SENSITIVITY ANALYSIS OF DISCRETE MODELS AND APPLICATION IN BIOLOGICAL NETWORKS

by

Gaoxiang Zhou

B.S., Beihang University, 2016

Submitted to the Graduate Faculty of

Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science

University of Pittsburgh

2018

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This thesis was presented

by

Gaoxiang Zhou

It was defended on

April 4, 2018

and approved by

Natasa Miskov-Zivanov, Ph.D., Assistant Professor,

Department of Electrical and Computer Engineering

Zhi-Hong Mao, Ph.D., Associate Professor,

Department of Electrical and Computer Engineering

Samuel Dickerson, Ph.D., Assistant Professor,

Department of Electrical and Computer Engineering

Thesis Advisor: Natasa Miskov-Zivanov, Ph.D., Assistant Professor,

Department of Electrical and Computer Engineering

SENSITIVITY ANALYSIS OF DISCRETE MODELS AND APPLICATION IN BIOLOGICAL NETWORKS

Gaoxiang Zhou, M.S.

University of Pittsburgh, 2018

Understanding sensitivity is an important step to study system robustness against perturbations and adaptability to the environment. In this work, we model and investigate intracellular networks via discrete modeling approach, and we propose a framework to study sensitivity in these models. The discrete modeling approach assigns a set of discrete values and an update rule to each model element. The models can be analyzed formally or simulated in a deterministic or a stochastic manner. In our framework, we define element activity and sensitivity with respect to the state distribution of the modeled system. Previous sensitivity analysis approaches assume uniform state distribution, which is usually not true in biology. We perform both static and dynamic sensitivity analysis, the former assuming uniform state distribution, and the latter using a distribution estimated from stochastic simulation trajectories under a particular scenario.

Within our sensitivity analysis framework, we first compute element-to-element influences, then we extend the element update functions to include weights according to these computed influences. Adding weights to element interaction rules helps to identify key elements in the model and dominant signaling pathways that determine the behavior of the overall model. When studying cellular signaling networks, we are particularly interested in the response of elements to perturbations, as our goal is often to reach the desired model state via least number of interventions. We have applied our sensitivity analysis framework on pathway extraction and evaluation in the intra-cellular networks that controls T cell differentiation. Additionally, we propose four different ranking algorithms to extract the most

important pathways from a given source element to a given target element. We then evaluate these four algorithms using cross validation of corresponding extraction results. Our results show that, in different application occasions, different pathway extraction and evaluation algorithms should be adopted to help find "globally valid" or "globally effective" pathways.

Keywords: Discrete Modeling Approach, Static Sensitivity, Dynamic Sensitivity, Pathways Extraction, T-cell Differentiation.

TABLE OF CONTENTS

1.0 INTRODUCTION 1

2.0 BACKGROUND 4

2.1 Discrete Modeling Approach 4

2.2 Model Simulation 5

2.3 A Dependent Multi-valued Probabilistic Boolean Network 7

3.0 METHODOLOGY 12

3.1 Element Influence 13

3.2 Element Sensitivity 14

3.3 Static Analysis 14

3.4 Dynamic Analysis 15

4.0 COMPUTATION 17

4.1 Binary Decision Diagrams-based Influence Computation 17

4.2 An Improved ROBDD-based Influence Computation 20

5.0 APPLICATION 23

5.1 Pathways Extraction and Evaluation 23

5.2 Node Importance 25

5.3 Self-influence: An Indicator Of Loop Feedback 27

6.0 CASE STUDY: T-CELL DIFFERENTIATION 29

6.1 Element-level Analysis: Element Sensitivity 30

6.2 Interaction-level Analysis: Element Influence 31

6.3 System-level Analysis: Pathways Extraction 33

6.4 Scenarios Comparison 34

7.0 EXTRACTION ALGORITHMS 36
7.1 Four Extraction Evaluation Methods 37
7.2 Cross Validation of Pathways Extraction 38
7.2.1 Space Cross Validation 39
7.2.2 Time Cross Validation 42
8.0 FUTURE WORK 44
8.1 Possible Development 44
8.2 Current Bottleneck 45
9.0 CONCLUSION 46
BIBLIOGRAPHY 47

