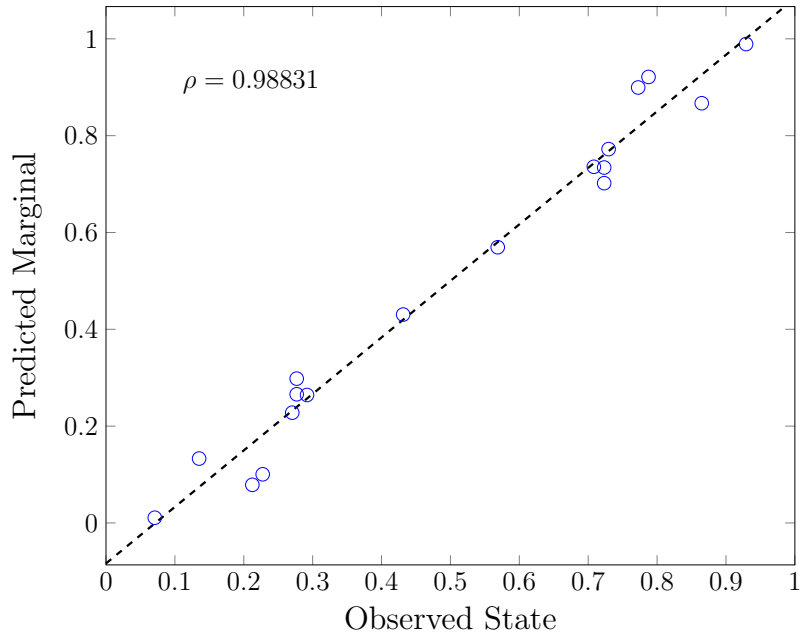
*Note:* At the 95% confidence interval, the  $t$ -statistics mean difference between model-predicted marginals and observed states is expected to lie between  $-0.0435$  and  $0.2229$  (i.e., for 0-state level).

Then, I constructed a 95% confidence interval using  $t$ -statistics to estimate the mean difference between the inferred marginals and the observed experimental states. For each level state, the expected mean difference interval contained zero. Thus, at the 95% confidence

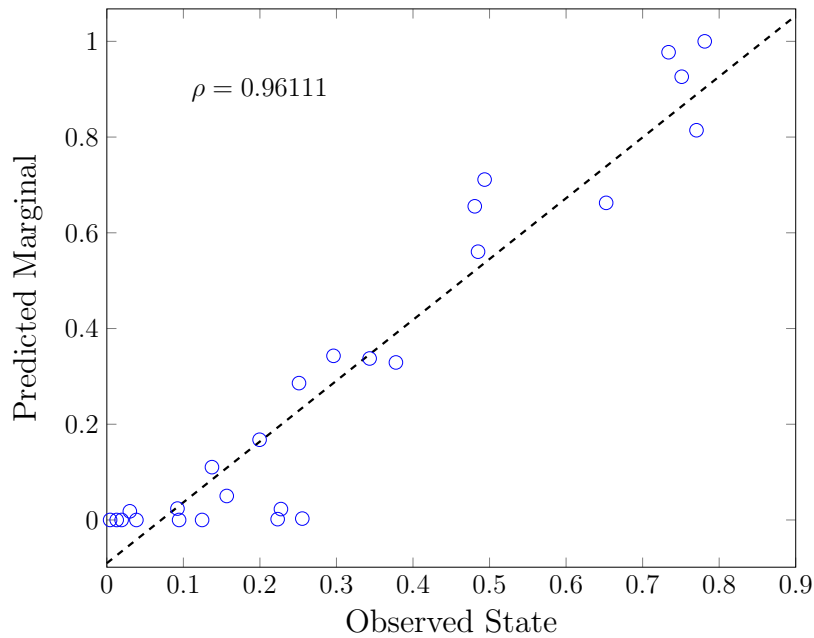
level, the results show that there is statistically no significant difference between the proposed model’s predicted marginals and the observed experimental states. Furthermore, I show the correlation plots between the variable gene  $G_i$ ’s inferred marginal posterior in equation (3.6) at convergence and the corresponding probability of its experimental observations in Figure 3.5. In both plots, the correlation results are well above 0.95, thereby indicating good approximations of the observed states.

Next, I investigated the performance of the model under the network perturbation methods mentioned above. Figure 3.6(a) summarizes the comparison between inferred marginal posterior distributions and the observed experimental states for random-shuffled datasets. Figure 3.6(b) does the same for randomized topology. In general, the Pearson correlation coefficient deteriorates with increased randomization, which highlights deviation of the model predictions from experimental observations. The simulation results demonstrate that random regulatory networks are inconsistent to and do not adequately explain the observed gene expression profiles.

To further test the consistency between extant real biological data and the given GRN, I demonstrated the performance of the proposed FGN model through a gene deletion or knockout (KO) experiment. In gene KO experiments, the expression of a target protein molecule is stopped by removing the protein-coding regions from the genome. From the dataset, *recA* was identified as the most knocked-out gene in the experimental samples, having a total of 68 observations. Next, I modified the model accordingly by fixing the state of the *recA* variable node to zero and eliminating the corresponding factor node, then applied LBP to estimate the marginal posteriors of the other variables. Then, I compared the predictions against the observed states. Figure 3.7 depicts the model prediction accuracy under *recA* gene KO experimental conditions in which both 2- and 3-states discretization obtain correlation values of  $\rho > 0.90$ . These results confirm that the given sub-network topology in *E. coli* is consistent with the gene expression data obtained from various biological experiments.

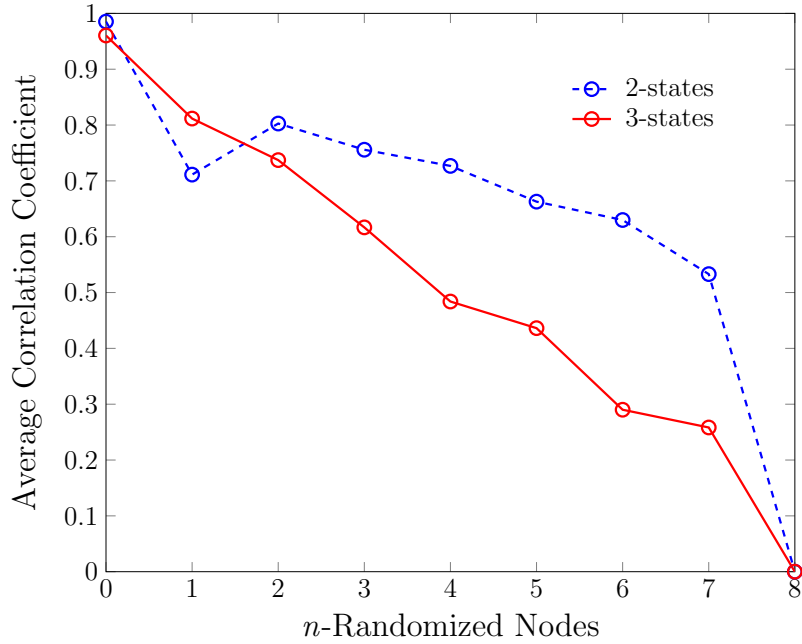


(a)

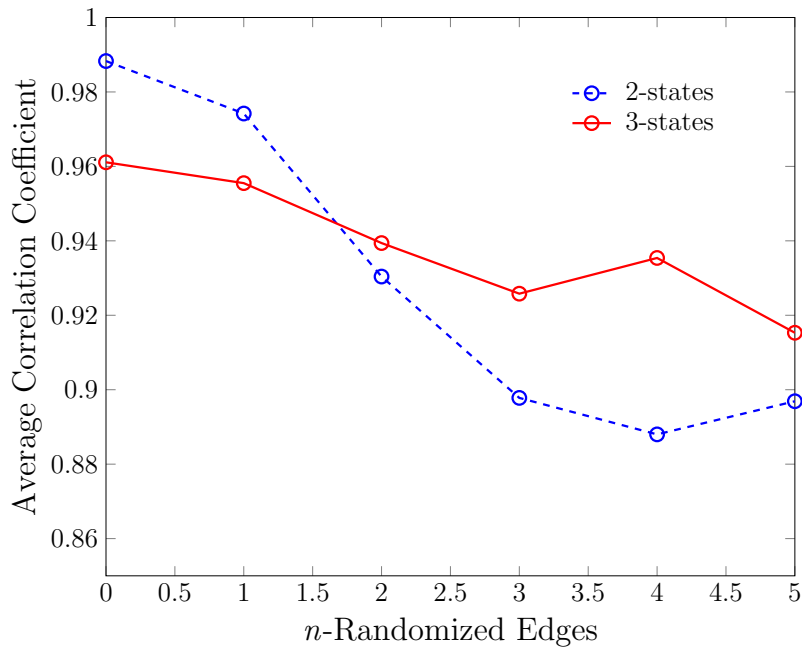


(b)

Figure 3.5: Pearson correlation plots between proportions of observed states and FGN inferred marginal posteriors for SOS response network: (a) 2-states discretization, p-value =  $1.6858 \times 10^{-14}$ , and (b) 3-states discretization, p-value =  $1.7540 \times 10^{-15}$ . Correlation coefficient  $\rho$  is given in each plot.

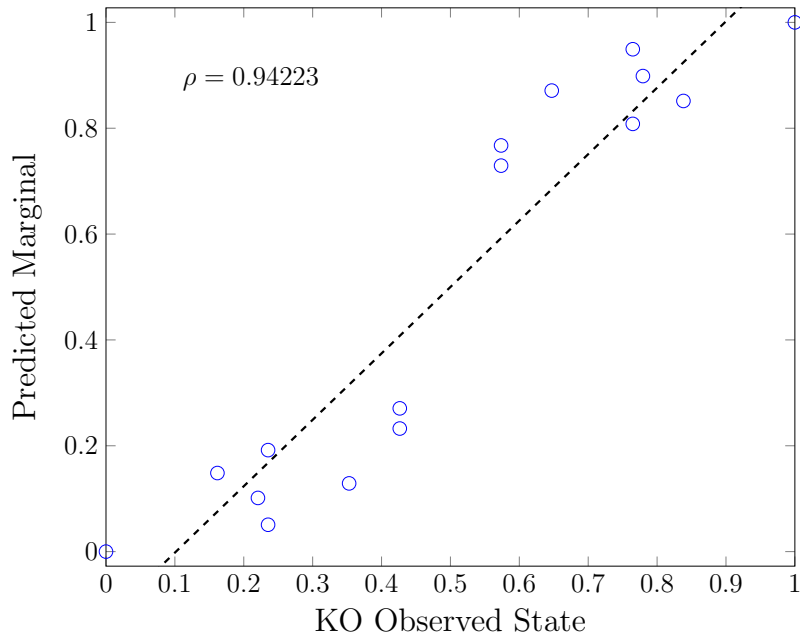


(a)

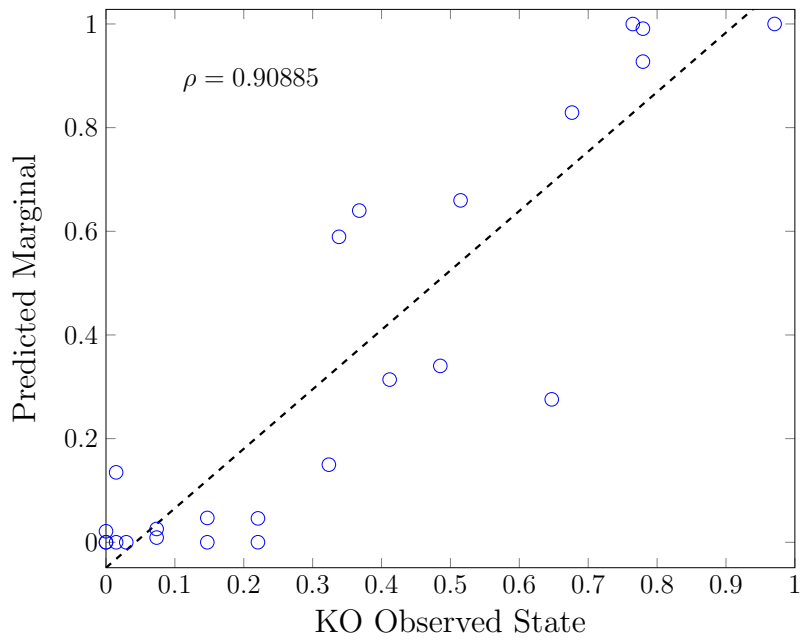


(b)

Figure 3.6: Average Pearson correlation coefficient plots of  $n$ -randomized nodes or edges for SOS response network. Model predictions are repeatedly evaluated, and averages of correlation coefficients computed in  $10 \cdot |E|$  random attempts: (a) randomly shuffled datasets where states are swapped between different variable nodes, and (b) randomly shuffled  $n$ -edges over network structure.



(a)



(b)

Figure 3.7: Pearson correlation plots between proportions of gene experimental observations and FGN inferred beliefs in *recA* gene KO model: (a) 2-states discretization, and (b) 3-states discretization. Correlation coefficient  $\rho$  is given in each plot.

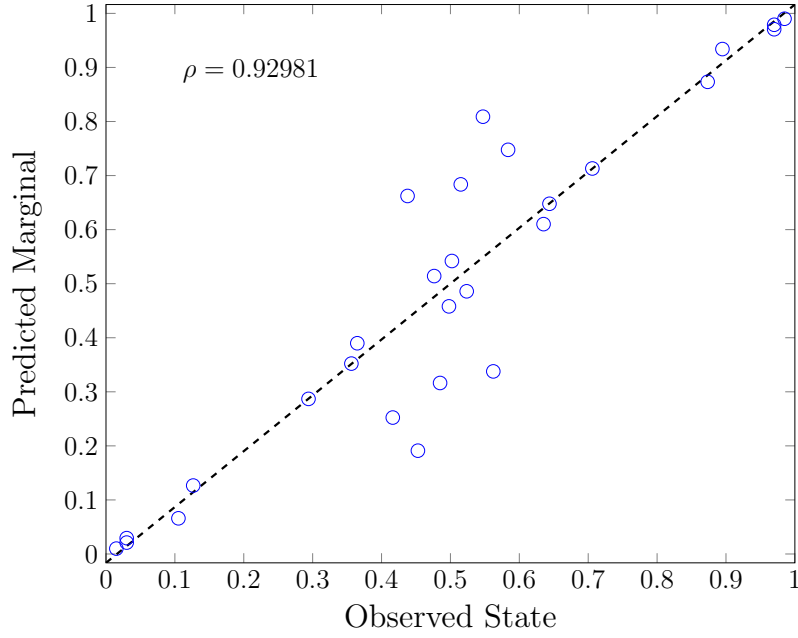
### 3.3.2 Acid-Resistance Model

For the AR gene network (see Figure 3.4), similarly, I constructed an FGN model and applied the LBP inference algorithm to estimate the marginal beliefs. In this model, the message update algorithm converged at 14 and 20 iterations for 2- and 3-states discretization levels, respectively. Initial messages at the variable nodes were set to a uniform vector of ones. For the gene regulation functions, the Pearson correlation plots are shown in Figure 3.8. Here, too, the correlation coefficients are greater than 0.90. Furthermore, the discrepancy of inferred marginals against randomized node- and edge-network models were tested as in the case of the SOS response pathway. Similar observations were made, as depicted in Figure 3.9. However, in this model, performance under gene KO experimental conditions are not shown since the available expression data do not have any of the genes participating in this pathway as a deleted gene.

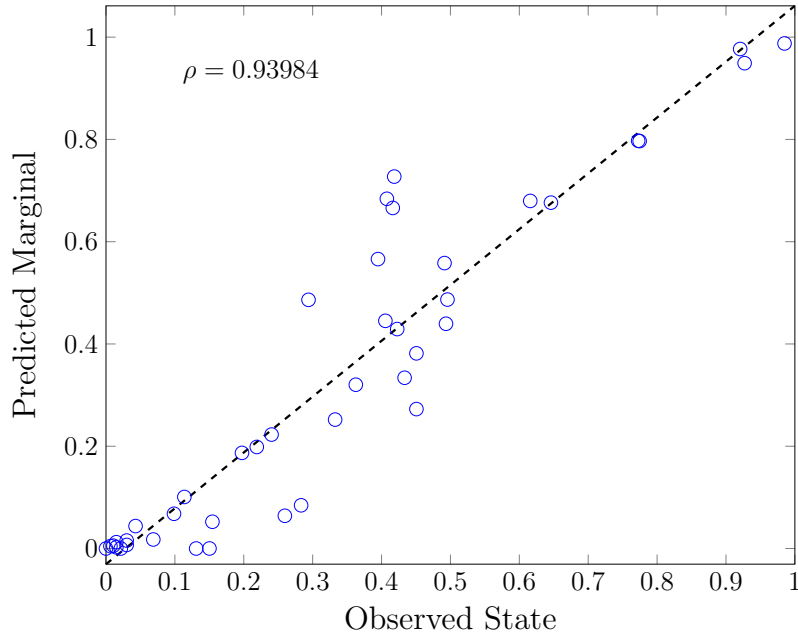
### 3.3.3 Sparse and Dense Networks

Here, I demonstrated the performance of the model to discriminate good GRNs from bad ones. First, I implemented very sparse and dense networks by deleting and adding links between gene nodes in the true gene regulatory network, in order to represent a poor network. In the SOS response network, sparse networks were created by removing 5–17 random edges. Similarly, in the AR regulatory network, I deleted between 3 and 15 random edges. On the other hand, dense networks were created by adding 5–30 random edges and 5–40 random edges, in the SOS response and AR regulatory networks, respectively. For each sparse or dense network analysis, the model predictions were repeatedly evaluated 100 times and the average Pearson correlations computed. Moreover, in the analysis of sparse gene networks, it is assumed that any isolated gene node is not part of the network and set its inferred marginal to a vector of all zeros.

Figure 3.10 shows boxplots of the average correlation coefficients comparison between the true regulatory networks and the poor networks for both 2- and 3-states discretization levels. The SOS response network model predictions are shown in Figure 3.10(a). As can



(a)



(b)

Figure 3.8: Pearson correlation plots between observed states node proportions and FGN inferred marginal posteriors for AR response network: (a) 2-states discretization, p-value =  $8.5579 \times 10^{-13}$ , and (b) 3-states discretization, p-value =  $2.920 \times 10^{-20}$ .

be seen, for the 2-states discretization analysis, the median correlation coefficient decreased to 0.8301 (16% decrease) and 0.8886 (10% decrease) from the true gene network correlation

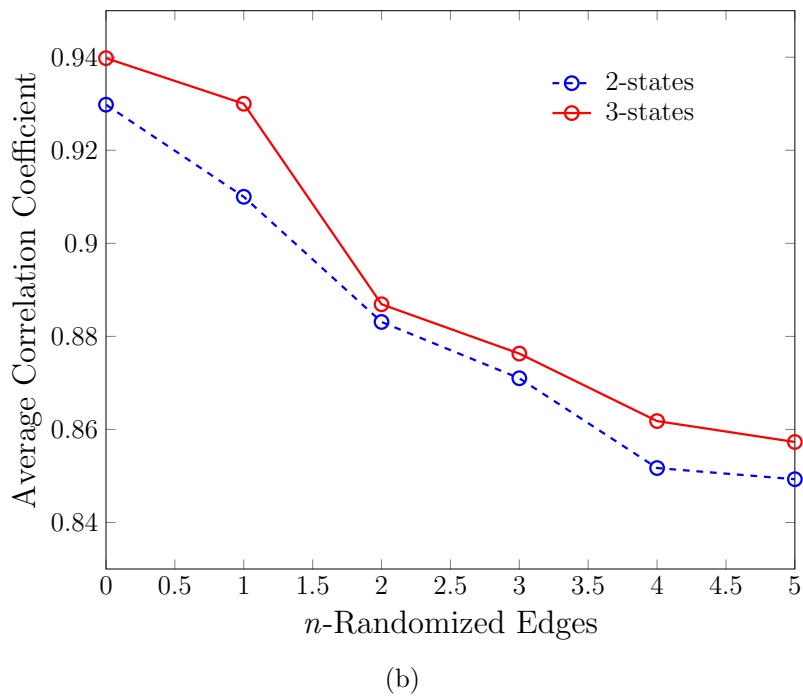
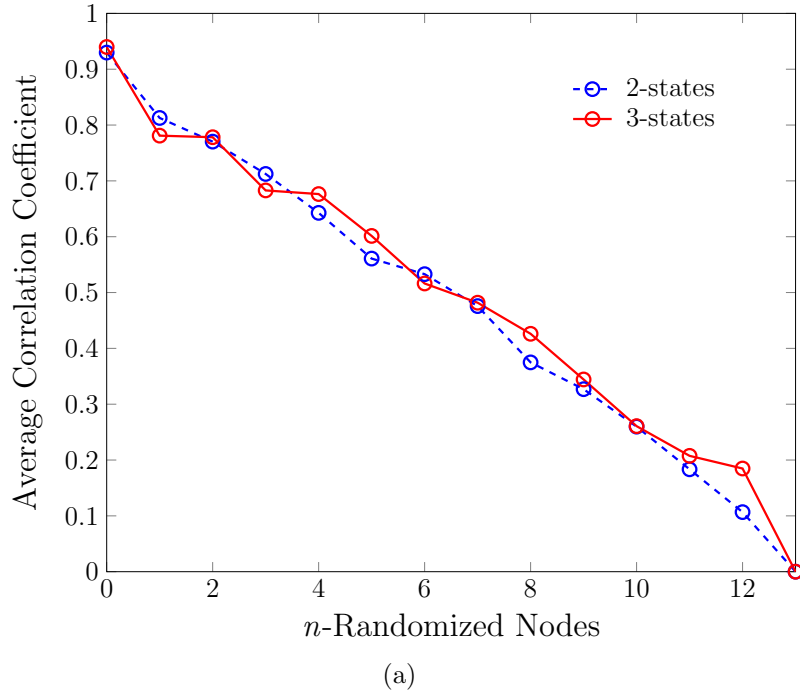
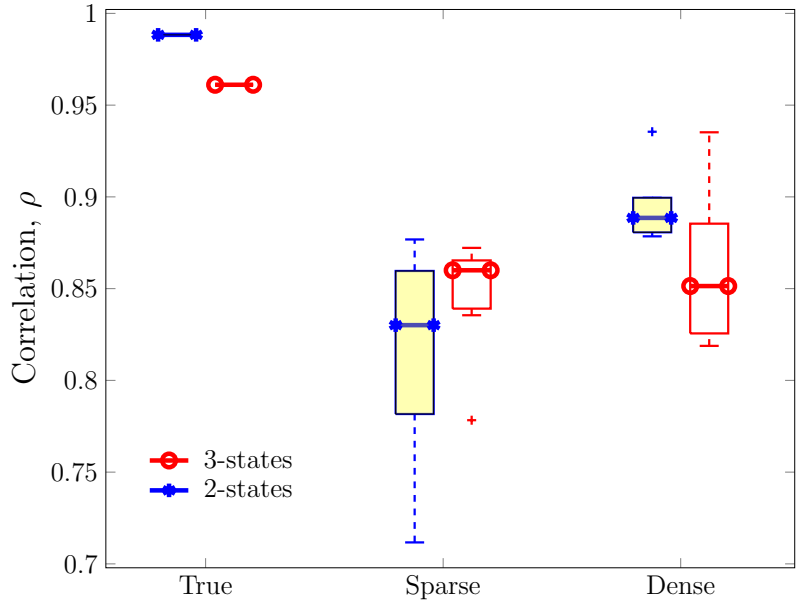
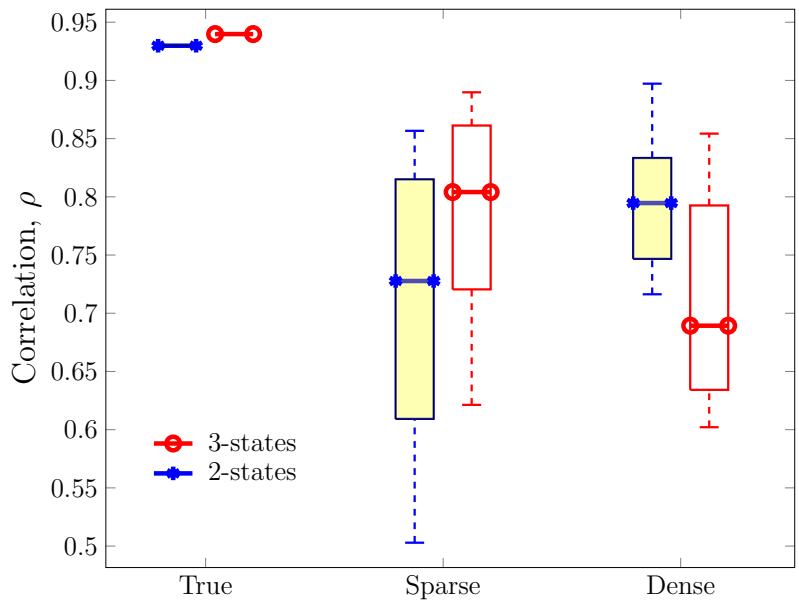


Figure 3.9: Average Pearson correlation coefficient plots of  $n$ -randomized nodes or edges for AR response network. Model predictions are repeatedly evaluated, and averages of correlation coefficients are computed in  $10 \cdot |E|$  random attempts: (a) randomly shuffled datasets where states are swapped between different variable nodes, and (b) randomly shuffled  $n$ -edges over network structure.



(a)



(b)

Figure 3.10: Evaluation of probabilistic FGN model on GRNs with different sparsity in: (a) SOS response network, and (b) AR regulatory network. Performance is compared to very sparse and dense networks for both 2-states (yellow-fill blue boxplots) and 3-states (red boxplots) discretization levels. Sparse and dense networks are obtained by deletion of edges from and addition of edges to true (original) GRNs, respectively.

(i.e., 0.9883, see Figure 3.5(a)), in sparse and dense networks, respectively. Furthermore, it is observed that in the 3-states discretization analysis, the median correlation coefficient of the true network (i.e., 0.9611) decreased by 10.5% for sparse networks and 11.4% for dense networks. Similarly, Figure 3.10(b) summarizes the model predictions in the AR regulatory network. Here, too, the median correlation coefficient of the sparse networks in 2-states discretization analysis was observed to be 0.7277 (21.7% decrease) and in 3-states discretization, 14.4% less than the true network correlation coefficient of 0.9398. Moreover, for the dense network analysis, the median correlation coefficients decreased by 14.5% and 25% in 2-states and 3-states discretization, respectively. In general, it was observed that as the number of edges added or removed increased, the average Pearson correlation coefficients deteriorated. These results suggest that the probabilistic model can robustly separate good GRNs from poor ones.

### 3.3.4 Impact of Discretization Levels

Table 3.3 illustrates the performance of the proposed probabilistic model as the number of discretization levels increases. Moreover, the corresponding average CPU runtime (this value could change depending on the processor speed of the computing machine being used.) is shown, and as expected, the network with the greater number of edges has a higher computation time. Results show a decline in predictions with a higher clustering resolution. It is noteworthy that clustering algorithms tend to be less robust with respect to the larger overlap between clusters as  $k$  increases [55]. In addition, as the number of states increased above 6 (i.e., in the case of the SOS response network) and above 9 (i.e., for the AR network), it was observed that the discretization optimization failed due to the creation of ill-conditioned covariance matrices of the GMM. This, I noted, stems from the nature of the normalized gene expression data used in this work. Certain genes, as observed, especially in the SOS response network, have a data range as low as 2.8 or have multiple data points close to each other, thus leading to instances of non-invertibility of covariance matrices. Intuitively then, an appropriate value of  $k$  is limited by the properties of the dataset.

Table 3.3: FGN model predictions on increasing number of quantization levels for SOS response network and AR regulatory network, and corresponding CPU runtimes.

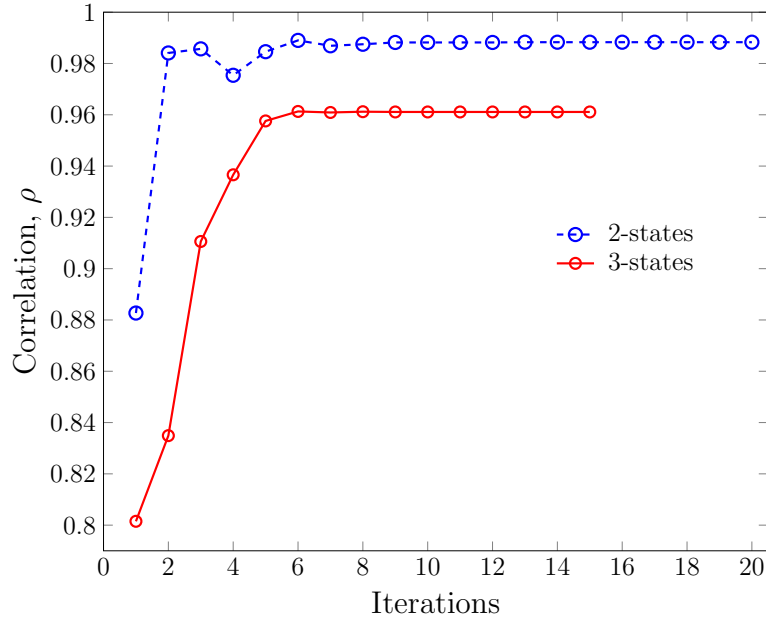
No. of States, $k =$	SOS Response Network		AR Regulatory Network	
	Approximate Correlation	CPU Runtime	Approximate Correlation	CPU Runtime
2	0.9811	0.26 s	0.9298	0.15 s
3	0.9611	0.43 s	0.9398	0.26 s
4	0.9691	8.02 s	0.9378	0.33 s
5	0.9405	13.08 s	0.8808	0.37 s
6	0.9092	19.46 s	0.8929	0.75 s
7	–	–	0.8999	1.53 s
8	–	–	0.8822	1.59 s
9	–	–	0.8737	3.38 s

### 3.3.5 Computational Complexity

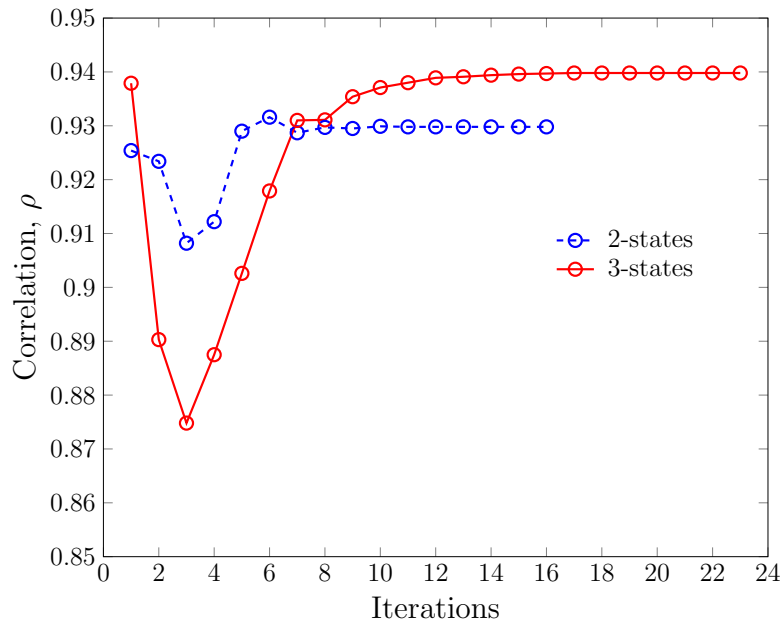
Overall, the main computational cost of running the software is the message update equation (i.e., equations (3.4) and (3.5)), which incurs  $\mathcal{O}(k^2)$  [56] per iteration for each pair of variable and factor nodes in the FGN model, if the state space of all  $g_i$  has  $k$  possible states. Recall that the message sent is a function of the parent. Also, it is assumed that the column vectors of a regulatory function  $f(:, j)$  for  $j = 0, 1, \dots, k - 1$ , and their normalization constants have been pre-computed and stored, which can be done off-line. This computational cost grows linearly with the number of edges. Figure 3.11 depicts a practical overview of the software performance and run time, where the predicted  $\rho$  is plotted against increasing iteration of the message-passing algorithm for both SOS response and AR regulatory networks.

In Figure 3.11(a), the messages converged after 20 and 11 iterations for 2- and 3-states discretization levels, respectively. Similarly, in Figure 3.11(b), the messages converged after 14 and 20 iterations. An inspection of the figures reveal that the rate of convergence is at

least linear,<sup>2</sup> which is consistent with findings in the existing literature [45].



(a)



(b)

Figure 3.11: Pearson correlation plots of LBP message-passing convergence with increasing iteration for both 2- and 3-states discretization levels in: (a) SOS response network, and (b) AR regulatory network. CPU runtime until convergence is on average 0.26s and 0.43s in SOS network for 2- and 3-states discretization, respectively. Accordingly, CPU runtime for AR network is on average 0.15s and 0.26s.

<sup>2</sup>Note that linear convergence means that the error (i.e., difference between two successive  $\rho$ 's) decreases exponentially.

### 3.4 Summary

In this chapter, I explored a probabilistic graphical model representation of biological networks and applied a message-passing algorithm to investigate the allowable stable states and consistency between some of the existing experimentally verified pathways in the *Bacterium* genome to the diverse high-throughput real biological data. The mathematical formulation of the model describes steady-state behavior of systems where the steady-state assumption is highly adequate for a typical high-throughput experimental sampling rate. Also, in its current form, the model is already capable of handling both minimal time required for spreading perturbations in the network and undelayed feedback loops. I then applied statistical analyses to compare the variability between the model-inferred steady-state marginals and the experimental observed states. The findings reveal a high correlation between the given network pathways and the diverse experiments gene-expression data. This implies that the small sub-networks considered are strongly supported by the measured gene expression data.

One major simplification that was applied in the network models is the assumption of static gene regulatory networks, which do not adequately explain transcriptional gene expression, at least not on a cellular system-wide level. In order to enable true inference of cellular functions and organization, future methods on more complex models that consider the fluidity of biological networks (i.e., changing networks with time, context, and conditions), temporality, and multi-omics data would be required. Furthermore, in this work, I applied the modeling of distributions over discrete functions, primarily since most of the current biological knowledge on regulatory relations and transcriptional switches is essentially qualitative. Noteworthy, in its current form, the model cannot predict actual non-discretized gene expression levels. However, in order to do so, the framework can readily be adapted by defining continuous probability distribution over the regulatory function,  $f_i$ . This would require much more data to adequately learn the regulatory relations toward more significant results [36].

## CHAPTER 4

### BOOLEAN FACTOR GRAPH MODEL OF BIOLOGICAL NETWORKS

#### 4.1 Introduction

In biological networks, the temporal evolution of gene or protein expressions constitutes a dynamical system. Modeling the coupled dynamics and characterization of the long-run behavior of such networks is perhaps the most important task in genomic signal processing. In the literature, long-run distribution has been conjectured to correspond to the phenotype of a cell [1]. Consequently, different analytic and computational models have been proposed to capture the behavior of complex gene regulatory networks, including differential equations [12, 14], Bayesian networks [18], and Boolean networks (BNs) [16, 57]. Among deterministic dynamical systems, perhaps the BN model has received the most significant research effort since it was introduced by Kauffman [1, 58]. BNs constitute an important class of models for regulatory networks of gene interactions, in that they are simple and capture some fundamental characteristics of gene regulations, and their rule-based structure carries physical and biological meaningful phenomena, for instance, stability, hysteresis, cellular state dynamics, and the possession of a switch-like behavior [59].

In this chapter, the goal is to predict the impact on the long-run behavior and network state progression caused by perturbation (also referred to as disturbance) of regulatory functions. The study focuses on BNs with perturbation, particularly on gene deletions and random state perturbation. Note that allowing genes to randomly flip states is biologically meaningful [60, 61].

In the literature, the dynamical properties of Boolean networks have been studied based on two fundamental types of perturbations: state and structural. In state perturbation, genes or protein expression states in the network are flipped to modulate the dynamics. State perturbation is considered temporary because it resets the initial states of the underlying deterministic rule and does not alter the network structure [61]. Hence, the network

attractors and the basins of attraction remain invariant. However, if the BN model has multiple attractors, state perturbations may cause convergence to a different attractor than the original one and may lead to a change in the steady-state distribution of the BN. State perturbations have been studied mostly by analyzing the collective behavior of a large number of random BNs [61, 62, 63]. On the other hand, structural perturbation, also referred to as functional perturbation [64, 65], has a more fundamental impact on BNs. The long-run distribution is changed permanently, or the progression of states is halted since the underlying rule-based structure is altered. As a result, functional perturbation has the potential to reverse or force the gene network to transition from undesirable stable states, which is a useful tool in developing gene therapies. Functional perturbations are less studied, and most algorithms proposed are rather cumbersome because they require the computation of transition probabilities before and after perturbations. Also, most perturbation studies employ Markov chain [66] analysis to empirically estimate the steady-state distribution of a network.

This chapter proposes a computational framework that combines the formulation of BNs [1] and factor graphs [21, 67] to investigate the global dynamical property and impact of gene knockout in regulatory networks of gene interactions. With the flexibility and genericity of factor graph formalism [21], the methods proposed here may aid in the analysis of Boolean genetic graphs using a wide range of biological rules or processes. The model is formalized as a Boolean factor graph and propose a message-passing protocol to evolve network states. The framework and structure of the proposed model can allow us to track the progress of network states. Thus, it has the potential for supporting network intervention analysis. I employed a synchronous updating scheme in the model. The synchronous update approach is chosen for simplicity; however, in reality, molecular processes or events are not coordinated in time. Also, in gene knockout analysis, the requirement for the accurate specification of time delays or of priorities that are difficult to define or may be context-sensitive obscures the implementation of an asynchronous update strategy.

The methodology is then applied to study a sample network regulating the cell cycle of the budding yeast, referred to as the Li model [23]. In their work [23], Li et al. did

not systematically analyze the effect of reported gene deletions. Simulation results on yeast cell-cycle gene deletion analyses are supported by experimental data in the literature. In addition, the findings show that the Li model is consistent with real gene-expression data.

#### 4.1.1 Related Work

Several methods have been proposed in the literature to qualitatively reproduce some known dynamical features of the wild-type biological systems, as well as the consequences of single gene deletions. Novák and Tyson [68] proposed a differential equation model and used numerical integration techniques to model the control of the restriction point of the mammalian cell cycle. However, this model appears difficult to extend. Also, Fauré et al. [69] applied a logical modeling technique to delineate the main dynamical properties of the mammalian cell cycle network. They assessed the merits and limits of synchronous network updating assumptions in BNs against asynchronous assumptions. However, in their model, the effect of each regulator depends on the presence of co-regulators. Similarly, many simulation and analysis software tools for logical models exist, including GINsim [70], BoolNet [71], bioLQM [72], and CellNetAnalyzer [73]. However, these tools rarely consider error analysis in biological networks.

Other tools of dynamical systems theory like bifurcation analysis [74] and time-scale analysis based on the sign of Jacobian eigenvalues [75] provide temporal patterns that are often comparable to experimental data, which is a real advantage. Moreover, DNA content analysis by flow cytometry [76, 77] has been employed to study the effects of single gene deletion and gene over-expression on cell cycle progression. Such models contain detailed information about time evolution of the system. However, modeling the actual time duration of cellular processes requires knowledge of a large number of biochemical parameters that are difficult to find [40]. In addition, when interest is in the prediction of the sequential pattern of states and the long-run distribution of cellular processes, the exact time course of the regulatory network dynamics may be neglected. A recent report indicates that some gene networks are so robustly designed that timing is insignificant [78].

## 4.2 Model and Methods

This section provides a background on Boolean networks as a model for representing gene regulatory networks, and factor graphs, which are required to understand the proposed model and the analysis in this dissertation. I then introduce and describe a message-passing protocol employed on the proposed model to evolve network states as messages. Finally, this section presents a brief introduction of a sample biological network used for application of the model and methods.

### 4.2.1 Boolean Networks

Formalism of the BN model underscores the fundamental generic principles rather than quantitative biochemical details, which establishes a natural framework for capturing the dynamics of gene networks and their regulatory mechanisms, yielding insights into their overall behavior. For consistency of notation with materials in the literature [57], a Boolean network  $G(V,F)$  is defined by a set of  $n$  binary-valued nodes  $V = \{x_1, \dots, x_n\}$  and a list of Boolean functions  $F = \{f_1, \dots, f_n\}$ . In systems biology, the set of nodes  $V$  could represent biological entities such as genes, mRNAs, and transcription factors (TFs). Each node  $x_i \in \{0, 1\}$  has  $k_i$  parent nodes (i.e., regulators). Also, let  $Pa_i = \{Pa_{i,1}, Pa_{i,2}, \dots, Pa_{i,k_i}\}$  denote the set of parents of  $x_i$ . For clarity, in this work, I refer to the biological entities as genes of a network. The state of  $x_i$  denotes the expression of the node quantized to only two levels. In this model,  $x_i = 1$  means that gene  $i$  is expressed (active), and  $x_i = 0$  means that it is not expressed (inactive). Whenever a gene is expressed, it could affect the expression or suppression of other genes. Therefore, the value or state of a gene at time  $t + 1$  is given deterministically by its regulators at time  $t$  through a Boolean function  $f_i \in F$  as

$$x_i(t + 1) = f_i(x_{i1}(t), x_{i2}(t), \dots, x_{ik_i}(t)) , \quad (4.1)$$

where  $\{i1, \dots, ik_i\} \subseteq \{1, \dots, n\}$ , and  $k_i$  is the connectivity of node  $x_i$ . The network function  $F$  represents the rules of regulatory interactions between the genes. Given the network state at time  $t$  as  $\mathbf{x}(t) = (x_1(t), \dots, x_n(t))$ , the state transition  $\mathbf{x}(t) \rightarrow \mathbf{x}(t + 1)$  is governed by  $F$

as  $\mathbf{x}(t+1) = F(\mathbf{x}(t))$ . In addition, if  $Pa_i = \emptyset$  (i.e.,  $x_i$  has no parents), then  $x_i(t+1) = x_i(t)$ .

The set of all possible states (i.e., state space) of the BN contains  $2^n$  network states, so that after a finite number of state transitions, the initial sequence becomes transformed into a stable sequence of zero-dimensional fixed points known as singleton attractors. In certain instances, the initial sequence may eventually transition into a set of cyclical attractors. All states that lie on trajectories flowing to an attractor comprise its *basin of attraction*. Of note, the attractors capture the long-run behavior of a dynamical system. Also, in biological systems, the key idea is to perceive each stable attractor configuration as representing one possible biological phenotype or cell type [1]. An example of a simple Boolean genetic graph with four genes is shown in Figure 4.1(a). In this network, a directed arrow edge implies an activation interaction link, while a blunt edge denotes an inhibition influence.

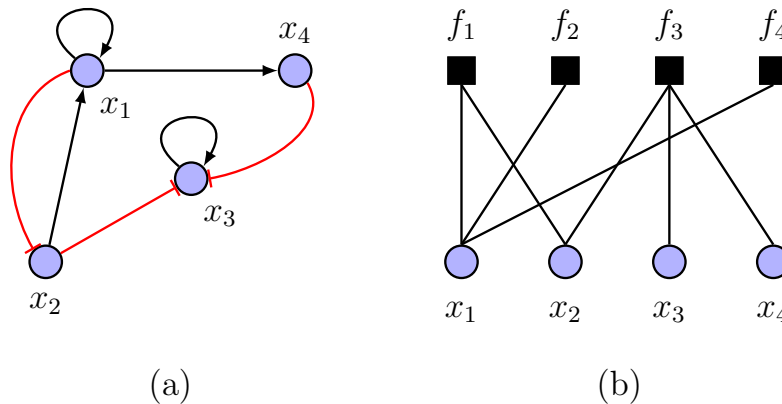


Figure 4.1: (a) Simple directed gene graph with  $n = 4$ . (b) Equivalent undirected factor graph representation of (a). The state of gene  $x_i$  at time  $t + 1$  is governed by Boolean function  $f_i$ , given a set of interacting genes. For instance,  $x_3(t + 1) = f_3(x_2(t), x_3(t), x_4(t))$ . Red blunt edges indicate inhibition links, whereas black arrowheads represent activation interactions.