LIST OF TABLES

1 Truth tables of three PBN realizations of the toy example model 9

2 A traditional method to compute element influence using truth table 17

3 Comparison among numbers of active pathways under different scenarios . . . 35

LIST OF FIGURES

1 Discrete modeling approach and a comparison between different simulation schemes . . . 6

2 A grouped state transition diagram shows the steady state distribution is highly dependent on initial states . . . 10

3 Flowchart diagram of our sensitivity analysis framework . . . 12

4 A binary decision diagram sample and BDDs-based influence computation method . . . 18

5 A ROBDD shows its power to compute conditional node influence . . . 21

6 With the influence computation, weighted directed graphs are generated from the influence map according to state distributions . . . 23

7 A weighted directed graph $G(V, E, W)$ of a real biological model . . . 25

8 Finding out the node importance in a weighted directed graph . . . 26

9 Sensitivity distribution of all elements in T cell model under four scenarios . . . 30

10 Element influence matrix of T cell model under four different scenarios . . . 32

11 Pathways from *TCR_HIGH* to *FOXP3* under static analysis . . . 33

12 Pathways from *TCR_HIGH* to *FOXP3* under high-dose scenario . . . 34

13 Space cross ranking percentage of six system source-target pairs following four proposed methods under static analysis . . . 39

14 Space cross ranking percentage of four system source-target pairs following four proposed methods under high-dose scenario . . . 40

15 Space cross ranking percentage of four system source-target pairs following four proposed methods under low-dose scenario . . . 41

16 Time cross ranking percentage of extractions under different scenarios following four proposed methods 42

1.0 INTRODUCTION

The sensitivity analysis of a biological model indicates how sensitive the model and its elements can be to internal or external changes. Understanding sensitivity is an important step to study system robustness against perturbations and adaptability to the environment[1], both of which are considered indispensable for a living organism. Most of the previous sensitivity studies focused on the probabilistic Boolean networks (PBN)[2], with all nodes assigned a random update rule selected from several candidate Boolean functions. In this work, we model and investigate intra-cellular networks via discrete modeling approach, and we propose a framework to study sensitivity in these models.

Biochemical networks are often modeled as sets of reactions or reaction rules, and then analyzed using ordinary differential equations (ODEs)[3]. Several issues are commonly encountered when using reaction networks as models of intra-cellular networks. First, the reaction networks grow exponentially with the number of network components (receptors, ligands, kinases, etc.). Assuming that there are fast methods today that can be used to numerically solve these large sets of ODEs, the main obstacle that still remains is the lack of knowledge about all the network details. It is often the case that we are only familiar with indirect cause-effect relationships for some interactions in the network, and that we do not know exact mechanisms and the parameters necessary to create ODEs. The discrete modeling approach assigns to each model element a set of discrete values and a deterministic update rule according to its known direct or indirect regulators. The models can be analyzed formally or simulated in a deterministic or a stochastic manner. Different from the commonly used simultaneous (synchronous) update schemes[4], we model the stochasticity by applying random-order sequential update scheme[5], which is better suited for studying biological networks.

There is some prior research work on sensitivity analysis of biological networks. An influence matrix was introduced in [2] as an affiliation with state transition matrix. With the help of influence matrix, Markov chains were introduced in [4] and [6] to study the steady-state probability distribution. Additionally, [4] and [6] also proposed a general way to induce the network to reach the desired state. It was shown in [7] that the expected average sensitivity determines the well-known critical transition curve. Detailed proof was given in [8] to extend the results in [7] to networks with arbitrary connectivity K and to random networks with biased Boolean functions. Previous applications of sensitivity analysis in biological networks include network inference[9], intervention[10], and stochasticity/robustness modeling[11]. Taking sensitivity into account via a penalty term in the inference procedure improves the accuracy of predictions[9]. A long-term sensitivity was introduced in [10], with some experiments to show the method's performance in long-run intervention. Element influence was used in [11] as function failure probability, where the authors proposed the stochasticity in functions (SIF) model to study stochasticity in Boolean models.