Furthermore, it is presumed that any node out of the  $n$  possible nodes can get perturbed independently of other nodes. In the BN setting, network state perturbation is represented by a random flip of value from 0 to 1, or vice versa. Since the genome is an open system with external inputs, it is known that genes may become either inhibited or activated due to external stimuli. In this model, an error is introduced into the network with a positive

probability  $\epsilon \ll 1$  when the state of a node changes due to random gene perturbations.

### 4.2.2 Boolean Factor Graph Model

In biological systems, for each cell type and for each function performed by a cell, the regulatory network has a specific form that determines what biochemical processes will be executed and in what order. For instance, in Li’s yeast cell-cycle model, the stationary  $G_1$  attractor configuration has 13 state transitions once the cell has committed to division [23]. Therefore, attractors bear strong biological implications, and the question of interest is to understand their nature and properties, and how they respond to perturbation in the network. This section presents a study and analysis of the behavior of gene networks by representing them in terms of Boolean factor graphs.

Following the factor graph formalism described in Chapter 2, in this chapter, a variable node is denoted by gene  $x_i$ , whereas a factor node represents a Boolean function,  $f_i$ . Figure 4.1(b) shows the equivalent bipartite form of the Boolean gene graph in Figure 4.1(a). To convert a Boolean network into a bipartite graph, we can simply draw an edge between a variable node  $x_i$  and a factor node  $f_j$ , if the scope of  $f_j$  contains  $x_i$ . In simple terms, a gene connected to a factor node  $f_i$  exerts an influence on the operation of gene  $x_i$  within the assumption of one time unit. For instance, in Figure 4.1(b), genes  $x_2$  and  $x_4$  exert an influence on gene  $x_3$  through the Boolean function  $f_3$  following the deterministic equation (4.1). The factor graph representation is convenient and has been utilized widely in the literature but in a probabilistic setup [6, 21, 34]. In addition to the structure of the proposed model, in this section, I describe a simplified Boolean function model at the factor nodes and formulate a message-passing algorithm as an inference tool to evolve network states.

### Boolean Functions

Boolean functions consist of a set of rules specifying how a given node in a graph changes its value over time, as a function of the past or current states of its parent nodes. These functions represent the simple dynamics of inhibition and activation between interacting nodes. Martin et al. [16] used an activation-inhibition Boolean function model as

an inference algorithm to reverse engineer the regulatory network of gene interactions from microarray time series data. As an example, a simplified Boolean function is modeled that takes into consideration the current state of the regulated node  $x_i$ . It is the belief that this model, though simple, may still preserve certain biologically meaningful patterns of interactions. A similar Boolean model was employed in [69, 70], where the logical combination of interactions on a regulated node was compared to the concentration/activity level of that node to make a decision on the new concentration level.

By definition, Boolean function  $f_i$  is formulated at the factor node for a variable  $x_i$  using activation and inhibition functions depicted by the truth tables shown in Table 4.1. The activation-inhibition Boolean functions take into account the present state of the child (i.e., regulated) node. In accordance with the logical rule introduced by Li et al. [23], these Boolean functions stipulate that only when a regulator node is active does it contribute information to the child node. For instance, given two interacting nodes in a network where  $x_1$  activates  $x_2$ , the state of node  $x_2$  at time  $t + 1$  is defined by a Boolean function as  $x_2(t + 1) = x_1(t) \vee x_2(t)$ . Similarly, if  $x_1$  inhibits  $x_2$ , then  $x_2(t + 1) = (x_1(t) \oplus x_2(t)) \wedge x_2(t)$ . The logical operators  $\{\vee, \wedge, \oplus\}$  used bear the usual meanings, and all operations are in GF(2).

Table 4.1: Boolean truth tables for both activating and inhibiting gene interactions

<b>Activation</b>			<b>Inhibition</b>		
$x_1$	$x_2$	$x'_2$	$x_1$	$x_2$	$x'_2$
0	0	0	0	0	0
0	1	1	0	1	1
1	0	1	1	0	0
1	1	1	1	1	0

Note: Column  $x_1$  represents the state of a regulator node, and column  $x_2$  denotes the child node state at time  $t$ . The output state of the child node at time  $t + 1$  is denoted by column  $x'_2$ . For activation,  $x'_2 = x_1 \vee x_2$ , and for inhibition,  $x'_2 = (x_1 \oplus x_2) \wedge x_2$ . Only when a parent node is active does it contribute information to the child node.

This chapter considers simple deterministic rules for illustration of the proposed model. However, due to the simple nature of factor graphs, many other typical processes of biological systems or complex rules may easily be emulated. Such processes may include cooperative effects of active regulators, targeted inhibitions, longer activation times for certain nodes, etc. For instance, one can implement logical cooperative effects at a factor node following the activation-inhibition function proposed by Martin et al. [16] of the form  $x(t + 1) = (x_{a1}(t) \vee x_{a2}(t) \vee \dots) \wedge \neg (x_{r1}(t) \vee x_{r2}(t) \vee \dots)$ , where  $x_{a1}, x_{a2}, \dots$  are activators, and  $x_{r1}, x_{r2}, \dots$  are inhibitors or repressors acting on a node. The operator  $\neg$  denotes a logical NOT. Such activation-inhibition Boolean functions can be implemented as a single conceptual computational rule.

### Message-Passing Algorithm for Network Inference

Having formulated the Boolean functions at the factor nodes, the next step develops and describes a message-passing algorithm as an inference technique to evolve network states as messages on the Boolean factor graph. Message-passing techniques such as junction tree, sum-product, and belief propagation have been successfully employed in the decoding of codes on graphs [21, 24, 25, 26]. Similarly, since gene regulatory networks are cyclic in nature, a variant of the message-passing algorithm referred to as loopy belief propagation has been employed as an inference tool in biological systems, albeit in a probabilistic setting [19, 34].

Here, the evolution of network states begins at the variable nodes of a factor graph. At the beginning, initialize the variable nodes of the factor graph with one of the possible  $2^n$  network states. Each variable node performs no computation, but simply sends out its current state as a message to all its neighboring factor nodes, including its corresponding factor node. Formally, a message sent from a variable node,  $i$ , to a factor node,  $j$ , is denoted as  $\lambda_{ij}$ , where  $\{i, j\} \subseteq \{1, \dots, n\}$ , as shown in Figure 4.2(a). Therefore, a factor node  $f_i$  receives a set of  $k_i$  messages from its neighboring variable nodes, in addition to message  $\lambda_{ii}$ . Recall that  $k_i$  is the cardinality of regulators of gene  $x_i$ . For each message  $\lambda_{ij}$  received at

$f_i$ ,  $f_i$  computes a value (0 or 1) of what the next state of  $x_i$  should be, based on  $\lambda_{ij}$  and  $\lambda_{ii}$  using the Boolean function truth tables in Table 4.1. Then,  $f_i$  performs majority voting among these  $k_i$  values to form a *belief*  $\mu_i$  as the next state of  $x_i$ , as shown in Figure 4.2(b), and sends it to  $x_i$ . Perform the majority rule to adapt the Boolean functions in accordance with logical rules in [23]. In the next iteration or time step,  $x_i$  sends out its acquired new state. For example, consider factor node  $f_2$ . In each iteration,  $f_2$  receives messages  $\lambda_{12}$ ,  $\lambda_{22}$ , and  $\lambda_{32}$ . Using the activation Boolean function truth table,  $f_2$  computes an output value using  $\lambda_{12}$  and  $\lambda_{22}$  since node  $x_1$  activates node  $x_2$ . Similarly, since node  $x_3$  inhibits node  $x_2$ ,  $f_2$  uses the inhibition Boolean function truth table to compute an output value based on incident messages  $\lambda_{22}$  and  $\lambda_{32}$ . Then,  $f_2$  performs a majority voting over all output values to form a belief  $\mu_2$  and sends this value to node  $x_2$  as its new state.

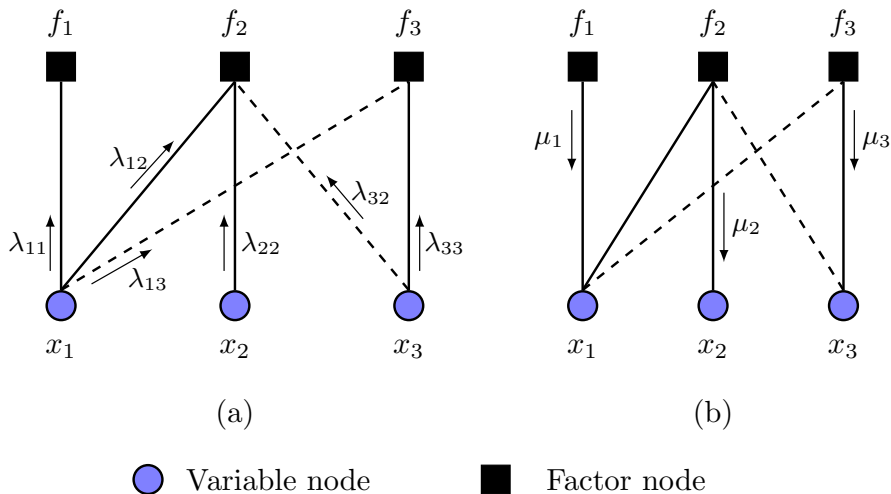


Figure 4.2: Message passing update in a sample Boolean factor graph: (a) messages sent from variable nodes to factor nodes denoted by  $\lambda$  values, and (b) messages sent from factor nodes to variable nodes given by  $\mu$  values. Solid (dashed) edges denote activation (inhibition) interactions.

In the proposed model, it is assumed that at each iteration, all nodes are synchronously updated in accordance with the regulatory rules assigned to them, and this process is then repeated. The network is said to have attained a stable sequence if, at time  $t$ , the value of variable nodes are invariant for all times  $t' \geq t$ .

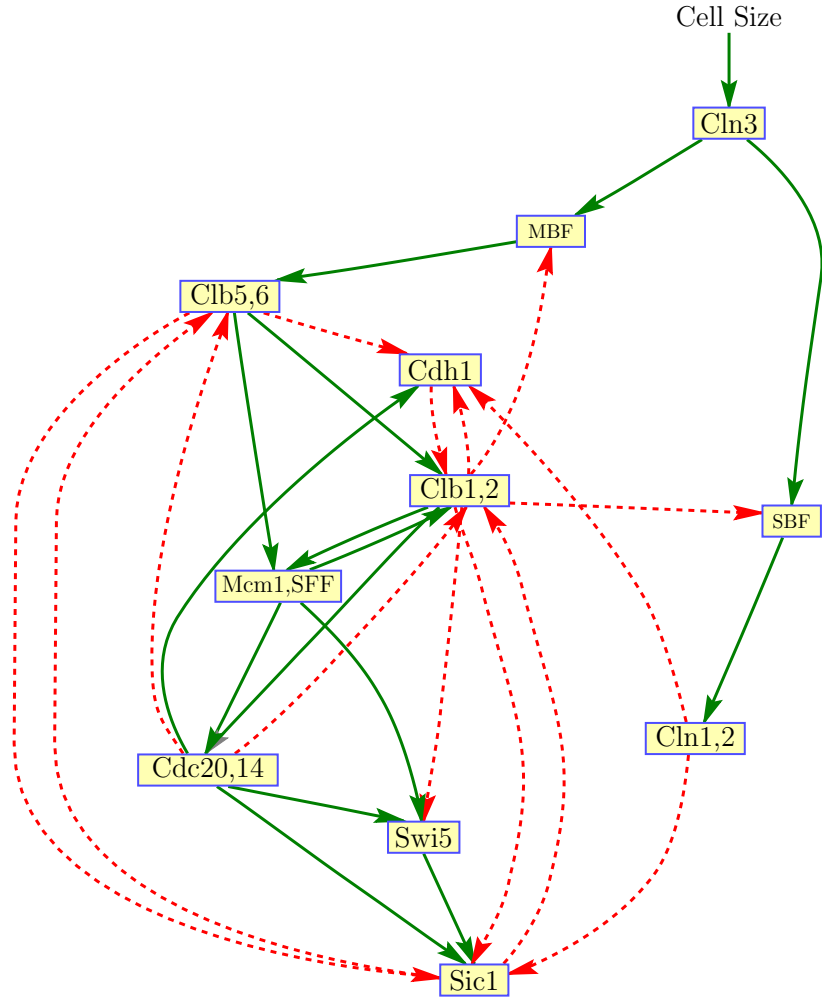


Figure 4.3: Simplified yeast cell-cycle network adapted from [23]. Solid green edges denote activation links, whereas dashed red edges represent inhibition links. Nodes Cln3, Cln1,2, Swi5, Cdc20,14, and Mcm1,SFF have self degradation. Cln3 is a “stater kinase” used to trigger the  $G_1$  - S phase transition when the cell grows sufficiently large.

### 4.2.3 Model Network: Yeast Cell Cycle

In this chapter, the yeast cell-cycle network model presented by Li et al. [23] is used as an illustrative example to demonstrate the application of the proposed model and methodologies in systems biology. This network was constructed using experimentally verified and known key regulators reported in the literature. Figure 4.3 shows the connectivity among the various nodes with corresponding interaction type. The logical network consists of 11 nodes participating in the regulatory process that controls the cell cycle in budding

yeast. This process consists of four phases:  $G_1$ , S,  $G_2$ , and M. At the  $G_1$  phase, the cell grows and commits to division under appropriate conditions. In the S phase, DNA is synthesized and chromosomes are replicated.  $G_2$  is a “gap” between S and M. The final phase, M, corresponds to mitosis, in which chromosomes are separated and the cell divides before returning to the  $G_1$  phase, thereby completing one cycle. The M phase encompasses several subphases, namely prophase, metaphase, anaphase, and telophase. In the model, nodes are classified into four classes: cyclins ( $G_1$  cyclins Cln1,2 and Cln3, and the S/M cyclins Clb1,2 and Clb5,6), inhibitors of cyclin/Cdk1 complexes (Sic1, Cdh1, Cdc20,14), transcription factors (MBF, SBF, Swi5, and Mcm1,SFF), and the checkpoint cell size. Studies on cell cycle rely mainly on cell size changes to initiate cell division at a point called “Start” in budding yeast [76, 79]. For example, when a yeast cell evaluates its growth in the late  $G_1$  phase and moves to the S phase, it commits itself to a new round of DNA synthesis and mitosis before returning to  $G_1$ .

Though not depicted in Figure 4.3, nodes Cln3, Cln1,2, Swi5, Cdc20,14, and Mcm1,SFF have self-degradation. In their work [23], Li et al. implemented self-degradation as a time-delayed interaction at the variable nodes. That is, if the state of a self-degrading node at time  $t$  is 1 and for the entire delay period  $t_d$  the states of its regulator nodes are 0, then at time  $t + t_d$ , the node will be degraded to 0. In the simulations done here,  $t_d$  is set to 1. According to [23], under a synchronous network-update scheme, the logical network in Figure 4.3 has seven singleton attractors or fixed points, shown in Table 4.2, as its basin size with the largest attractor consisting of 1,764 states ( $\approx 86\%$  of the total state space, i.e.,  $2^{11}$ ). This large fixed point is consistent with the stationary  $G_1$  phase of the cell cycle, in which the cyclin/Cdk1 inhibitors Sic1 and Cdh1 are expressed while all other nodes are inactive [12]. This is referred to as the  $G_1$  attractor. Of the seven attractors, only the  $G_1$  attractor represents an observable biological state, because under normal conditions, the cell will be sitting in this state unless perturbed. Moreover, Li and colleagues performed network perturbation by deleting or adding an interaction edge, as well as changing the activation and inhibition links. They observed that for most perturbations, the relative changes of the

basin size of the G1 attractor were small. In summary, Li et al. [23] concluded that this yeast cell-cycle logical network is robustly designed.

Table 4.2: Attractors of budding yeast cell cycle [23]

Basin Size	Genes										
	Cln3	MBF	SBF	Cln1,2	Cdh1	Swi5	Cdc20,14	Clb5,6	Sic1	Clb1,2	Mem1,SFF
1,764	0	0	0	0	1	0	0	0	1	0	0
151	0	0	1	1	0	0	0	0	0	0	0
109	0	1	0	0	1	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	1	0	0
7	0	1	0	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0

*Note:* Each stable point is represented in a row. The genes' columns show the state of a gene in the respective fixed point.

### 4.3 Applications of Boolean Factor Graph Model

This section demonstrates the performance of the proposed model by providing three use case examples, namely gene-deletion analysis, network consistency analysis, and node connectivity analysis. Then biological insights are deduced based on results and findings.

#### 4.3.1 Gene-Deletion Analysis

Gene deletion, also widely known as gene knockout (KO), is a type of perturbation on the network structure. This structural perturbation alters the connectivity or Boolean functions of the network and, as a result, may lead to changes in the functionality of a biological network. When structural changes occur, the network fixed points and basins of attraction will be impacted and subsequently its long-run behavior. These changes can be permanent unless an intervention is implemented. A salient motivation for studying structural perturbation include the following: (1) biological systems are modular, robust, and subject to uncertainties; thus, it is desirable to elucidate the effect of a small difference in network models on their dynamic behavior; (2) gene regulations have intrinsic stochasticity,

and it is of interest to predict the outcome of any change in regulation; and (3) it is important for practical use, such as design and analysis of therapeutic intervention strategies [64]. Also, gene KO analysis could lead to a knowledge of critical nodes in a network whose perturbation leads to significant functional changes in the biological system in order to reduce the network size by eliminating the redundant components.

In this section, the proposed model is employed to verify the impact of gene KOs on the yeast cell-cycle progression based on the Li model. In the literature, for biological gene KO experiments, the expression of a target protein or gene molecule is stopped by eliminating the protein-coding regions from the genome. Therefore, in this case, the factor graph model is modified accordingly by fixing the state of the target node to zero and eliminating the corresponding factor node. Also, the viability of a budding yeast cell cycle is accounted for if it is able to go through all four phases ( $G_1/S/G_2/M$ ) having 13 state transitions. Subsequently, to validate the model, I compared the model simulation findings to the published experimental observations on gene KO experiments. The simulations confirm biological results in budding yeast cell-cycle experiments. The results demonstrate that the model can possibly be used in predictive gene KO analysis.

### **Deletion of $G_1$ Stabilizers**

Deletion of all  $G_1$  stabilizers (Sic1 and Cdh1) results in inviable cells [12]. This lethality might be caused by deletion of Sic1, which creates some DNA damage checkpoint (not modeled here) that would arrest the cells in the telophase, M phase. Furthermore, deletion of either Cdh1 or Sic1 allows the cell to undergo a start. In the model, Cdh1 deletion results in a viable cell, which is consistent with the literature [80]. However, in Sic1 KO, though the mutant cell is able to replicate its DNA, it gets stuck in the telophase, as reported by [12, 81]. Also, Schwab et al. studied the degradation of mitotic cyclins in sic1 deletion yeast strains, reporting that degradation of the cyclin subunit requires inhibition of the mitotic kinase-mediated by Sic1 [82]. They further observed that sic1 deletion mutant strains were inviable. Table 4.3 presents the model simulation results of the evolution of

protein states for Sic1 KO, indicating that the cell-cycle sequence goes from the excited state and then arrests in the M phase.

Table 4.3: Temporal evolution of protein states in Sic1 gene deletion

Time	Genes											Phase
	Cln3	MBF	SBF	Cln1,2	Cdh1	Swi5	Cdc20,14	Clb5,6	Sic1	Clb1,2	Mcm1,SFF	
1	1	0	0	0	1	0	0	0	<b>0</b>	0	0	Start
2	0	1	1	0	1	0	0	0	<b>0</b>	0	0	G <sub>1</sub>
3	0	1	1	1	1	0	0	1	<b>0</b>	0	0	S
4	0	1	1	1	0	0	0	1	<b>0</b>	0	1	G <sub>2</sub>
5	0	1	1	1	0	1	1	1	<b>0</b>	1	1	G <sub>2</sub>
6	0	0	0	1	0	1	1	1	<b>0</b>	1	1	M
7	0	0	0	0	0	1	1	0	<b>0</b>	1	1	M

Note: The cell cycle gets stuck in the M phase. The “Start” phase refers to a series of linked events that prepare a cell for budding and DNA replication. Completion of the “Start” phase and commitment to a new cycle of cell division precedes the actual end of the G<sub>1</sub> phase. Bold states in the sequence rows denote the state of the deleted node. Also, the number of time steps in each phase does not reflect its actual duration.

According to a study by Hoose et al. [76], any gene deletion that changes the length of the G<sub>1</sub> phase relative to other cell-cycle phases will alter the DNA content profile. In yeast, DNA content analyses have been used to measure the effects of cell-cycle arrest when essential genes are either knocked out [77] or over-expressed [83]. In their work, Hoose et al. [76] reported that the majority of gene deletions affecting cell progression lead to a lengthened G<sub>1</sub> phase. However, they also observed that cells lacking Sic1 (Cdk inhibitor of Clb/Cdk complexes) move more quickly into the S phase. That is, the mutant cell goes through a shorter G<sub>1</sub> phase, representing premature DNA replication and genome instability [81]. Applying the model confirms this, as observed in Sic1 gene deletion. Sic1 deletion results in only one time step of the G<sub>1</sub> phase, compared to three time steps in a normal cell [23]. Furthermore, in the literature, it has been reported that Cdc20 transcription is activated in the M phase by a transcription factor complex Mcm1/SFF, which is activated in turn by Clb1,2 [12]. Thus, the activation of gene Cdc20 drives the cell progression from

the M to G<sub>1</sub> phase. According to the simulation results, deletion of Cdc20,14 blocks cells in the M phase, which is again consistent with other published reports [12].

### Deletion of G<sub>1</sub>, S, and M Cyclins

Using the proposed model, first, both Cln3 and Cln1,2 are knocked out and it is observed that the cell fails to execute “Start” and remain in the stationary G<sub>1</sub> phase, as shown in Table 4.4.

Table 4.4: Yeast cell cycle progression in Cln3 and Cln1,2 deletion

Time	Genes											Phase
	Cln3	MBF	SBF	Cln1,2	Cdh1	Swi5	Cdc20,14	Clb5,6	Sic1	Clb1,2	Mcm1,SFF	
1	<b>0</b>	0	0	<b>0</b>	1	0	0	0	1	0	0	Stationary G <sub>1</sub>

Accordingly, in the literature, deletion of all three Cln genes arrests cells in the G<sub>1</sub> phase because the start-signal facilitators are missing, and the cell is not able to bud [12]. Clb1,2 is essential for successful mitosis. The lack of Clb1,2 is lethal as the cell arrests in the G<sub>2</sub> phase, because other Cdk/cyclin complexes cannot initiate mitosis [80, 81]. This lethality underscores the key role of Clb1,2 in regulating cell-cycle events. Table 4.5 shows the model’s temporal evolution of protein states in a Clb1,2 gene-deletion simulation.

Table 4.5: Temporal evolution of protein states in Clb1,2 gene deletion

Time	Genes											Phase
	Cln3	MBF	SBF	Cln1,2	Cdh1	Swi5	Cdc20,14	Clb5,6	Sic1	Clb1,2	Mcm1,SFF	
1	1	0	0	0	1	0	0	0	1	<b>0</b>	0	Start
2	0	1	1	0	1	0	0	0	1	<b>0</b>	0	G <sub>1</sub>
3	0	1	1	1	1	0	0	0	1	<b>0</b>	0	G <sub>1</sub>
4	0	1	1	1	0	0	0	0	0	<b>0</b>	0	G <sub>1</sub>
5	0	1	1	1	0	0	0	1	0	<b>0</b>	0	S
6	0	1	1	1	0	0	0	1	0	<b>0</b>	1	S
7	0	1	1	1	0	1	1	1	0	<b>0</b>	1	G <sub>2</sub>

Note: The cell cycle arrests in the G<sub>2</sub> phase. Bold states in the sequence rows denote the state of the deleted node.

Here, too, it is seen that the Boolean factor graph model is able to confirm the impact of single gene knockouts consistent with the literature. Furthermore, from the literature, it is known that Clb5,6 is responsible for the initiation of DNA replication in the S phase [77, 80]. Similarly, in the proposed model, the absence of Clb5,6 stops cell progression into the S phase, as expected [32]. As observed, protein evolution arrests on the fourth time step of Table 4.5. The cell cannot initiate DNA synthesis and exhibits a G<sub>1</sub> arrest phenotype.

### Deletion of Transcription Factors

In the Li model, when the yeast cell size reaches a threshold, Cln3 activates SBF and MBF, the transcription factors of Cln1,2 and Clb5,6, respectively. Although the Clb5,6 level rises, it is inhibited by the G<sub>1</sub> stabilizer, Sic1. However, since Cln1,2 cannot be repressed by Sic1, it can phosphorylate Sic1, making it susceptible to degradation. Consequently, Clb5,6 becomes active and phosphorylates the second G<sub>1</sub> stabilizer, Cdh1, resulting in complete Cdh1 inactivation. In applying this model, I first deleted SBF and MBF and observed that the suppression of both of these transcription factors causes the cell to arrest before the “Start” transition, which is consistent with published materials [79]. Table 4.6 shows the model’s temporal evolution in this case.

Table 4.6: Temporal evolution of protein states in SBF and MBF TFs deletion

Time	Genes											Phase
	Cln3	MBF	SBF	Cln1,2	Cdh1	Swi5	Cdc20,14	Clb5,6	Sic1	Clb1,2	Mcm1,SFF	
1	1	0	0	0	1	0	0	0	1	0	0	Start
2	0	0	0	0	1	0	0	0	1	0	0	Stationary G <sub>1</sub>

Note: The cell cycle arrests before the “Start” transition (i.e., cells do not progress, even into the first G<sub>1</sub> phase).

Conversely, the absence of either SBF or MBF is sufficient for budding yeast cells to execute “Start,” as was also observed in the simulation. However, the cell arrests in the G<sub>1</sub> phase as shown in Table 4.7. This observation confirms published experiment reports about the two TFs. Of note, SBF is composed of two components, Swi4 and Swi6 genes, and in-

hibited by Whi5, whereas MBF is composed of Swi6 and Mbp1 genes [84, 85]. Consequently, Kraikivski et al. [79] reported that a mutant yeast cell with a single-gene deletion of either Swi4 or Mbp1 is viable.

Table 4.7: Temporal evolution of protein states in MBF gene deletion

Time	Genes											Phase
	Cln3	MBF	SBF	Cln1,2	Cdh1	Swi5	Cdc20,14	Clb5,6	Sic1	Clb1,2	Mcm1,SFF	
1	1	0	0	0	1	0	0	0	1	0	0	Start
2	0	0	1	0	1	0	0	0	1	0	0	G <sub>1</sub>
3	0	0	1	1	1	0	0	0	1	0	0	G <sub>1</sub>
4	0	0	1	1	0	0	0	0	0	0	0	G <sub>1</sub>

Note: The cell executes the “Start” but arrests in the G<sub>1</sub> phase.

### 4.3.2 Gene Network Consistency Analysis

Another application of the proposed model is to test the consistency of the existing biological networks against real gene-expression data, that is, to quantify how well a network is supported by data. As an example, let us consider the Li’s model yeast cell-cycle network. It is my understanding that this kind of consistency analysis on the yeast model has not been carried out in the literature. A dataset of uniformly normalized expression profiles was obtained from the M<sup>3D</sup> database [51]. This compendium data provides a bulk download of human-curated, computable experimental metadata and computer-validated data for integrity. Compendium data used on the yeast genome (version 3 build 2) contain 904 microarray profiles collected under a wide range of experimental conditions, including wild-type, gene(s) deletion, varying oxygen concentrations, fermentation, sporulation, different media, etc. For the analysis in this section, discretized gene-expression data is employed.

#### Discretization of Data

In this section, I used `fitgmdist` function in MATLAB to model the relations between the continuous observations on a gene and its discrete logical state. `fitgmdist` implements the iterative EM learning algorithm to fit a mixture of Gaussian models to data. By default,

`fitgmdist` implements the `k-means++` algorithm for initialization to choose `k` initial cluster centers. Here, a Gaussian component corresponds to a specific logical state of a gene, and `k = 2` is used. Also, note that each node’s state may designate a different range of gene-expression levels defined by the estimated parameters (i.e., mixture proportions, mean, and variance statistics) of the Gaussian mixture model on each node.

For each discretization of gene data, the EM algorithm is repeated ten times using a new set of initial cluster values and a maximum number of 1,000 iterations allowed. Then, the Bayes information criterion (BIC) score of the discretization model is computed to quantify how good the gene expressions fit with the mixture of two Gaussian models. A likelihood-based measure of model fit to compare multiple models fit to the same data is  $BIC = 2 * N \log L + p * \log(n)$ , where  $N \log L$  is the negative loglikelihood,  $n$  is the number of observations, and  $p$  is the number of estimated parameters specified as a numeric vector of length `k`. The model with the lowest BIC score is the best fitting model. Table 4.8 shows BIC measures of the gene-specific discretization for `k = 1, \dots, 5`.

Table 4.8: Bayes information criterion (BIC) measure of Gaussian mixture model discretization used to fit gene-expression data for different `k` number of components.

Genes	BIC Scores ( $1 \times 10^3$ )				
	k = 1	k = 2	k = 3	k = 4	k = 5
Cln3	2.6183	2.6020	2.6193	2.6397	2.6600
MBF	2.0492	1.8838	1.8913	1.9039	1.9182
SBF	2.0083	1.9608	1.9640	1.9781	1.9901
Cln1,2	2.6117	2.3853	2.3923	2.4117	2.4318
Cdh1	1.7429	1.4212	1.4066	1.4234	1.4404
Swi5	2.3690	2.1339	2.1354	2.1468	2.1660
Cdc20,14	1.7202	1.6175	1.6211	1.6326	1.6525
Clb5,6	2.0188	1.9799	1.9333	1.9374	1.9499
Sic1	1.4909	1.4785	1.4824	1.5027	1.5232
Clb1,2	2.5741	2.5267	2.5218	2.5330	2.5470
Mcm1,SFF	1.5210	1.4607	1.4757	1.4957	1.5123

Note: The lowest BIC value is the best fitting model, as highlighted in yellow.

Except for *Cdh1*, *Clb5,6*, and *Clb1,2* gene-expression data that have  $k = 3$  as the best model fit, the remainder of the genes have  $k = 2$  as the optimal mixtures of Gaussian model distribution. Therefore, I used only  $k = 2$  logical states  $\{0, 1\}$  for all nodes, corresponding to the up-regulation (1) and down-regulation (0) of genes, in conformity with Boolean models.

## Results

First, verification of the proposed methodology accurately reproduced the attractors distribution as reported by Li et al. [23], with the largest fixed point attracting 86% of the 2048 initial states. Then, the proposed Boolean factor graph model was employed to study the state evolution of the discretized gene-expression data. From the data, there are a total of 904 initial states. That is, each experimental observation point equals a state sequence in the 11-node logical network. Starting from each of the 904 initial states, all of these states eventually evolved into one of the two fixed points shown in Table 4.9. Remarkably, using real biological data, the  $G_1$  attractor is the largest fixed point, attracting 822 ( $\approx 90.9\%$ ) of the 904 initial states.

Table 4.9: Attractors of cell cycle on real biological data

Basin Size	Genes										
	<i>Cln3</i>	<i>MBF</i>	<i>SBF</i>	<i>Cln1,2</i>	<i>Cdh1</i>	<i>Swi5</i>	<i>Cdc20,14</i>	<i>Clb5,6</i>	<i>Sic1</i>	<i>Clb1,2</i>	<i>Mcm1,SFF</i>
822	0	0	0	0	1	0	0	0	1	0	0
82	0	1	0	0	1	0	0	0	1	0	0

Note: Each stable point is represented in a row. The genes' columns show the state of a gene in the respective fixed point.

Additionally, I implemented a similar discretization scheme as above; however, a random sampling is employed to select  $k$  initial cluster centers. Under this discretization scheme, it was observed that the initial states eventually flow into the two fixed points shown in Table 4.9 with the same  $G_1$  attractor, attracting 91.92% of states. As expected, the percentage of states in the  $G_1$  attractor using real gene-expression data is comparably and above the 86% wild-type basin size. Based on these results, it is sufficient to conclude

that even under diverse experimental conditions, the stability of the cell state is guaranteed. Thus, one can consider the basin of attraction of the  $G_1$  attractor as the allowable states that the cell can assume under external influence or stimuli. Once the stimuli are removed, the cell flows back to the stationary state. In addition, these findings may imply that the Li model is consistent with the microarray data obtained from various biological experiments. Of note, if there was a disparity between the data and the network considered, then one can expect to see more initial states transitioning to other attractors than to the  $G_1$  attractor.

### 4.3.3 Node Connectivity Analysis

In this section, the proposed Boolean factor graph model is applied to study the impact of node connectivity in biological networks and analytically characterize the dynamics of error propagation and recovery in Boolean gene networks. It is assumed that an initial random state perturbation introduces an error in the Boolean network. A state perturbation may result from either environmental or biological fluctuations that affect cellular decisions in gene networks. For the analysis, let us consider random networks with given degree distributions as models of genetic graphs. This enables us to capture biological networks with arbitrary degree distributions. Of note, the degree of a node in a factor graph is the number of edges incident to it. For an ensemble of random Boolean networks, it is practical use polynomials to represent the degree distributions of the networks as

$$\rho(x) = \sum_{j \geq 1} \rho_j x^j, \quad (4.2)$$

where  $\rho_j$  denotes the fraction of edges incident to a factor node with degree  $j$ , constrained to

$$\sum_{j \geq 1} \rho_j = 1. \quad (4.3)$$

To demonstrate how node connectivity would influence the stability of a biological network, the study is extended to consider an ensemble of random networks with ten nodes. For each experiment, at least 50 networks are sampled with factor nodes having irregular

degree distributions to mimic real biological networks, and set  $\lfloor j \rfloor \in \{1, 2, \dots, 8\}$  to denote the average connectivity of the network. Moreover, each network has random types of edge influence (activation and inhibition). Starting from each of the  $2^{10}$  initial states, allow the states to evolve and then count the number of resultant fixed points. Figure 4.4 shows boxplots of the resultant number of network attractors with increasing average factor node degree,  $\lfloor j \rfloor$ .

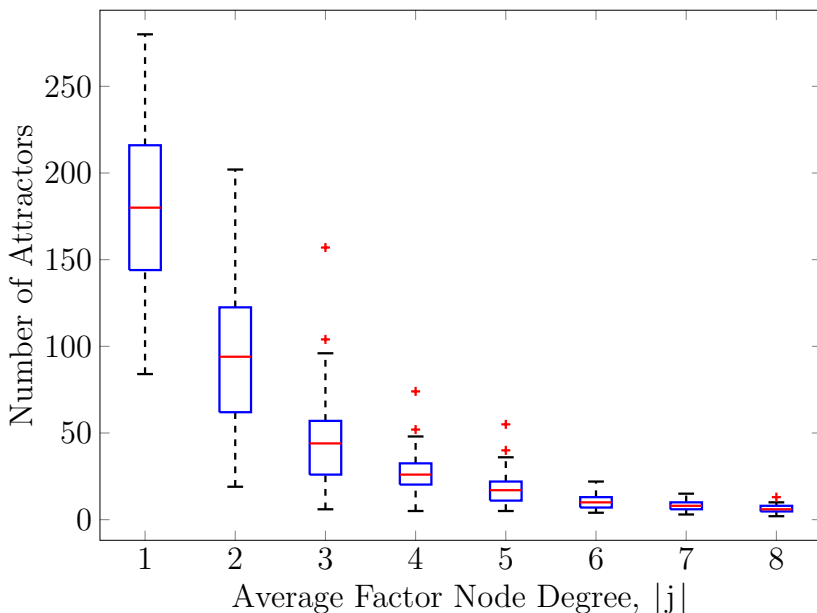


Figure 4.4: Boxplots of the number of attractors of random biological networks with increasing average connectivity. Each network has ten nodes with irregular factor node degree distribution, and  $\lfloor j \rfloor$  denotes the largest integer that is less than or equal to  $j$ . The median of the boxplots follows a power-law distribution.

It was observed that the number of attractors, as depicted by the median of the boxplots, has a power-law distribution. Based on this observation, we can gain useful insights into the nature of biological graphs. With increasing average node degree  $j$ , the basin size of the stationary attractor increases. Consequently, the homeostatic stability of a cell increases monotonically. Furthermore, for the ten-node random networks considered, as  $j$  becomes greater than 6, there are instances of both singleton and cycle attractors. This is interesting since in the literature it has been reported that large-scale or highly interconnected

networks converge into a complex attractor where the system irregularly oscillates among a set of states, especially when an asynchronous update scheme is employed [86]. From the results, one can conclude that in biological Boolean networks, nodes with a higher degree of connections are likely the key contributors to the presence of attractor cycles. Similarly, biological networks where  $\rho(x) = x$  are basically unstable, and any error caused by a random disturbance on a node cannot be corrected unless the node is self-regulating.

## 4.4 Performance Analysis

### 4.4.1 Models Comparison

This section provides a qualitative and quantitative comparative analysis of traditional Boolean approaches used to analyze budding yeast as reported in the literature and as given by the model’s simulation output. The proposed model is compared to GINsim [70] and BoolNet [71] software tools that are based on logical formalism.

#### GINsim model

In their work, Fauré and Thieffry [87] employed the GINsim model [70] to conduct a comparative study of logical models of cell cycle control in eukaryotes. For example, the authors encoded and adapted Li’s budding yeast model [23] by transcribing the logical rules into the GINsim model. Although the global topology of the logical network is preserved, the authors introduced positive feedback loops on several nodes, namely MBF, SBF, Clb5,6, Clb1,2, Cdh1, and Sic1. In contrast, the self-degradation loops seen in Li’s model were eliminated (Figure 1, top left in [87]).

Subsequently, in the analysis of the functionality of regulatory circuits of the resultant network model, Fauré et al. [87] deduced that the positive self-activating loops help in the maintenance of alternative, artefactual stable states. Using proper logical rules and employing a synchronous update scheme, the authors observed that all trajectories in the state space of the revised network converge towards a single stable state corresponding to the  $G_1$  attractor. Therefore, the model’s finding is consistent with the GINsim analysis of the yeast cell cycle model as reported in the literature, in particular regarding the dominant

stable state of the logical network.

According to result deductions, the GINsim model does not readily allow for the implementation of a majority voting rule thus obscuring a direct comparison with the proposed model. Besides, the model can provide non-binary logical analysis by defining appropriate non-binary logical functions at the factor nodes and employing a non-binary message-passing algorithm.

### **BoolNet Model**

Here, the BoolNet model [71] is employed to provide a comparative study of the dynamical behavior of Li's logical network. Using the **BoolNet** package in the R environment, the logical network is transposed as a text file containing temporal elements and encoded it in a symbolic form, i.e., as expression trees [71]. I then implemented a majority voting rule on the network nodes using the `maj()` command available in the **BoolNet** package. Also, time delays were used to transcribe self-degradation loops in the resultant logical network. However, according to model evaluation, the BoolNet model does not take into consideration the current state of the regulated node in deciding the next state of the node. Incorporating the current state of the regulated node in the BoolNet model creates a self-regulating loop. This hindered the full implementation of the model using **BoolNet**.

Identification of stable states in the resultant logical model resulted in three attractors consisting of one single attractor and two simple cycle attractors having two network states. The single attractor corresponds to the  $G_1$  attractor and has a basin of 1,472 states, or approximately 71.88% of initial states. The cycle attractors are composed of the following states: (1) {00001101110, 00000000001} and (2) {00000011010, 00000000011}, corresponding to a basin size of 370 states and 206 states, respectively. States of genes are encoded in the following order: Cln3, MBF, SBF, Cln1,2, Cdh1, Swi5, Cdc20,14, Clb5,6, Sic1, Clb1,2, and Mcm1. Except for the observed  $G_1$  attractor, the presence of cycle attractors does not match the observations made by Li et al. [23]. One may consider the two cycle attractors as spurious limit cycles. In summary, the flexibility of factor graph formalism can allow us to

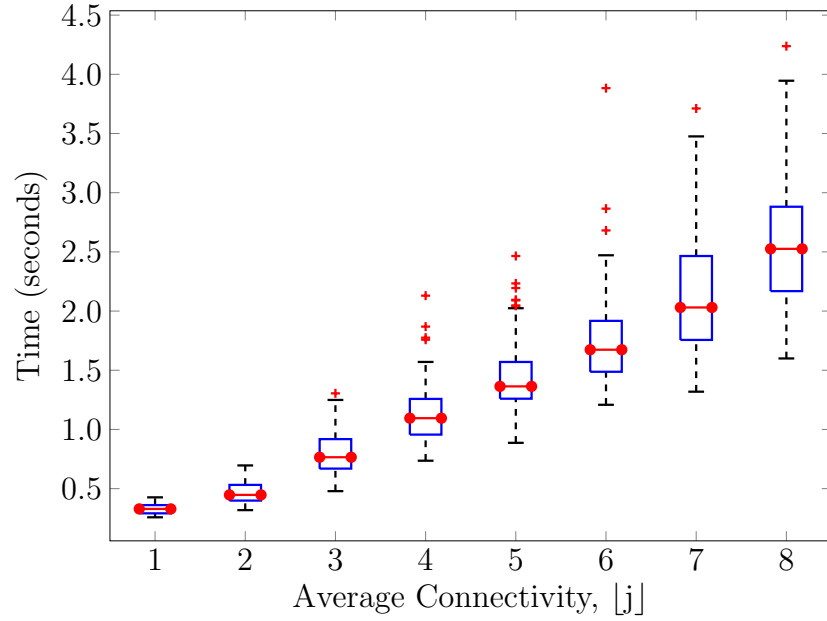
implement certain biological processes and decisions that would otherwise be neglected by traditional Boolean approaches.