In our framework, we define element influence and sensitivity with respect to the state distribution of the modeled system, using a discrete modeling approach. Previous sensitivity analysis approaches assume uniform state distribution, which is usually not true in biology. We perform both static and dynamic sensitivity analysis, the former assuming uniform state distribution, and the latter using a distribution estimated from stochastic simulation trajectories under a particular scenario. Under the DiSH simulator scheme[5], we are able to obtain sufficient trajectories to analyze the model. In addition, we also propose a Binary Decision Diagrams-based method to compute element influences. Within our sensitivity analysis framework, we first compute element-to-element influences, then we extend the element update functions to include weights according to these computed influences. Adding weights to these interaction rules helps to identify key elements in the model, as well as dominant signaling pathways that determine the behavior of the overall model.

To the best of our knowledge, previous sensitivity analysis research did not focus on detecting crucial pathways (elements regulations sequence) in a complicated Boolean network. For a well-studied or informative regulatory model with a large number of elements and complicated interactions, biologists are interested in extracting important pathways which

can dominate the control on a targeted element. In this work, we also discuss how these pathways can be extracted with the help of sensitivity analysis. We propose two scenarios to apply sensitivity. First, the static analysis approach assumes that all possible network states are equally distributed. Note that almost all the previous sensitivity research is based on this naive assumption. Second, we also investigate the dynamic approach where states are biased towards specific trajectories, as simulation can provide us the preferred network states under a given scenario, and the distribution of states which is closer to experimental observations. We then refine our pathways extraction method by improving the sensitivity scores propagation algorithm. This ensures the balance between long regulation pathways and short ones and gives more flexibility for different application occasions. In order to evaluate the extracted pathways, we also develop cross validation to assess that extractions are "globally valid" in the regulations of different targeted elements (validation in space) and are "globally effective" starting from different initial states (validation in time). We have applied our sensitivity analysis framework on pathway extraction and evaluation in the intra-cellular networks that controls T cell differentiation, and the examples from the T cell model are regulation[12] presented throughout the thesis.

In Chapter 2, we describe the background of discrete modeling approach and discuss and compare the model simulation schemes. In Chapter 3, we give the methodology details to define element influence and sensitivity and apply these definitions to both static and dynamic anaylsis. In Chapter 4, we propose a Binary Decision Diagrams-based computation method which fits the dynamic case quite well and also improve its complexity performance. In Chapter 5, we use our sensitivity analysis results to generate a weighted directed graph and give potential applications of this graph. In Chapter 6, we study the case of T cell differentiation model using sensitivity analysis and illustrate results in different analysis levels. In Chapter 7, we refine the pathways extraction algorithms to be adaptive to application occasions.

2.0 BACKGROUND

2.1 DISCRETE MODELING APPROACH

The construction of a model begins with identifying the key system components, and their interactions, usually through literature reading, data analysis or discussion with experts[13]. The extraction of this information from knowledge sources allows modelers to define the set of *model elements*, and for each element, the set of other elements that regulate it, as well as the polarity (positive or negative) of these regulations. The set of regulators is often called *influence set*, and the influence sets in a model can be illustrated as *influence map* (graph) $G(V, E)$, where nodes $\mathbb{V} = \{x_1, x_2, ..., x_N\}$ represent model elements, and edges E represent regulatory interactions between elements.