#### 4.4.2 Computation Cost

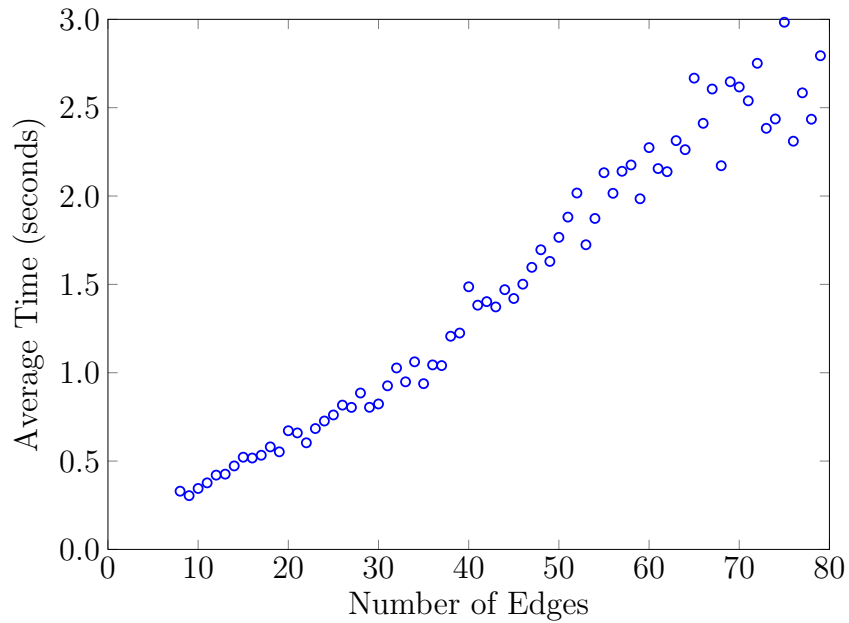
This section describes the performance and simulation analysis using random Boolean networks of ten nodes to illustrate the computational cost of the proposed methodology. Since the factor graph representation of a network preserves the network complexity [21], the main computational cost of running the proposed methodology is the network update strategy using the proposed message-passing model. Figure 4.5 shows the performance of the proposed methodology in terms of the computational cost in searching the global attractors in a network. The search depicts a linear time computation with the average connectivity of factor nodes or with the total number of edges in the network, as illustrated in Figures 4.5(a) and 4.5(b), respectively. This observation is consistent with findings in the literature regarding the computational complexity of message-passing algorithms in factor graphs where the computational cost grows linearly with the average degree of nodes times the number of nodes [25].

To describe an empirical estimation of the proposed model’s computational complexity, it is assumed that the entire network state space has been pre-computed and stored, which can be done offline. As such, searching the global attractors incurs  $\mathcal{O}(2^n)$ , where  $n$  denotes the number of network nodes. According to message-passing algorithm, the computation at the factor nodes occur in parallel. Therefore, in each network update or iteration, a factor node performs  $k_i$  Boolean computations. Recall that  $k_i$  corresponds to the number of edges between parent-child nodes. Given that each Boolean computation has a constant complexity of  $\mathcal{O}(1)$ , the cost of performing  $k_i$  Boolean computations is bounded by  $\mathcal{O}(k_i)$ . Moreover, the cost of computing the majority vote over  $k_i$  values is  $\mathcal{O}(k_i)$ . Thus, the total time complexity of a factor node is  $\mathcal{O}(k_i)$ .

On the other hand, a variable node in the proposed model simply sends out the value of its current state. This incurs a constant time complexity of  $\mathcal{O}(1)$ . Therefore, the overall time complexity of running the proposed model is  $\mathcal{O}(k_i)$  per iteration for each pair of variable



(a)



(b)

Figure 4.5: Time to evaluate the global attractors in random Boolean networks of ten nodes with respect to: (a) average connectivity of factor nodes, and (b) total number of edges in the network. The running cost depicts a linear performance with the graph connectivity.

and factor nodes in searching the attractor of an initial state. Moreover, in message-passing algorithms, the number of iterations is always limited when the algorithm converges. Hence, for a constant number of iterations, the complexity of the model is proportional to the total number of edges in the graph, i.e., of order  $\mathcal{O}(\sum_{i=1}^n k_i)$ . This is a linear time complexity and is consistent with the results in Figure 4.5. For large biological networks, e.g., genome-wide regulatory networks, where the node connectivity is sparsely distributed, the complexity is linear in the number of nodes.

## 4.5 Discussion

Identification of all attractors in a biological network is one of the key aspects in understanding the nature and dynamics of a biological system. In the literature, attractors have been found to fall into three groups, namely singletons, simple or limit cycles, and complex attractors [86]. For BNs of moderate size, i.e., networks with less than 20 nodes such as the illustrative Li model used, the proposed model and methods can allow us to identify the attractors from the initial network states without the need for using a parallelized algorithm to reduce the computation time. However, as the network size increases beyond 20 nodes, the number of initial states to test grows exponentially. One can go around this limitation by specifying a subset of nodes in which all combinations are tested as reported by Irurzun-Arana et al. [86], or using a heuristic search starting from a number of predefined or randomly chosen states [71]. Similarly, other works in the literature also indicate that network reduction methods [88, 89, 90] can be employed to handle the analysis of large models [70].

Here, the proposed model and methods rely on simulations or enumerations of states to identify network global attractors. Thus, the model incurs a computational cost of evaluating state transitions *online* compared to some classical Boolean models such as GINsim and BoolNet that enumerate all state transition graphs or tables *offline* before identifying the network attractors. Note that the proposed approach may increase the computation cost, in particular when an extensive attractor search for a large network model is required. However, message passing can allow us to access and explore the dynamics of the interactions

in a network after perturbations. Also, it provides a step towards understanding the impact of perturbations and how they propagate in the network [91].

Furthermore, by employing network tools such as connectivity analysis, we can gain insights for characterizing the resilience of biological networks to perturbations. Moreover, the observations made support the conclusions by Davidich and Bornholdt [92], that simple Boolean function models can provide a means to reproduce and predict some biologically relevant dynamic features and network perturbation effects without full knowledge of biochemical kinetic parameters. However, these simplified models do not in any way render the precise dynamical models useless. Precise dynamical rules have a real advantage of modeling biological systems more accurately, albeit at an increased computational cost.

Despite their limitations and simple nature, Boolean networks have proven to be effective for qualitatively explaining the dynamics of biological systems. For instance, BN models have been found useful for the analysis of large-scale dynamic systems in which a detailed kinetic characterization is not feasible due to either limited knowledge or data restrictions. Though not covered in this dissertation, gene over-expression can be implemented using the proposed methodology by fixing the state of a particular node in a network to a value of one.

## 4.6 Summary

Computational models have been increasingly used to deduce and understand the nature of molecular interactions in biological systems and are widely accepted by the scientific community. In this chapter, it has been demonstrated that complex biological systems can be encoded into mathematical models. In particular, I explored a Boolean factor graph model representation of biological networks and applied a message-passing algorithm to study and analyze the behavior of genetic graphs as well as to predict the consequences of structural perturbations in biological networks.

Additionally, the validity of the proposed model was verified to characterize the dynamics of the yeast cell cycle and the consequences of gene deletion. For the simplified Li model sample network used, the Boolean factor graph model is able to capture the high-level dynamics of protein states, which is consistent with other published reports in the literature.

Simulation results shown that even in a larger cell-cycle network with multiple interactions and components performing similar functions, one can expect to infer fine details on how structural changes in a network affect its long-run dynamics. In addition, from the results it can be deduced that the yeast cell cycle is not only robust [23] but remains stable under diverse experimental conditions.

## CHAPTER 5

### ERROR PROPAGATION ANALYSIS IN BIOLOGICAL NETWORKS

#### 5.1 Introduction

The execution of biological functions relies on faithful signal propagation from one gene to another. The execution process may be hindered by biological fluctuations that introduce noise or variability in gene expressions. In the literature, noise is considered a fundamental, inherent aspect of gene expression, having a diverse functional role in biological processes [60]. This noise can emanate from either environmental or biological fluctuations due to small changes in external stimuli such as pH changes, mutagens, heat stress, etc. [93]. According to Wang and Zhang [94], variability or randomness in gene expression can propagate to a higher level of biological organization and present hindrances to proper biological functions such as decision-making, development, spatiotemporal population dynamics, and reaching optimal fitness. In addition, it poses entropy-increasing effects of limiting signal fidelity, robustness, and channel capacity of signaling relays. For example, in genomic signal processing, genetic switches that control cellular decisions can flip under fluctuations that bring the biological system close to the threshold for a phase transition [60].

Despite the hindrances introduced by gene expression variation, studies indicate that fluctuations can be biologically meaningful [60, 95, 96, 97]. Variability plays an important role in natural resistance to harmful chemicals [96], and has been found useful for balancing precision and diversity in eukaryotic gene expression [97]. In addition, stochasticity in biology becomes a benefit when it is used to generate novelty at higher levels of organization [95, 98].

The majority of studies on biological noise are concerned mainly with elucidating sources of gene expression noise [93], understanding mechanisms and effects of expression variations in biological systems [99, 100, 101, 102, 103, 104], and characterizing noise properties using small-noise approximations [105, 106]. However, little research emphasis has been placed on the study and quantification of noise propagation in biological systems. Noise

propagation is considered important in assessing the information capacity of a regulatory interaction, i.e., the number of distinct stable states of a target gene expression level that can be achieved by varying the concentration of a transcription factor [107]. Knowledge of noise propagation can enable us to develop strategies that enhance the performance of downstream cascades in many biological applications. For instance, to engineer predictable behavior through synthetic gene networks, one must develop reliable means for connecting smaller functional artificial gene circuits to realize predictable high-order networks.

In this chapter, noise propagation in regulatory networks of gene interactions is studied and quantified by applying a messages-passing algorithm that evolves network states. For analytical purposes, I consider noise to introduce errors randomly on the state of nodes in the network. Then, I explored Boolean network (BN) model representation of biological networks where nodes represent biological entities such as proteins, mRNAs, genes, etc., and the edges represent the types of interaction, either inhibiting or activating, between nodes [1, 16]. BNs are simple and have proven to effectively and qualitatively explain some fundamental characteristics, e.g., steady-state behavior, of biological networks.

To analyze error propagation in gene networks, the BN model representation of a biological network is transformed into an equivalent factor graph or bipartite graph following the formulation introduced in Chapter 4. Subsequently, I applied a message-passing algorithm as an inference technique to the factor graph model, where messages represent the states of the nodes plus the error function in the network. Then, I employed the concept of *density evolution* used in the performance analysis of factor graphs [32, 33] to characterize error propagation in biological networks after an initial disturbance. The resultant closed-form recursive formula derived is referred to as the “density evolution equation.” Based on the DE equation, I derived a necessary condition for network parameters to guarantee vanishing errors or optimize the design of a biological system for high signal fidelity.

### 5.1.1 Related Work

Pedraza and van Oudenaarden [108] quantified how noise propagates in gene networks by measuring expression correlations between genes in single cells. They considered a simple

synthetic gene network consisting of only four genes. Their approach provides a step toward understanding noise propagation; however, no rigorous analysis or how the method can be extended into more complex biological networks is provided. In another work, Arola-Fernández et al. [109], applied error propagation to estimate uncertainty in the critical threshold for some dynamical processes in complex networks with noisy links. In their work, they considered network noise resulting from experimental errors such as device accuracy, sampling biases, or data entry. However, they did not analyze errors emanating from gene expression variations.

In biological networks, feedback loops are common and are known to have a critical role in cellular signaling [110, 111]. Zhang et al. [110] employed frequency domain analysis [111, 112] to investigate the role of feedback loops in sensitivity and noise amplification on the dynamical behavior of a biological system. They observed that interlinked positive and negative feedback loops dynamically tune noise propagation signals rather than monotonically suppressing or amplifying them, as would be expected in single feedback loops. However, their biochemical and kinetic network model demands more knowledge, namely kinetics of individual processes as well as deduction of signal sensitivity and noise-amplification parameters. Information on kinetic parameters can be extracted from the literature; however, this is difficult to find [40].

## 5.2 Density Evolution Analysis on Gene Networks

This section presents a Boolean network model and its equivalent factor graph model in biological systems. In a quest to study error correction in biological systems, Milenkovic and Vasic [113] established a direct relationship between BN models of gene regulatory networks and bipartite graphs used in decoding algorithms in coding theory [31]. This relationship stems from a key experimental observation in that biological networks have sparsely distributed and possibly long edges [1]. Given an initial disturbance due to noise in the system, a recursive DE equation is derived that numerically characterizes network parameters for resiliency and robustness against network disturbance.

### 5.2.1 Network Model

For the analysis, let us consider a Boolean gene regulatory graph defined by a set of  $n$  binary-valued nodes  $\{x_1, \dots, x_n\}$  representing biological entities, and a list of Boolean functions  $\{f_1, \dots, f_n\}$  denoting the rules of regulatory interaction between the nodes (i.e., genes). Each  $x_i \in \{0, 1\}$ , where  $i = 1, \dots, n$  has  $k_i$  nodes assigned to it, which completely determines its value at time  $t + 1$  following  $f_i$ , that is,

$$x_i(t + 1) = f_i(x_{i1}(t), x_{i2}(t), \dots, x_{ik_i}(t)), \quad (5.1)$$

where  $\{i1, \dots, ik_i\}$  is the index set of variables that are connected to node  $x_i$ . The state of  $x_i$  denotes the expression of the gene, where  $x_i = 1$  indicates that the gene is active (expressed), and  $x_i = 0$  means the gene is inactive. Here, I assumed all genes update synchronously in accordance with the  $f_i$ 's assigned to them, and the process is repeated iteratively in time.

In this chapter, I considered two types of activation-inhibition Boolean functions proposed in the literature [16, 69]. First, I considered Boolean functions that encode majority voting rule (employed in Chapter 4) to determine the next state of a node in the network. The majority rule is simple; however, it is less likely used in many biological models that often rely on complex logical rules to account for biological processes such as cooperative effects, selective inhibitions, and dual interactions. Second, I focused on Boolean functions that encode complex biological rules or processes such as the cooperative effect of the interacting genes of the form  $x(t + 1) = (x_{a1}(t) \vee x_{a2}(t) \vee \dots) \wedge \neg (x_{r1}(t) \vee x_{r2}(t) \vee \dots)$ , where  $x_{a1}, x_{a2}, \dots$  are activators, and  $x_{r1}, x_{r2}, \dots$  are inhibitors or repressors acting on the node. The logical operators  $\{\vee, \wedge, \text{and } \neg\}$  bear the usual meanings. An example of a gene network with three genes and its equivalent bipartite graph is shown in Figure 5.1. The state of a gene  $x_i$  at time  $t + 1$ , for instance,  $x_2(t + 1) = f_2(x_1(t), x_3(t))$ , is given by  $x_1(t) \wedge \neg x_3(t)$ .

To evolve network states on the factor graph, a factor node receives input from its neighboring variable nodes, computes an output according to equation (5.1), and passes it to its corresponding variable node. In the subsequent iteration, a variable node simply sends its current state to its neighboring factor nodes, i.e., a variable node performs no computation

on its received message. The network states evolve iteratively until a stable state is attained.

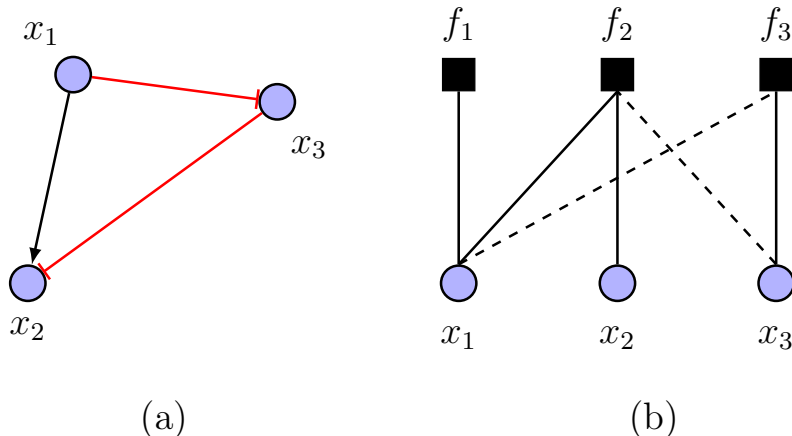


Figure 5.1: (a) Simple directed gene graph, and (b) equivalent undirected factor graph representation of (a). Red blunt edges in (a) and black dashed edges in (b) indicate inhibition links, whereas black arrowheads represent activation interactions.

### 5.2.2 Error Analysis I

In the literature, the performance of graphical models such as factor graphs depends on the degree distributions of their nodes on the graph [25, 33, 35]. In message-passing algorithms employed on factor graphs, DE refers to tracking the evolution of the probability density function of error messages between variable nodes and factor nodes. Here, for the first time, DE analysis is applied to study biological networks. I hypothesized that DE can be used to provide an exact analytic characterization of the impact on the cell attractors caused by state and/or structural perturbations. The result is a closed-form formula referred to as a “DE equation.” In this chapter, I derived and employed the DE equation to provide numerical and analytical investigation of random state perturbations on the resiliency and robustness of biological networks.

For Boolean networks with perturbation, an error is introduced with a positive probability  $\epsilon \ll 1$  by which the state of a node is randomly changed. Implicitly, it is assumed that there is an independent identically distributed (i.i.d.) random perturbation over the variable nodes in the graph. Based on the proposed Boolean factor graph model in Chapter 4

and majority voting rule, the evolution of this perturbation is tracked following the message-passing protocol established in Section 4.2, for both activation and inhibition interactions. If this probability decreases at the end of each iteration, then the network will attain the  $G_1$  attractor, whereas if this probability increases, then spurious attractors will be obtained.

Figure 5.2 shows the evolution of random errors in a three-gene Boolean factor graph model. For the analysis, I first considered the activation interaction link, whereby gene  $x_1$  activates  $x_2$ . Let messages  $\lambda_{12}$  and  $\lambda_{22}$  denote the states of variable nodes  $x_1$  and  $x_2$  sent to the factor node  $f_2$  in the  $(l - 1)$ -th iteration of the message-passing procedure. Also, let  $\beta_1$  and  $\beta_2$  be events whereby messages  $\lambda_{12}$  and  $\lambda_{22}$  have occurred in error, respectively, as shown in Figure 5.2. In addition,  $\beta'$  is the event that there is an error in the output message of factor node  $f_2$  to variable node  $x_2$ . It is further assumed that perturbations that introduce errors occur with equal probability on any variable node.

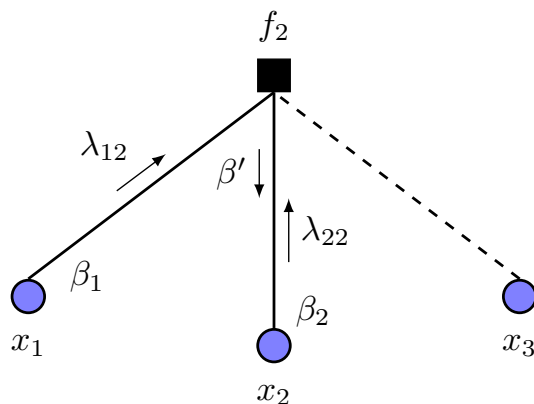


Figure 5.2: Message passing in Boolean factor graph with state perturbation probability  $\epsilon$ .

Gene  $x_1$  ( $x_3$ ) activates (inhibits) gene  $x_2$ .  $\beta$  is an event that a gene is perturbed. The dashed edge denotes inhibition interaction. Messages  $\lambda$  are passed between variable nodes and factor nodes.

Therefore, using the Boolean function truth tables in Table 4.1, the probability that there is an error in the  $l$ -th iteration of node  $x_2$  can be described in terms of the  $(l - 1)$ -th

iteration (i.e.,  $\epsilon_l = f(\epsilon_{l-1})$ ) as

$$\begin{aligned}
\epsilon_l &= p(\beta'|\beta_1, \bar{\beta}_2) \cdot p(\beta_1, \bar{\beta}_2) + p(\beta'|\bar{\beta}_1, \beta_2) \cdot p(\bar{\beta}_1, \beta_2) \\
&\quad + p(\beta'|\beta_1, \beta_2) \cdot p(\beta_1, \beta_2). \\
&= \frac{1}{2}\epsilon_{l-1}(1 - \epsilon_{l-1}) + \frac{1}{2}(1 - \epsilon_{l-1})\epsilon_{l-1} + \frac{1}{2}\epsilon_{l-1}^2. \\
&= \epsilon_{l-1} \left( 1 - \frac{1}{2}\epsilon_{l-1} \right). \tag{5.2}
\end{aligned}$$

Similarly, by considering the inhibition edge in Figure 5.2, the error probability in the  $l$ -th iteration of node  $x_2$  can be obtained using the Boolean truth table for inhibition. The iterative equation for inhibition would be the same as equation (5.2). Supposing that node  $f_2$  is of degree  $j$ , then at the  $l$ -th iteration, the total propagated error probability in the message sent from  $f_2$  to node  $x_2$  is given by

$$\epsilon_l = \sum_{k=\lceil \frac{j-1}{2} \rceil}^{j-1} \binom{j-1}{k} y_{l-1}^k (1 - y_{l-1})^{j-1-k}, \tag{5.3}$$

where  $y_{l-1} = \epsilon_{l-1} (1 - \frac{1}{2}\epsilon_{l-1})$ . In the proposed model, a variable node sends its current state to its connected factor nodes in the subsequent iteration. Therefore, given a biological network with random state perturbations and factor node degree distribution  $\rho(x)$ , the average error probability on any particular gene node can be described by the recursive DE equation model as

$$\epsilon_l = \sum_{j=1}^{d_c} \rho_j \left[ \sum_{k=\lceil \frac{j-1}{2} \rceil}^{j-1} \binom{j-1}{k} y_{l-1}^k (1 - y_{l-1})^{j-1-k} \right], \tag{5.4}$$

where  $d_c$  is the maximum degree of the factor nodes.

By having a DE equation at hand, various connectivity analyses can be conducted. A simple but interesting one is as follows. Consider the DE equation (5.4). Expanding the

right-hand side of the equation yields

$$\epsilon_l = \rho_1 + (\rho_2 + 2\rho_3) \epsilon_{l-1} + O(\epsilon_{l-1}^2). \quad (5.5)$$

For  $\epsilon_l$  to be less than  $\epsilon_{l-1}$  for every  $l$  (i.e., vanishing state disturbances), it is necessary that  $\epsilon_{l-1}$  be larger than the first two terms on the right-hand side of equation (5.5). That is

$$\rho_1 + (\rho_2 + 2\rho_3) \epsilon_{l-1} < \epsilon_{l-1}. \quad (5.6)$$

Thus,

$$\rho_2 + 2\rho_3 < 1, \quad (5.7)$$

$$\rho_3 < 0.5. \quad (5.8)$$

The inequalities (5.6) – (5.8) provide some interesting intuitions. First, note that  $\rho_1$  is a constant, and as such, errors on degree-one distribution nodes (i.e., genes without regulators) do not vanish unless they are self-regulating. Second, the inequalities (5.7) and (5.8) indicate that in order to achieve a resilient genetic network, a large portion of the factor nodes should have degree  $j > 3$ . However, while  $j$  is in theory unbounded and can be equal to  $n$ , i.e., number of nodes, it is noteworthy that gene networks follow a power-law distribution with an exponent greater than 2 [114]. This restricts the upper bound on  $j$ . In this error model, factor nodes with higher degrees provide more information about the true state of their corresponding variable nodes from neighboring nodes.

### 5.2.3 Error Analysis II

In this section, I performed error propagation analysis of biological processes that implement a logical cooperative effect at the factor nodes following the activation-inhibition function proposed by Martin et al. [16]. Consider a portion of a factor graph with five variable nodes, as shown in Figure 5.3. Nodes  $x_1$  and  $x_2$  activate node  $x$ , whereas nodes  $x_3$  and  $x_4$  repress  $x$ .

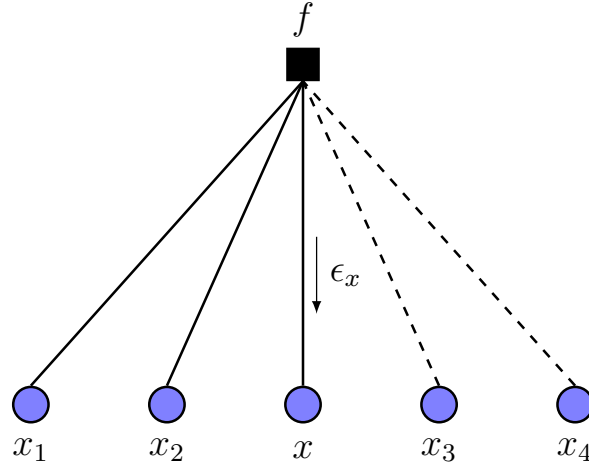


Figure 5.3: Factor graph with initial disturbance error probability  $\epsilon$ . Genes  $x_1$  and  $x_2$  activate gene  $x$ , whereas genes  $x_3$  and  $x_4$  inhibit gene  $x$ . Dashed edges denote inhibition interactions.

The next state of node  $x$  is given by the formulation

$$x = \underbrace{(x_1 \vee x_2)}_{x_a} \wedge \underbrace{\neg(x_3 \vee x_4)}_{x_r}, \quad (5.9)$$

where  $x_a$  ( $x_r$ ) is the logical combination of all activators (repressors) regulating a node, respectively. Therefore,  $x = x_a \wedge x_r$ . Each node in the graph is initially disturbed with a small probability  $\epsilon$  independently from other nodes. It is assumed that  $\epsilon$  is sufficient enough to change the state of a node. Let  $\epsilon_x$  denote the event that there is an error on node  $x$  due to random error from its regulators. To quantify the overall error propagation in the biological network, I represented the logical cooperative effect form of biological processes using truth tables, as shown in Table 5.1.

Furthermore, let  $\epsilon_l$  represent the propagated error in the  $l$ -th iteration of message passing. We may look at the evolution within one (any) iteration, and see how the disturbance probability changes for the gene nodes. If this probability increases at the end of each iteration, then a cascade will occur. However, if it decreases, then the effect of disturbance is minimized. Let us consider the  $l$ -th iteration on node  $x$  in Figure 5.3. Using the cooperative effect truth table in Table 5.1, we can represent the probability of error in node  $x$  as

Table 5.1: Boolean truth tables for implementing logical cooperative rule of equation (5.9)

Activation			Repression			Cooperative Effect		
$x_1$	$x_2$	$x_a$	$x_3$	$x_4$	$x_r$	$x_a$	$x_r$	$x_a \wedge x_r$
0	0	0	0	0	1	0	0	0
0	1	1	0	1	0	0	1	0
1	0	1	1	0	0	1	0	0
1	1	1	1	1	0	1	1	1

$$\epsilon_l = \Pr(\epsilon_x | x_a = 0, x_r = 0) \cdot \Pr(x_a = 0, x_r = 0) \quad (5.10a)$$

$$+ \Pr(\epsilon_x | x_a = 1, x_r = 0) \cdot \Pr(x_a = 1, x_r = 0) \quad (5.10b)$$

$$+ \Pr(\epsilon_x | x_a = 0, x_r = 1) \cdot \Pr(x_a = 0, x_r = 1) \quad (5.10c)$$

$$+ \Pr(\epsilon_x | x_a = 1, x_r = 1) \cdot \Pr(x_a = 1, x_r = 1). \quad (5.10d)$$

Equation (5.10) is obtained by considering the probability of having an error on node  $x$ , i.e.,  $\epsilon_x$ , given the input states,  $x_a$  and  $x_r$ . Assuming that we are at the beginning of the  $l$ -th iteration, we describe this error in terms of the  $(l-1)$ -th iteration, i.e.,  $\epsilon_l = f(\epsilon_{l-1})$ . The detailed derivation to solve equation (5.10) is shown in the Appendix. According to equation (A.11), it becomes

$$\begin{aligned} \epsilon_l = \frac{1}{2^{2k_i}} & \left\{ 2^{k_i} (2^{k_a} - 1) - \left[ (2^{k_a+1} - 1) (1 - \epsilon_{l-1})^{k_a} - 3 \cdot 2^{k_a} + 4^{k_a} + 1 \right] \right. \\ & \times (2 - 2\epsilon_{l-1})^{k_r} - (2^{k_r} - 1) \left[ 1 - (1 - \epsilon_{l-1})^{k_a} - 2^{k_a} + 4^{k_a} \right] \\ & \left. \times [(1 - \epsilon_{l-1})^{k_r} - 1] \right\}, \end{aligned} \quad (5.11)$$

where  $k_a$  ( $k_r$ ) denotes the cardinality of activating (inhibiting) edges of a node, respectively. In addition,  $k_i = k_a + k_r$  is the discrete index set referred to as the degree of a node  $x_i$ .

To account for the average error probability in the network, I employed polynomials

that represent the degree distributions of the networks in terms of activation and inhibition as follows:

$$\rho(u, v) = \sum_{i, j \geq 0} \rho_{ij} u^i v^j, \quad (5.12)$$

where  $\rho_{ij}$  is the fraction of edges incident to a factor node with degree  $i$  activators ( $u$ ) and degree  $j$  repressors ( $v$ ), constrained to  $\sum_{i, j \geq 0} \rho_{ij} = 1$ . The polynomial representation allows us to model random networks with arbitrary degree distributions such as biological networks. Of note,  $\rho(u, v)$  is obtained from the topology of the network. Therefore, given a biological network with factor node degree distribution  $\rho(u, v)$  and experiencing a perturbation, the average error probability of the network can be obtained as

$$\begin{aligned} \epsilon_l = \sum_{k_a, k_r \geq 0} \frac{\rho_{k_a k_r}}{2^{2k_i}} \cdot \left\{ 2^{k_i} (2^{k_a} - 1) - \left[ (2^{k_a+1} - 1) (1 - \epsilon_{l-1})^{k_a} - 3 \cdot 2^{k_a} + 4^{k_a} + 1 \right] \right. \\ \times (2 - 2\epsilon_{l-1})^{k_r} - (2^{k_r} - 1) \left[ 1 - (1 - \epsilon_{l-1})^{k_a} - 2^{k_a} + 4^{k_a} \right] \\ \left. \times [(1 - \epsilon_{l-1})^{k_r} - 1] \right\}. \end{aligned} \quad (5.13)$$

Equation (5.13) represents the recursive DE equation for the cooperative effect biological model in equation (5.9). After an initial perturbation occurs in the network, error messages appear in the network. It is expected that the impact of disturbance in the network diminishes if and only if  $\epsilon_l \rightarrow 0$  as  $l \rightarrow \infty$ .

Once the recursive equation of density evolution is obtained for the network, it can be used to gain useful insights into the structure of biological networks and allow us to quantify network parameters in designing models for reverse engineering gene networks from gene-expression profiles. Consider the recursive DE equation (5.13). By taking the Taylor series from the right-hand side of equation (5.13) at  $\epsilon_{l-1} = 0$ , we obtain

$$\epsilon_l = \beta_{u,v} \epsilon_{l-1} + O(\epsilon_{l-1}^2), \quad (5.14)$$

where

$$\beta_{u,v} = \left[ \frac{3}{4}\rho_{10} + \frac{3}{4}\rho_{11} + \frac{5}{8}\rho_{12} + \frac{57}{128}\rho_{13} + \frac{7}{8}\rho_{20} + \rho_{21} + \frac{7}{8}\rho_{22} + \frac{163}{256}\rho_{23} + \frac{45}{64}\rho_{30} + \frac{129}{128}\rho_{31} + \frac{241}{256}\rho_{32} + \frac{45}{64}\rho_{33} \right]. \quad (5.15)$$

In this series expansion, I computed  $\epsilon_l$  for every  $\{k_a, k_r\} = \{0, \dots, 3\}$ . Here, the maximum value of  $\{k_a, k_r\}$  is limited to 3. Similarly, in most gene network reconstruction tools and algorithms proposed in the literature, researchers limit the size of each node's parents. This is done to reduce the computational complexity required to search all combinations of parents [17, 28, 114]. For vanishing errors, i.e.,  $\epsilon_l$  to be less than  $\epsilon_{l-1}$ , one can conclude from equation (5.14) that

$$\beta_{u,v}\epsilon_{l-1} < \epsilon_{l-1}. \quad (5.16)$$

Inequality (5.16) holds for every  $l$  if it holds for  $l = 1$ . Therefore, the necessary condition becomes  $\beta_{u,v} < 1$ . From equation (5.15), factors  $\rho_{21}$  and  $\rho_{31}$  are noted as the dominant factors having coefficients  $\geq 1$ . Also, factors  $\rho_{01}$ ,  $\rho_{02}$ , and  $\rho_{03}$  are absent; thus, I hypothesize that nodes with only inhibiting regulators have less significance in cascading errors in biological systems.

### 5.3 Discussion and Results

In this section, I provided a quantitative evaluation of the proposed error propagation model in MATLAB environment. The density evolution equations (5.6), (5.13), and the inequality  $\beta_{u,v} < 1$  are evaluated, followed by a computational complexity analysis of the propagation model.

#### 5.3.1 Numerical Evaluations

First, equation (5.6) is evaluated for arbitrary BNs. Figure 5.4 shows an error evolution (initial  $\epsilon_0 = 0.25$ ) for a set of different factor node degree distributions in genetic graphs as the number of message-passing iterations grows. Moreover, I included the Li model with a network distribution given by  $\rho(x) = 0.025x + 0.05x^2 + 0.3x^3 + 0.2x^4 + 0.125x^5 + 0.3x^6$  in Figure 5.4. The Li model network distribution demonstrates its robustness to random state

disturbances. In addition, as expected, a violation of inequalities (5.7) and (5.8) is such that  $\epsilon_l \not\rightarrow 0$ .

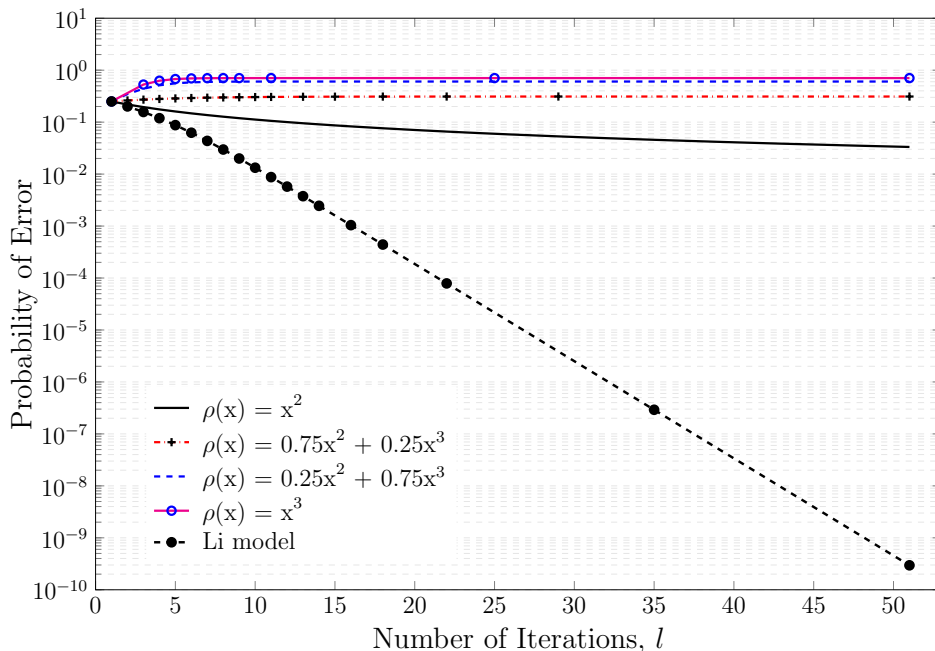


Figure 5.4: Error probability under density evolution for networks with different  $\rho(x)$  as  $l \rightarrow \infty$ . Initial  $\epsilon_0 = 0.25$ .

The analytical solutions given suggest that in the inference of BNs from gene-expression profiles, the DE equation can allow us to determine the degree distribution restrictions on the nodes for diminishing errors arising from random state perturbations. For instance, the DE analysis resulted in a necessary condition on the node degree distributions for biological systems to heal after an initial state perturbation. Furthermore, it revealed that low average connectivity may preclude the homeostatic stability of cellular systems since the number of attractors becomes high. These observations have significant implications on models used for inferring biological Boolean graphs from gene-expression data in reference to network stability. In such models, the authors have often limited the degree of connectivity to less than or equal to 3, citing model complexity and poor performance metrics [16, 57].

Next, I evaluated equation (5.13) for a sample random Boolean network with parameter value  $\beta_{u,v} = 0.6651$ , where the maximum  $k_a$  and  $k_r$  value equals 3. Then, I exemplified

a random network with random  $\rho_{k_a k_r}$  values such that  $\sum_{k_a, k_r \geq 0}^3 \rho_{k_a k_r} = 1$ , resulting in a  $\beta_{u,v}$  value of 0.6651 following equation (5.15). Figure 5.5, shows the total propagated error for different numbers of iterations,  $l$ . At the end of the first iteration, the average propagated error on any particular node decreases, and so  $\epsilon_l < \epsilon_{l-1}$  if and only if  $\beta_{u,v} < 1$ .

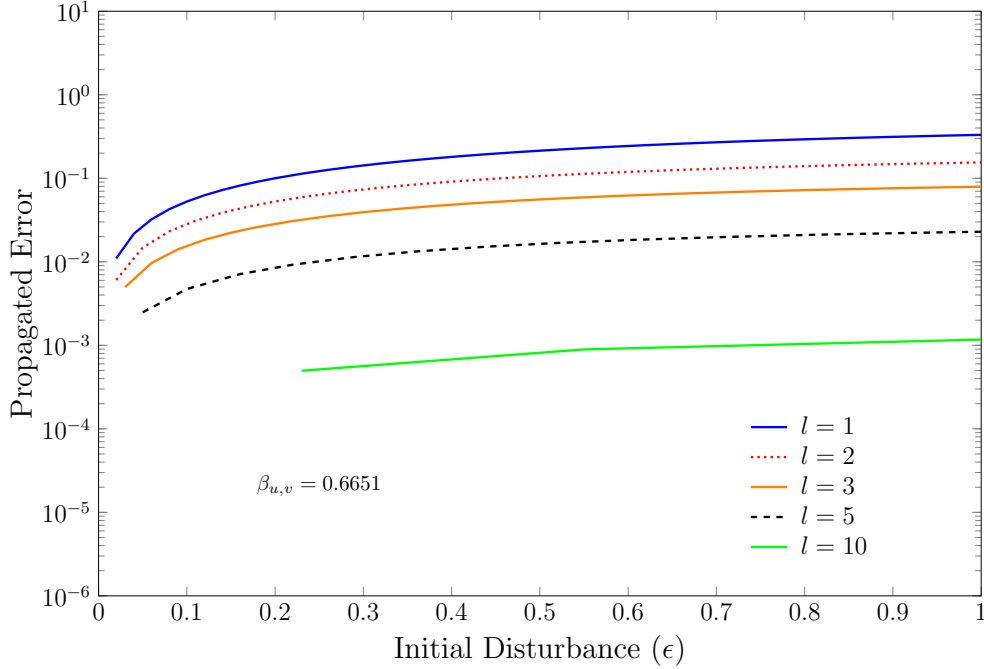


Figure 5.5: Propagated error in biological network with  $\beta_{u,v} = 0.6651$  in different iterations w.r.t. the initial disturbance. As  $l \rightarrow \infty$ , the evolved error vanishes in  $\beta_{u,v} < 1$ .

Similar to observations made in decoding algorithms for codes on graph, the network or graph structure plays an important role in the propagation of errors or noise in genetic graphs as depicted in Figure 5.6. In Figure 5.6, the plots show the quantity of error propagated in sample Boolean gene graphs, such as the budding yeast cell-cycle network model presented by Li et al. [23], a random network with  $\beta_{u,v} = 0.5451$ , and two theoretical networks represented using polynomial degree distributions given by  $\rho(u, v) = 0.9u^3v + 0.1u^3v^2$  and  $\rho(u, v) = 0.4u^2v + 0.6u^3v$ , both of which guarantee that  $\beta_{u,v} \geq 1$ . Moreover, the yeast logical network [23] considered has the following polynomial degree distribution according to equation (5.12):

$$\begin{aligned} \rho(u, v) = & 0.0345u + 0.1379uv + 0.1034uv^2 + 0.1379uv^3 \\ & + 0.1379u^2 + 0.1034u^2v + 0.3448u^2v^3. \end{aligned} \tag{5.17}$$

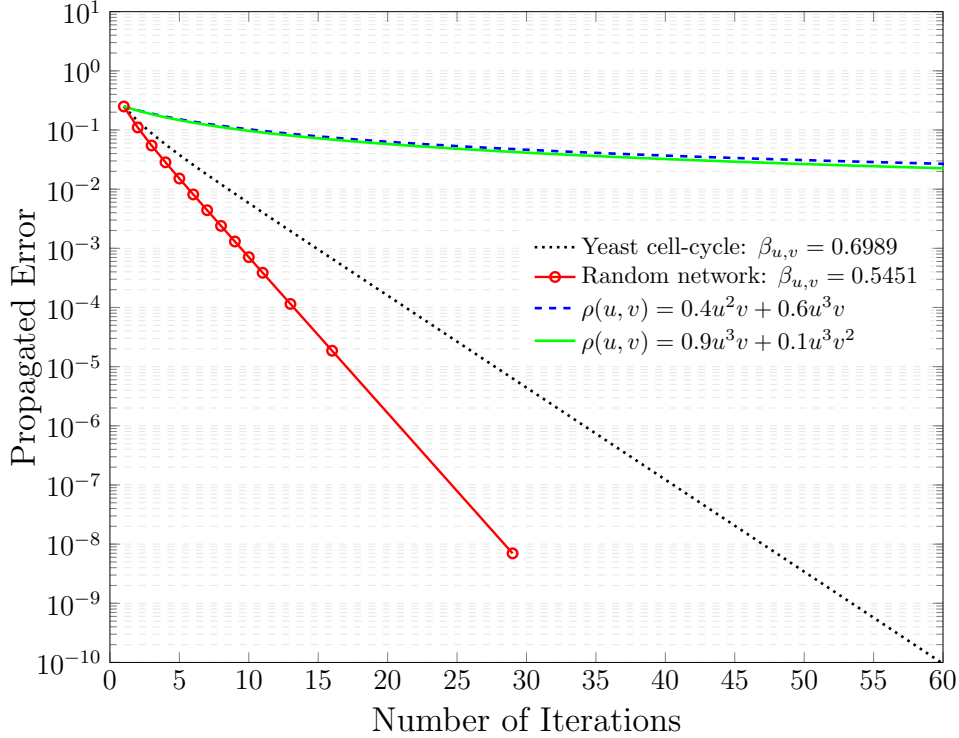


Figure 5.6: Probability of error for networks with different polynomial degree distributions, i.e.,  $\beta_{u,v}$  values (shown in the plot) versus the number of iterations,  $l \rightarrow \infty$ . Initial disturbance  $\epsilon_0 = 0.25$ . Networks  $\rho(u, v) = 0.9u^3v + 0.1u^3v^2$  and  $\rho(u, v) = 0.4u^2v + 0.6u^3v$  both have  $\beta_{u,v} \geq 1$ .

The polynomial degree distribution in equation (5.17) yields  $\beta_{u,v} = 0.6989$ , which implies that the network may be resilient against random state disturbances. As can be seen in Figure 5.6, for networks with  $\beta_{u,v} < 1$ , an error introduced due to a network disturbance vanishes as  $l \rightarrow \infty$ , in accordance with the inequality (5.16). In addition, a violation of this inequality is such that  $\epsilon_l \not\rightarrow 0$ . For instance, the two networks evaluated in Figure 5.6 with dominant  $\rho_{21}$  and  $\rho_{31}$  factors are such that  $\beta_{u,v} \geq 1$ ; thus, their corresponding line plots do not converge to zero. If the error vanishes, then we may deduce that the initial states of the network are reset, and therefore, the steady-state distribution of the network remains

invariant. Conversely, a non-zero error indicates that the initial perturbation throws the network out of its steady-state condition, which is consistent with several theoretical and experimental studies [101, 111]. That is, gene expression noise can create new stable states or destabilize existing ones [102, 103, 104].

Of note, Li et al. in their work [23] observed that the yeast logical network has an observable large stable state, thus making it robust against network perturbations. The analytical results highlight the importance of degree distribution restrictions on genetic graphs that will be necessary for understanding signal fidelity in natural networks as well as in the design of noise-tolerant artificial gene circuits. This observation supports conclusions made in the literature that the topology of a genetic graph does not only determine its information-processing capability but also encodes its sensitivity to noise [101, 111]. In simple terms, network topology provides a means to locally tune noise propagation.

In this chapter, it is assumed that the error introduced into the network is sufficient to change the state of nodes randomly in the graph. Unlike the check node functionality in codes on graphs that is simple and only performs a modulo 2 operation to derive what it believes about the value of its neighboring nodes, the factor nodes in a biological factor graph may assume a more complicated functionality. For example, to a certain extent, the cooperative effect biological process implemented using discrete states tends to correct errors introduced in the network. From the DE equation, a necessary condition is obtained as a function of the network connectivity, i.e.,  $\beta_{u,v} = f(\rho_{ij})$ . Hence, we may deduce that network structure plays a critical role in the stability of biological graphs.

### 5.3.2 Computational Complexity

In the proposed model, the overall computational cost of characterizing the propagated error in the network is given by the complexity of the message-passing algorithm employed on the factor graph model. This cost grows linearly with the number of edges in the network [25, 32]. The computation at the factor nodes occurs in parallel, and in each iteration a factor node performs only a single linear computation given by equation (5.9). Given that each Boolean computation incurs a constant complexity, i.e.,  $\mathcal{O}(1)$ , the total time

complexity of a factor node is  $\mathcal{O}(1)$ . On the other hand, variable nodes in the factor graph model simply send out the value of their current state. Similarly, a variable node incurs a constant time complexity of  $\mathcal{O}(1)$ . Therefore, the time complexity of the proposed model is  $\mathcal{O}(1)$  per iteration for each pair of factor node and variable node. For a constant number of iterations, the complexity of this model is proportional to the number of nodes in the graph, i.e., of order  $\mathcal{O}(n)$ . In large biological networks where few highly connected nodes regulate large poorly connected nodes, the complexity of the model is linear in the number of nodes.

### 5.3.3 Models Comparison

In the literature, noise propagation in biological systems has been studied theoretically and experimentally. For example, Pedraza and Oudenaarden [108] explored noise propagation in genetic cascades using single-cell measurements. Hooshangi and Weiss [115] also computationally and theoretically analyzed noise propagation in linear cascades and reported that the existence of feedback loops in biological systems may provide robustness to extrinsic noise. They further showed that it is impossible to achieve lower-than-intrinsic noise control for  $n$ -ring gene regulatory networks with an odd number of nodes and negative regulations. In their work [110], Zhang et al. performed bifurcation analyses of ordinary differential equations for the Myc/E2F/miR-17-92 network [116], in order to study the role of feedback loop in sensitivity and noise amplification on the dynamic properties of the system.

These previous works mostly depend on the knowledge of biochemical and kinetic reactions which may be difficult to obtain for large genetic networks. Moreover, they focus mainly on investigating noise propagation in biological systems by considering how feedback loops impact noise in the system. These investigations have provided insights into how to tune or control noise; however, a study on how to quantify noise in large arbitrary genetic networks without the explicit determination of biochemical equations is lacking, thereby obscuring a direct quantitative comparison with the proposed model.

### 5.3.4 Application to Hub-Like GRNs

The general framework of factor graphs is a powerful tool that has been employed in many fields of research [21, 25, 32, 33]. For example, in coding theory, factor graphs have been used to model networks of 100,000 nodes and more, due to the sparse distribution of nodes in these networks. Similarly, a key experimental observation is that large biological graphs have sparsely distributed and possibly long edges [117, 118], i.e., a few highly connected nodes known as hubs regulate large, poorly connected nodes. In regulatory hub-like networks or large biological networks, one can adapt equation (5.15) by including higher-order terms in the Taylor series expansion of equation (5.13) as well as increasing the maximum allowable  $k_a$  and  $k_r$  values to obtain a more refined necessary condition in terms of  $\beta_{u,v}$  for vanishing errors or noise.

## 5.4 Summary

In this chapter, I explored Boolean graphical model representation of biological networks and applied message passing to evolve errors or network perturbations in such networks. Subsequently, a density evolution analysis is provided to study the behavior of biological systems in the presence of noise. The derived closed-form expressions enable an analytic approach to quantify and characterize the evolution of errors due to randomness in gene expressions.

The analytical results yielded a necessary condition on the network parameters for network resilience against perturbations. In other words, given the activation-inhibition Boolean functions implemented to model the cooperative effect biological rules, the derived necessary condition can allow us to constrain network parameters such as connectivity in designing reliable and noise-tolerant artificial gene circuits. This chapter's work and approach provide a step towards understanding error propagation in complex genetic graphs with the aim of inciting research on noise control and intervention strategies for noisy biological systems. Finally, a possible future path would be to consider error propagation in factor graph models with unique computational rules on each factor node.

## CHAPTER 6

### CONCLUSION AND FUTURE WORK

#### 6.1 Conclusion

The driving force behind this dissertation was the need to elucidate and improve our understanding of stochastic complex genome interactions in biological systems. Moreover, to quantify the response of genetic networks to perturbations and errors such as DNA mutations and gene knockouts. To achieve these objectives, I formulated and developed computational frameworks based on Bayesian network and Boolean network model representations of gene networks. Also, unsupervised machine learning techniques were applied to obtain optimal gene expression clusters necessary for computing the regulatory functions of genes. Then, I developed network inference algorithms to analyze the performance and dynamic properties of these networks.

#### 6.2 Future Work

This research work opened up many theoretical and practical research possibilities that offer potential for future investigation, which are briefly described:

##### 6.2.1 Signal Processing on Graphs

Massive amounts of data are being generated, recorded, and processed in every aspect of human life. For instance, beginning with next-generation sequencing of the human genome, health monitoring apps and devices, our social networks, financial and banking data, traffic patterns, and online marketing preferences. This data resides on irregular and complex structures that obscures processing using standard tools.

A graph, defined as a structure that encodes relationships among a set of elements, offers the ability to model such data and their complex interactions. Here, the object of interest is the information overlaid on the graph. In graph signal processing, the structure of the data is modeled using a graph, and then the available information as a signal defined

on the graph is explored. To understand data residing on graphs, we need to redesign the classical signal processing concepts and tools used to study signals in communication engineering, such as filtering, Fourier transform, frequency response, etc.

### 6.2.2 Error Propagation in Gene Networks

Accurate prediction of noise propagation in biological networks is key to understanding faithful signal propagation, and useful in the design of noise-tolerant synthetic gene circuits. Knowledge on how biological fluctuations propagate up the development ladder of biological systems is currently lacking. Thus, there is a need to determine how the variability that originates at the gene-expression level propagates up to the level of tissues, organs, whole organisms, and populations. Along this developmental ladder, there are multiple layers of regulation, both extra-cellular and intra-cellular, that can shape the characteristics of noise.

Analyzing noise and characterizing its propagation on such complex multi-scale, non-linear, and non-equilibrium systems not only pose serious technical challenges but call for new conceptual paradigms. In this research direction, the objective is to develop computational methods and tools for studying noise propagation in complex networks and how this propagation impacts the dynamical properties of the network, as well as to formulate theoretical frameworks for characterizing signal fidelity in multi-layered gene networks.

One of the tasks in this study would be to investigate the influence of network structure or architecture on the stationary dynamics of biological networks. For instance, in a recent publication [109], the authors estimated the macroscopic uncertainty in the critical threshold for certain dynamical processes in networks with noisy edges. They considered a family of dynamical processes where the critical point depends on the largest eigenvalue of the connectivity matrix. Under this task, one may be interested in understanding the role of node degree distribution of the network in the estimation of a critical threshold that is necessary to predict the onset of phase transitions in decision-making biological processes. Another task would be to explore and design noise-tolerant synthetic gene circuits by taking advantage of the principles used in the analysis and design of today's most powerful and

practical error-tolerant control codes in engineering communication theory. This task would attempt to illustrate design principles necessary for reducing or controlling biological fluctuations, as well as investigate the key network parameters that can be optimized in designing noise-tolerant artificial gene networks.

This research advances the body of knowledge for developing computational tools and algorithms needed to grasp the complexities and interconnectedness of gene networks while exploring the stochastic interactions among connected entities. The outcome of this research has merit in both natural and synthetic biology applications, for example, designing artificial gene circuits for medical, biotechnological, and industrial applications; and understanding genomic functions and the behavior of gene networks for the development of new therapeutics and drugs.

## REFERENCES

## REFERENCES

- [1] S. A. Kauffman, *The origins of order: Self-organization and selection in evolution*. USA: Oxford University Press, 1993.
- [2] K. Tchourine, C. Vogel, and R. Bonneau, “Condition-specific modeling of biophysical parameters advances inference of regulatory networks,” *Cell Reports*, vol. 23, no. 2, 2018, pp. 376–388.
- [3] X. Zhou and X. Cai, “Inference of differential gene regulatory networks based on gene expression and genetic perturbation data,” *Bioinformatics*, July 2019.
- [4] P. B. Madhamshettiwar, S. R. Maetschke, M. J. Davis, A. Reverter, and M. A. Ragan, “Gene regulatory network inference: Evaluation and application to ovarian cancer allows the prioritization of drug targets,” *Genome Medicine*, vol. 4, no. 5, 2012, p. 41.
- [5] R. J. Prill, D. Marbach, J. Saez-Rodriguez, P. K. Sorger, L. G. Alexopoulos, X. Xue, N. D. Clarke, G. Altan-Bonnet, and G. Stolovitzky, “Towards a rigorous assessment of systems biology models: The DREAM3 challenges,” *PloS One*, vol. 5, no. 2, 2010, p. e9202.
- [6] G. Karlebach and R. Shamir, “Modelling and analysis of gene regulatory networks,” *Nature Reviews Molecular Cell Biology*, vol. 9, no. 10, 2008, pp. 770–780.
- [7] G. Stolovitzky, D. Monroe, and A. Califano, “Dialogue on reverse-engineering assessment and methods: The DREAM of high-throughput pathway inference,” *Annals of the New York Academy of Sciences*, vol. 1115, no. 1, 2007, pp. 1–22.
- [8] D. Marbach, J. C. Costello, R. Küffner, N. M. Vega, R. J. Prill, D. M. Camacho, K. R. Allison, A. Aderhold, R. Bonneau, Y. Chen *et al.*, “Wisdom of crowds for robust gene network inference,” *Nature Methods*, vol. 9, no. 8, 2012, pp. 796–804.
- [9] P. Chouvardas, G. Kollias, and C. Nikolaou, “Inferring active regulatory networks from gene expression data using a combination of prior knowledge and enrichment analysis,” *BMC Bioinformatics*, vol. 17, no. 5, 2016, p. 181.
- [10] X. Yu, H. Gao, X. Zheng, C. Li, and J. Wang, “A computational method of predicting regulatory interactions in *Arabidopsis* based on gene expression data and sequence information,” *Computational Biology and Chemistry*, vol. 51, 2014, pp. 36–41.
- [11] A. J. Hartemink, D. K. Gifford, T. S. Jaakkola, and R. A. Young, “Combining location and expression data for principled discovery of genetic regulatory network models,” in *Biocomputing 2002*. World Scientific, 2001, pp. 437–449.

## REFERENCES (continued)

- [12] K. C. Chen, L. Calzone, A. Csikasz-Nagy, F. R. Cross, B. Novak, and J. J. Tyson, “Integrative analysis of cell cycle control in budding yeast,” *Molecular Biology of the Cell*, vol. 15, no. 8, 2004, pp. 3841–3862.
- [13] M. de Hoon, S. Imoto, and S. Miyano, “Inferring gene regulatory networks from time-ordered gene expression data using differential equations,” in *International Conference on Discovery Science*. Springer, 2002, pp. 267–274.
- [14] S. Kauffman, “Homeostasis and differentiation in random genetic control networks,” *Nature*, vol. 224, no. 5215, 1969, pp. 177–178.
- [15] A. Lovrics, Y. Gao, B. Juhász, I. Bock, H. M. Byrne, A. Dinnyés, and K. A. Kovács, “Boolean modelling reveals new regulatory connections between transcription factors orchestrating the development of the ventral spinal cord,” *PloS One*, vol. 9, no. 11, 2014, p. e111430.
- [16] S. Martin, Z. Zhang, A. Martino, and J.-L. Faulon, “Boolean dynamics of genetic regulatory networks inferred from microarray time series data,” *Bioinformatics*, vol. 23, no. 7, 2007, pp. 866–874.
- [17] F. Liu, S.-W. Zhang, W.-F. Guo, Z.-G. Wei, and L. Chen, “Inference of gene regulatory network based on local Bayesian networks,” *PLoS Computational Biology*, vol. 12, no. 8, 2016, p. e1005024.
- [18] N. Friedman, M. Linial, I. Nachman, and D. Pe’er, “Using Bayesian networks to analyze expression data,” *Journal of Computational Biology*, vol. 7, no. 3-4, 2000, pp. 601–620.
- [19] I. Gat-Viks, A. Tanay, D. Rajman, and R. Shamir, “A probabilistic methodology for integrating knowledge and experiments on biological networks,” *Journal of Computational Biology*, vol. 13, no. 2, 2006, pp. 165–181.
- [20] M. W. Covert, E. M. Knight, J. L. Reed, M. J. Herrgard, and B. O. Palsson, “Integrating high-throughput and computational data elucidates bacterial networks,” *Nature*, vol. 429, no. 6987, 2004, pp. 92–96.
- [21] F. R. Kschischang, B. J. Frey, H.-A. Loeliger *et al.*, “Factor graphs and the sum-product algorithm,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, 2001, pp. 498–519.
- [22] S. Maslov, “Topological and dynamical properties of protein interaction networks,” in *Protein-Protein Interactions and Networks*. Springer, 2008, pp. 115–137.
- [23] F. Li, T. Long, Y. Lu, Q. Ouyang, and C. Tang, “The yeast cell-cycle network is robustly designed,” *Proceedings of the National Academy of Sciences*, vol. 101, no. 14, 2004, pp. 4781–4786.

## REFERENCES (continued)

- [24] S.-Y. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, “On the design of low-density parity-check codes within 0.0045 dB of the shannon limit,” *IEEE Communications Letters*, vol. 5, no. 2, 2001, pp. 58–60.
- [25] T. J. Richardson and R. L. Urbanke, “The capacity of low-density parity-check codes under message-passing decoding,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, 2001, pp. 599–618.
- [26] J. Pearl, *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. San Francisco: Elsevier, 2014.
- [27] N. Friedman, K. Murphy, and S. Russell, “Learning the structure of dynamic probabilistic networks,” *arXiv preprint arXiv:1301.7374*, 2013.
- [28] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, “The max-min hill-climbing Bayesian network structure learning algorithm,” *Machine Learning*, vol. 65, no. 1, 2006, pp. 31–78.
- [29] D. M. Chickering, “Learning Bayesian networks is NP-complete,” in *Learning from Data*. Springer, 1996, pp. 121–130.
- [30] H. Rue and L. Held, *Gaussian Markov random fields: theory and applications*. USA: CRC press, 2005.
- [31] R. Tanner, “A recursive approach to low complexity codes,” *IEEE Transactions on Information Theory*, vol. 27, no. 5, 1981, pp. 533–547.
- [32] S. Kotiang and A. Eslami, “Boolean factor graph model for biological systems: the yeast cell-cycle network,” *BMC Bioinformatics*, vol. 22, no. 1, 2021, pp. 1–27.
- [33] A. Behfarnia and A. Eslami, “Error correction coding meets cyber-physical systems: Message-passing analysis of self-healing interdependent networks,” *IEEE Transactions on Communications*, vol. 65, no. 7, 2017, pp. 2753–2768.
- [34] S. Kotiang and A. Eslami, “A probabilistic graphical model for system-wide analysis of gene regulatory networks,” *Bioinformatics*, vol. 36, no. 10, 2020, pp. 3192–3199.
- [35] D. Divsalar, S. Dolinar, and F. Pollara, “Iterative turbo decoder analysis based on density evolution,” *IEEE Journal on Selected areas in Communications*, vol. 19, no. 5, 2001, pp. 891–907.
- [36] C. A. Gallo, R. L. Cecchini, J. A. Carballido, S. Micheletto, and I. Ponzoni, “Discretization of gene expression data revised,” *Briefings in Bioinformatics*, vol. 17, no. 5, 2016, pp. 758–770.
- [37] T. K. Moon, “The expectation-maximization algorithm,” *IEEE Signal Processing Magazine*, vol. 13, no. 6, 1996, pp. 47–60.

## REFERENCES (continued)

- [38] S. Tripathi, J. Lloyd-Price, A. Ribeiro, O. Yli-Harja, M. Dehmer, and F. Emmert-Streib, “sgnesR: An R package for simulating gene expression data from an underlying real gene network structure considering delay parameters,” *BMC Bioinformatics*, vol. 18, no. 1, 2017, p. 325.
- [39] M. Fratello, A. Serra, V. Fortino, G. Raiconi, R. Tagliaferri, and D. Greco, “A multi-view genomic data simulator,” *BMC Bioinformatics*, vol. 16, no. 1, 2015, p. 151.
- [40] E. Klipp, R. Herwig, A. Kowald, C. Wierling, and H. Lehrach, *Systems biology in practice: Concepts, implementation and application*. New Jersey: John Wiley & Sons, 2008.
- [41] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer, “COPASI—A complex pathway simulator,” *Bioinformatics*, vol. 22, no. 24, 2006, pp. 3067–3074.
- [42] J. S. Yedidia, W. T. Freeman, and Y. Weiss, “Constructing free-energy approximations and generalized belief propagation algorithms,” *IEEE Transactions on Information Theory*, vol. 51, no. 7, 2005, pp. 2282–2312.
- [43] G. F. Cooper, “The computational complexity of probabilistic inference using Bayesian belief networks,” *Artificial Intelligence*, vol. 42, no. 2-3, 1990, pp. 393–405.
- [44] S. M. Aji and R. J. McEliece, “The generalized distributive law,” *IEEE Transactions on Information Theory*, vol. 46, no. 2, 2000, pp. 325–343.
- [45] J. M. Mooij and H. J. Kappen, “Sufficient conditions for convergence of the sum-product algorithm,” *IEEE Transactions on Information Theory*, vol. 53, no. 12, 2007, pp. 4422–4437.
- [46] K. P. Murphy, Y. Weiss, and M. I. Jordan, “Loopy belief propagation for approximate inference: An empirical study,” in *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 1999, pp. 467–475.
- [47] S. Gama-Castro, H. Salgado, A. Santos-Zavaleta, D. Ledezma-Tejeida, L. Muniz-Rascado, J. S. García-Sotelo, K. Alquicira-Hernández, I. Martínez-Flores, L. Pannier, J. A. Castro-Mondragón *et al.*, “RegulonDB version 9.0: High-level integration of gene regulation, coexpression, motif clustering and beyond,” *Nucleic Acids Research*, vol. 44, no. D1, 2015, pp. D133–D143.
- [48] S. S. Shen-Orr, R. Milo, S. Mangan, and U. Alon, “Network motifs in the transcriptional regulation network of *Escherichia coli*,” *Nature Genetics*, vol. 31, no. 1, 2002, pp. 64–68.

## REFERENCES (continued)

- [49] K. Shimizu, “Metabolic regulation of a bacterial cell system with emphasis on *Escherichia coli* metabolism,” *ISRN Biochemistry*, vol. 2013, 2013.
- [50] J. W. Foster, “*Escherichia coli* acid resistance: Tales of an amateur acidophile,” *Nature Reviews Microbiology*, vol. 2, no. 11, 2004, pp. 898–907.
- [51] J. J. Faith, M. E. Driscoll, V. A. Fusaro, E. J. Cosgrove, B. Hayete, F. S. Juhn, S. J. Schneider, and T. S. Gardner, “Many microbe microarrays database: Uniformly normalized affymetrix compendia with structured experimental metadata,” *Nucleic Acids Research*, vol. 36, no. suppl\_1, 2007, pp. D866–D870.
- [52] B. S. Chlebus and S. H. Nguyen, “On finding optimal discretizations for two attributes,” in *International Conference on Rough Sets and Current Trends in Computing*. Springer, 1998, pp. 537–544.
- [53] D. E. Knuth, *Art of Computer Programming, volume 2: Seminumerical Algorithms*. Addison-Wesley Professional, 2014.
- [54] S. Maslov and K. Sneppen, “Specificity and stability in topology of protein networks,” *Science*, vol. 296, no. 5569, 2002, pp. 910–913.
- [55] M. Z. Rodriguez, C. H. Comin, D. Casanova, O. M. Bruno, D. R. Amancio, L. d. F. Costa, and F. A. Rodrigues, “Clustering algorithms: A comparative approach,” *PloS One*, vol. 14, no. 1, 2019, p. e0210236.
- [56] N. Noorshams and M. J. Wainwright, “Stochastic belief propagation: A low-complexity alternative to the sum-product algorithm,” *IEEE Transactions on Information Theory*, vol. 59, no. 4, 2012, pp. 1981–2000.
- [57] T. Akutsu, S. Miyano, and S. Kuhara, “Identification of genetic networks from a small number of gene expression patterns under the Boolean network model,” in *Biocomputing*. New Jersey: World Scientific, 1999, pp. 17–28.
- [58] S. A. Kauffman, “Metabolic stability and epigenesis in randomly constructed genetic nets,” *Journal of Theoretical Biology*, vol. 22, no. 3, 1969, pp. 437–467.
- [59] S. Huang, “Gene expression profiling, genetic networks, and cellular states: An integrating concept for tumorigenesis and drug discovery,” *Journal of Molecular Medicine*, vol. 77, no. 6, 1999, pp. 469–480.
- [60] L. S. Tsimring, “Noise in biology,” *Reports on Progress in Physics*, vol. 77, no. 2, 2014, p. 026601.
- [61] I. Shmulevich, E. R. Dougherty, and W. Zhang, “Gene perturbation and intervention in probabilistic Boolean networks,” *Bioinformatics*, vol. 18, no. 10, 2002, pp. 1319–1331.

## REFERENCES (continued)

- [62] S. Kauffman, C. Peterson, B. Samuelsson, and C. Troein, “Genetic networks with canalizing Boolean rules are always stable,” *Proceedings of the National Academy of Sciences*, vol. 101, no. 49, 2004, pp. 17 102–17 107.
- [63] X. Qu, M. Aldana, and L. P. Kadanoff, “Numerical and theoretical studies of noise effects in the Kauffman model,” *Journal of Statistical Physics*, vol. 109, no. 5, 2002, pp. 967–986.
- [64] I. Shmulevich, E. R. Dougherty, and W. Zhang, “Control of stationary behavior in probabilistic Boolean networks by means of structural intervention,” *Journal of Biological Systems*, vol. 10, no. 04, 2002, pp. 431–445.
- [65] Y. Xiao and E. R. Dougherty, “The impact of function perturbations in Boolean networks,” *Bioinformatics*, vol. 23, no. 10, 2007, pp. 1265–1273.
- [66] J. J. Hunter, “Stationary distributions and mean first passage times of perturbed markov chains,” *Linear Algebra and Its Applications*, vol. 410, 2005, pp. 217–243.
- [67] B. J. Frey, J. F. Brendan, and B. J. Frey, *Graphical models for machine learning and digital communication*. USA: MIT Press, 1998.
- [68] B. Novák and J. J. Tyson, “A model for restriction point control of the mammalian cell cycle,” *Journal of Theoretical Biology*, vol. 230, no. 4, 2004, pp. 563–579.
- [69] A. Fauré, A. Naldi, C. Chaouiya, and D. Thieffry, “Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle,” *Bioinformatics*, vol. 22, no. 14, 2006, pp. e124–e131.
- [70] A. G. Gonzalez, A. Naldi, L. Sanchez, D. Thieffry, and C. Chaouiya, “GINSim: A software suite for the qualitative modelling, simulation and analysis of regulatory networks,” *Biosystems*, vol. 84, no. 2, 2006, pp. 91–100.
- [71] C. Müssel, M. Hopfensitz, and H. A. Kestler, “BoolNet—an R package for generation, reconstruction and analysis of Boolean networks,” *Bioinformatics*, vol. 26, no. 10, 2010, pp. 1378–1380.
- [72] A. Naldi, “BioLQM: A Java toolkit for the manipulation and conversion of logical qualitative models of biological networks,” *Frontiers in Physiology*, vol. 9, 2018, p. 1605.
- [73] S. Klamt, J. Saez-Rodriguez, and E. D. Gilles, “Structural and functional analysis of cellular networks with CellNetAnalyzer,” *BMC Systems Biology*, vol. 1, no. 1, 2007, pp. 1–13.

## REFERENCES (continued)

- [74] D. Battogtokh and J. J. Tyson, “Bifurcation analysis of a model of the budding yeast cell cycle,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 14, no. 3, 2004, pp. 653–661.
- [75] A. Lovrics, A. Csikász-Nagy, I. G. Zsély, J. Zádor, T. Turányi, and B. Novák, “Time scale and dimension analysis of a budding yeast cell cycle model,” *BMC Bioinformatics*, vol. 7, no. 1, 2006, pp. 1–11.
- [76] S. A. Hoose, J. A. Rawlings, M. M. Kelly, M. C. Leitch, Q. O. Ababneh, J. P. Robles, D. Taylor, E. M. Hoover, B. Hailu, K. A. McEnery *et al.*, “A systematic analysis of cell cycle regulators in yeast reveals that most factors act independently of cell size to control initiation of division,” *PLoS Genet*, vol. 8, no. 3, 2012, p. e1002590.
- [77] L. Yu, L. P. Castillo, S. Mnaimneh, T. R. Hughes, and G. W. Brown, “A survey of essential gene function in the yeast cell division cycle,” *Molecular Biology of the Cell*, vol. 17, no. 11, 2006, pp. 4736–4747.
- [78] S. Braunewell and S. Bornholdt, “Superstability of the yeast cell-cycle dynamics: Ensuring causality in the presence of biochemical stochasticity,” *Journal of Theoretical Biology*, vol. 245, no. 4, 2007, pp. 638–643.
- [79] P. Kraikivski, K. C. Chen, T. Laomettachit, T. Murali, and J. J. Tyson, “From START to FINISH: Computational analysis of cell cycle control in budding yeast,” *NPJ Systems Biology and Applications*, vol. 1, no. 1, 2015, pp. 1–9.
- [80] L. Calzone, “Temporal organization of the budding yeast cell cycle: General principles and detailed simulations,” Ph.D. dissertation, Virginia Tech, 2003.
- [81] M. Tyers, “The cyclin-dependent kinase inhibitor p40Sic1 imposes the requirement for Cln G1 cyclin function at start,” *Proceedings of the National Academy of Sciences*, vol. 93, no. 15, 1996, pp. 7772–7776.
- [82] M. Schwab, A. S. Lutum, and W. Seufert, “Yeast Hct1 is a regulator of Clb2 cyclin proteolysis,” *Cell*, vol. 90, no. 4, 1997, pp. 683–693.
- [83] W. Niu, Z. Li, W. Zhan, V. R. Iyer, and E. M. Marcotte, “Mechanisms of cell cycle control revealed by a systematic and quantitative overexpression screen in *S. cerevisiae*,” *PLoS Genet*, vol. 4, no. 7, 2008, p. e1000120.
- [84] R. A. de Bruin, W. H. McDonald, T. I. Kalashnikova, J. Yates III, and C. Wittenberg, “Cln3 activates G1-specific transcription via phosphorylation of the SBF bound repressor Whi5,” *Cell*, vol. 117, no. 7, 2004, pp. 887–898.
- [85] H. Wijnen, A. Landman, and B. Futcher, “The G1 cyclin Cln3 promotes cell cycle entry via the transcription factor Swi6,” *Molecular and Cellular Biology*, vol. 22, no. 12, 2002, pp. 4402–4418.

## REFERENCES (continued)

- [86] I. Irurzun-Arana, J. M. Pastor, I. F. Trocóniz, and J. D. Gómez-Mantilla, “Advanced Boolean modeling of biological networks applied to systems pharmacology,” *Bioinformatics*, vol. 33, no. 7, 2017, pp. 1040–1048.
- [87] A. Fauré and D. Thieffry, “Logical modelling of cell cycle control in eukaryotes: A comparative study,” *Molecular BioSystems*, vol. 5, no. 12, 2009, pp. 1569–1581.
- [88] A. Naldi, P. T. Monteiro, and C. Chaouiya, “Efficient handling of large signalling-regulatory networks by focusing on their core control,” in *International Conference on Computational Methods in Systems Biology*. Springer, 2012, pp. 288–306.
- [89] A. Naldi, E. Remy, D. Thieffry, and C. Chaouiya, “Dynamically consistent reduction of logical regulatory graphs,” *Theoretical Computer Science*, vol. 412, no. 21, 2011, pp. 2207–2218.
- [90] A. Saadatpour, I. Albert, and R. Albert, “Attractor analysis of asynchronous Boolean models of signal transduction networks,” *Journal of Theoretical Biology*, vol. 266, no. 4, 2010, pp. 641–656.
- [91] S. Kotiang and A. Eslami, “Density evolution for noise propagation analysis in biological networks,” *IEEE Access*, vol. 10, 2022, pp. 4261–4270.
- [92] M. I. Davidich and S. Bornholdt, “Boolean network model predicts cell cycle sequence of fission yeast,” *PloS One*, vol. 3, no. 2, 2008, p. e1672.
- [93] P. S. Swain, M. B. Elowitz, and E. D. Siggia, “Intrinsic and extrinsic contributions to stochasticity in gene expression,” *Proceedings of the National Academy of Sciences*, vol. 99, no. 20, 2002, pp. 12 795–12 800.
- [94] Z. Wang and J. Zhang, “Impact of gene expression noise on organismal fitness and the efficacy of natural selection,” *Proceedings of the National Academy of Sciences*, vol. 108, no. 16, 2011, pp. E67–E76.
- [95] D. Noble, “The role of stochasticity in biological communication processes,” *Progress in Biophysics and Molecular Biology*, vol. 162, 2021, pp. 122–128.
- [96] T. M. Venancio, S. Balaji, S. Geetha, and L. Aravind, “Robustness and evolvability in natural chemical resistance: identification of novel systems properties, biochemical mechanisms and regulatory interactions,” *Molecular BioSystems*, vol. 6, no. 8, 2010, pp. 1475–1491.
- [97] J. M. Raser and E. K. O’Shea, “Control of stochasticity in eukaryotic gene expression,” *science*, vol. 304, no. 5678, 2004, pp. 1811–1814.
- [98] W. J. Blake, M. Kærn, C. R. Cantor, and J. J. Collins, “Noise in eukaryotic gene expression,” *Nature*, vol. 422, no. 6932, 2003, pp. 633–637.

## REFERENCES (continued)

- [99] V. Kohar and M. Lu, “Role of noise and parametric variation in the dynamics of gene regulatory circuits,” *NPJ systems biology and applications*, vol. 4, no. 1, 2018, pp. 1–11.
- [100] B. Munsky, G. Neuert, and A. Van Oudenaarden, “Using gene expression noise to understand gene regulation,” *Science*, vol. 336, no. 6078, 2012, pp. 183–187.
- [101] M. Kittisopikul and G. M. Süel, “Biological role of noise encoded in a genetic network motif,” *Proceedings of the National Academy of Sciences*, vol. 107, no. 30, 2010, pp. 13 300–13 305.
- [102] D. Frigola, L. Casanellas, J. M. Sancho, and M. Ibañes, “Asymmetric stochastic switching driven by intrinsic molecular noise,” *PloS one*, vol. 7, no. 2, 2012, p. e31407.
- [103] T. Biancalani, L. Dyson, and A. J. McKane, “Noise-induced bistable states and their mean switching time in foraging colonies,” *Physical review letters*, vol. 112, no. 3, 2014, p. 038101.
- [104] M. Assaf, E. Roberts, Z. Luthey-Schulten, and N. Goldenfeld, “Extrinsic noise driven phenotype switching in a self-regulating gene,” *Physical review letters*, vol. 111, no. 5, 2013, p. 058102.
- [105] G. Tkačik, A. M. Walczak, and W. Bialek, “Optimizing information flow in small genetic networks. III. A self-interacting gene,” *Physical Review E*, vol. 85, no. 4, 2012, p. 041903.
- [106] A. M. Walczak, G. Tkačik, and W. Bialek, “Optimizing information flow in small genetic networks. II. Feed-forward interactions,” *Physical Review E*, vol. 81, no. 4, 2010, p. 041905.
- [107] G. Tkačik, C. G. Callan, and W. Bialek, “Information flow and optimization in transcriptional regulation,” *Proceedings of the National Academy of Sciences*, vol. 105, no. 34, 2008, pp. 12 265–12 270.
- [108] J. M. Pedraza and A. van Oudenaarden, “Noise propagation in gene networks,” *Science*, vol. 307, no. 5717, 2005, pp. 1965–1969.
- [109] L. Arola-Fernández, G. Mosquera-Doñate, B. Steingger, and A. Arenas, “Uncertainty propagation in complex networks: From noisy links to critical properties,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, no. 2, 2020, p. 023129.
- [110] H. Zhang, Y. Chen, and Y. Chen, “Noise propagation in gene regulation networks involving interlinked positive and negative feedback loops,” *PloS One*, vol. 7, no. 12, 2012, p. e51840.

## REFERENCES (continued)

- [111] G. Hornung and N. Barkai, “Noise propagation and signaling sensitivity in biological networks: a role for positive feedback,” *PLoS Computational Biology*, vol. 4, no. 1, 2008, p. e8.
- [112] M. L. Simpson, C. D. Cox, and G. S. Saylor, “Frequency domain analysis of noise in autoregulated gene circuits,” *Proceedings of the National Academy of Sciences*, vol. 100, no. 8, 2003, pp. 4551–4556.
- [113] O. Milenkovic and B. Vasic, “Information theory and coding problems in genetics,” in *Information Theory Workshop*. IEEE, 2004, pp. 60–65.
- [114] K. Basso, A. A. Margolin, G. Stolovitzky, U. Klein, R. Dalla-Favera, and A. Califano, “Reverse engineering of regulatory networks in human B cells,” *Nature Genetics*, vol. 37, no. 4, 2005, pp. 382–390.
- [115] S. Hooshangi and R. Weiss, “The effect of negative feedback on noise propagation in transcriptional gene networks,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 16, no. 2, 2006, p. 026108.
- [116] Y. Li, Y. Li, H. Zhang, and Y. Chen, “MicroRNA-mediated positive feedback loop and optimized bistable switch in a cancer network involving mir-17-92,” *PLoS One*, vol. 6, no. 10, 2011, p. e26302.
- [117] T. S. Gardner, D. Di Bernardo, D. Lorenz, and J. J. Collins, “Inferring genetic networks and identifying compound mode of action via expression profiling,” *Science*, vol. 301, no. 5629, 2003, pp. 102–105.
- [118] J. Tegner, M. S. Yeung, J. Hasty, and J. J. Collins, “Reverse engineering gene networks: integrating genetic perturbations with dynamical modeling,” *Proceedings of the National Academy of Sciences*, vol. 100, no. 10, 2003, pp. 5944–5949.

## APPENDIX

## APPENDIX A

### ERROR PROPAGATION ANALYSIS

After a random perturbation or disturbance that introduces an error with probability  $\epsilon$  has occurred, we can describe the probability that an error exists on node  $x$  at the beginning of the  $l$ -th iteration in the form of equation (5.10). Under the assumption that the error events on  $x_a$  and  $x_r$  are independent and identically distributed (i.i.d),  $\Pr(x_a, x_r) = \Pr(x_a) \cdot \Pr(x_r)$ . To solve the first part on the right-hand side of equation (5.10a), i.e.,  $\Pr(\epsilon_x | x_a = 0, x_r = 0)$ , there is an error in  $x$ , if and only if (iff) both  $x_a$  and  $x_r$  have errors. Using the third Boolean function truth table in Table 4.1, only when both  $x_a \rightarrow 1$  and  $x_r \rightarrow 1$  does the output state change from the expected 0 state to 1. In this particular case, the input error on either  $x_a$  or  $x_r$  is not masked by the other. Hence, we can represent the first part of equation (5.10a) as follows:

$$\Pr(\epsilon_x | x_a = 0, x_r = 0) = \Pr(\epsilon_{x_a} | x_a = 0) \cdot \Pr(\epsilon_{x_r} | x_r = 0). \quad (\text{A.1})$$

Now consider the network configuration shown in Figure 5.2 with two activating edges and two inhibiting edges. It is assumed that an error is introduced with a positive probability  $\epsilon \ll 1$  by which the state of nodes can randomly be changed. Using the Boolean truth table for activation in Table 4.1, there will be an error in  $x_a$  if at least one of the edges  $x_1$  and  $x_2$  has an error, giving  $\Pr(\epsilon_{x_a} | x_a = 0) = 1 - (1 - \epsilon)^2$ . In general, for a node with  $k_a$  activating edges

$$\Pr(\epsilon_{x_a} | x_a = 0) = 1 - (1 - \epsilon)^{k_a}. \quad (\text{A.2})$$

Similarly, using the Boolean truth table for repression, the probability of error in  $x_r$  is computed by

$$\Pr(\epsilon_{x_r} | x_r = 0) = \frac{1}{2^{k_r}} [1 - (1 - \epsilon)^{k_r}], \quad (\text{A.3})$$

where  $k_r$  denotes the number of repression edges on a node. Subsequently, we can compute the probabilities  $\Pr(x_a = 0) = \frac{1}{2^{k_a}}$  and  $\Pr(x_r = 0) = \frac{2^{k_r} - 1}{2^{k_r}}$ . Hence, the expression for equation (5.10a) is given by

$$\begin{aligned} & \Pr(\epsilon_x | x_a = 0, x_r = 0) \cdot \Pr(x_a = 0, x_r = 0) \\ &= \frac{1}{2^{k_a}} \cdot \frac{2^{k_r} - 1}{2^{k_r}} \cdot \frac{1}{2^{k_r}} [1 - (1 - \epsilon)^{k_a}] [1 - (1 - \epsilon)^{k_r}] . \end{aligned} \quad (\text{A.4})$$

Next compute and evaluate the probability expression for equation (5.10b). From the third Boolean truth table in Table 4.1, observe that there will be an error in  $x$  iff  $x_r \rightarrow 1$  and no error on  $x_a$ . The probability of error in  $x_r$ , i.e.,  $\Pr(\epsilon_{x_r} | x_r = 0)$ , is given by equation (A.3). Using the activation and repression truth tables in Table 4.1 and considering the general case of a node with  $k_a$  and  $k_r$  input edges, compute the following probability of error:

$$\Pr(\epsilon_{x_a} | x_a = 1) = \frac{1}{2^{k_a}} [1 - (1 - \epsilon)^{k_a}] . \quad (\text{A.5})$$

Therefore,  $\Pr(\bar{\epsilon}_{x_a} | x_a = 1) = 1 - \Pr(\epsilon_{x_a} | x_a = 1)$ , where  $\bar{\epsilon}_{(\cdot)}$  denotes the event where there is no error. In addition, compute the probability  $\Pr(x_a = 1) = \frac{2^{k_a} - 1}{2^{k_a}}$  and then write the expression for equation (5.10b) as

$$\begin{aligned} & \Pr(\epsilon_x | x_a = 1, x_r = 0) \cdot \Pr(x_a = 1, x_r = 0) \\ &= \Pr(\bar{\epsilon}_{x_a} | x_a = 1) \cdot \Pr(\epsilon_{x_r} | x_r = 0) \cdot \Pr(x_a = 1, x_r = 0) \\ &= \frac{2^{k_a} - 1}{2^{k_a}} \cdot \frac{2^{k_r} - 1}{2^{k_r}} \left\{ 1 - \frac{1}{2^{k_a}} [1 - (1 - \epsilon)^{k_a}] \right\} \frac{1}{2^{k_r}} [1 - (1 - \epsilon)^{k_r}] . \end{aligned} \quad (\text{A.6})$$

Likewise, we can compute the probability of error in equation (5.10c) by

$$\begin{aligned} & \Pr(\epsilon_x | x_a = 0, x_r = 1) \cdot \Pr(x_a = 0, x_r = 1) \\ &= \Pr(\epsilon_{x_a} | x_a = 0) \cdot \Pr(\bar{\epsilon}_{x_r} | x_r = 1) \cdot \Pr(x_a = 0, x_r = 1) \\ &= \frac{1}{2^{k_a}} \cdot \frac{1}{2^{k_r}} [1 - (1 - \epsilon)^{k_a}] (1 - \epsilon)^{k_r} , \end{aligned} \quad (\text{A.7})$$

where an error results in  $x$  iff  $x_a \rightarrow 1$  and no error on  $x_r$ .

Finally, to solve equation (5.10d), note that there will be an error in  $x$  if at least one of the inputs in the third Boolean truth table of Table 4.1 has an error, i.e., the output

$x_a \wedge x_r$  changes state from 1 to 0. Hence, the first part of the right-hand side of equation (5.10d) can be expressed as:

$$\begin{aligned} & \Pr(\epsilon_x | x_a = 1, x_r = 1) \\ &= \Pr(\bar{\epsilon}_{x_a} | x_a = 1) \cdot \Pr(\epsilon_{x_r} | x_r = 1) + \Pr(\epsilon_{x_a} | x_a = 1) \cdot \Pr(\bar{\epsilon}_{x_r} | x_r = 1) \\ &+ \Pr(\epsilon_{x_a} | x_a = 1) \cdot \Pr(\epsilon_{x_r} | x_r = 1). \end{aligned} \quad (\text{A.8})$$

Furthermore, express the error probability  $\Pr(\epsilon_{x_r} | x_r = 1) = 1 - (1 - \epsilon)^{k_r}$ . Therefore, equation (A.8) yields

$$\begin{aligned} & \Pr(\epsilon_x | x_a = 1, x_r = 1) \\ &= \left\{ 1 - \frac{1}{2^{k_a}} [1 - (1 - \epsilon)^{k_a}] \right\} [1 - (1 - \epsilon)^{k_r}] + \frac{1}{2^{k_a}} [1 - (1 - \epsilon)^{k_a}] \\ &\times (1 - \epsilon)^{k_r} + \frac{1}{2^{k_a}} [1 - (1 - \epsilon)^{k_a}] [1 - (1 - \epsilon)^{k_r}]. \end{aligned} \quad (\text{A.9})$$

Then, we can express the probability of error in equation (5.10d) as

$$\begin{aligned} & \Pr(\epsilon_x | x_a = 1, x_r = 1) \cdot \Pr(x_a = 1, x_r = 1) \\ &= \frac{2^{k_a} - 1}{2^{k_a}} \cdot \frac{1}{2^{k_r}} \cdot \Pr(\epsilon_x | x_a = 1, x_r = 1). \end{aligned} \quad (\text{A.10})$$

Inserting equations (A.4), (A.6), (A.7), and (A.10) into equation (5.10) and after simplification, the total probability of error on node  $x$  is obtained as

$$\begin{aligned} \epsilon_l = \frac{1}{2^{2k_i}} & \left\{ 2^{k_i} (2^{k_a} - 1) - [(2^{k_a+1} - 1) (1 - \epsilon)^{k_a} - 3 \cdot 2^{k_a} + 4^{k_a} + 1] \right. \\ & \left. \times (2 - 2\epsilon)^{k_r} (2^{k_r} - 1) [1 - (1 - \epsilon)^{k_a} - 2^{k_a} + 4^{k_a}] [(1 - \epsilon)^{k_r} - 1] \right\}, \end{aligned} \quad (\text{A.11})$$

where  $k_i = k_a + k_r$ .