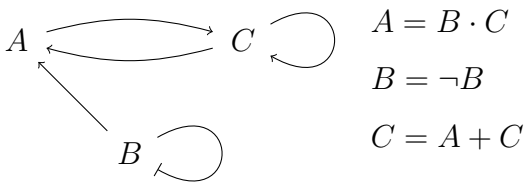
An influence map $G(V, E)$ alone is not sufficient to study the dynamics of the model. In order to create an *executable model*, it is necessary to assign update functions to a subset (or all) of model elements. In this work, we focus on discrete models. Therefore, we extend graph $G(V, E)$ to a discrete model, $M(V, F)$, where, for each model element, $x_i, i = 1, 2, ..., N$, we define its influence set, $VI_{x_i} \subset V$, $||VI_{x_i}|| = k_i$, which includes both positive regulators (activators) and negative regulators (inhibitors) of the element. We also define the number of all possible values of element x_i as n_i, representing the number of relevant discrete levels of activity of the element. In other words, x_i can only take values from the set $X_i = \{0, 1, ..., n_i - 1\}$. Finally, we define element update functions $F = \{f_1, f_2, ..., f_N\}$, where f_i is a discrete function mapping a k_i-dimensional non-negative vector to a non-negative integer in the set X_i. Boolean (logical) models are considered a special case of discrete models where the domain of all elements is $B = \{0, 1\}$, and the operators used in Boolean models include AND ("$\cdot$"), OR ("+"), and NOT ("$\neg$").

Using the logical model example in Figure 1(a), we illustrate the method that we use to study the dynamics of the modeled system. In this example, A, B, and C are model elements, and the influence sets of these three elements are $\{B, C\}$, $\{B\}$, $\{A, C\}$, respectively.

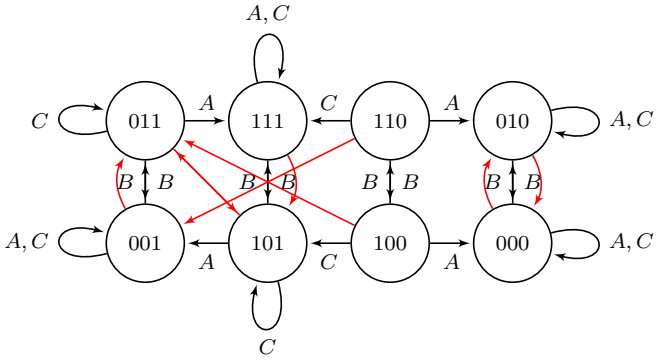
2.2 MODEL SIMULATION

Given a model $M(V, F)$ with all its elements and update functions, we can define a simulation scenario, and then simulate the model using the DiSH simulator[5]. Simulation *scenarios* are used to define: (1) initial values of all non-input model elements (i.e., nodes in the model graph that have arrows pointing at them), (2) initial values for all model inputs, and (3) when needed, perturbations that are assumed to happen at a particular model element, at a specified time point. The simulation is then executed following the scenario, from the initial state, until a pre-specified final state, which is indicated with the number of simulation time steps. One simulation run provides a trajectory of each model element between initial and final states.

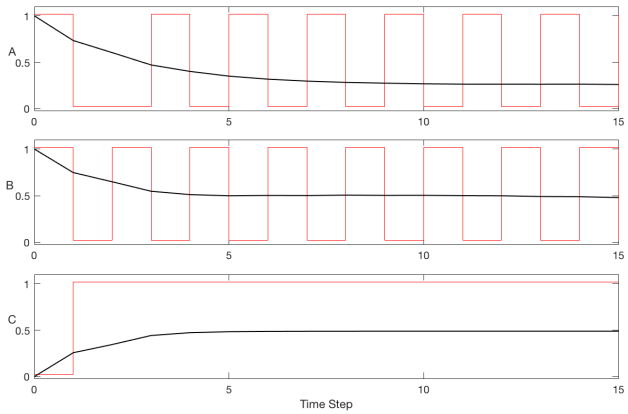
The simulation scheme that has been most often used to study logical models of biological networks is the simultaneous (synchronous) update scheme[2, 4, 6], where all elements are updated *simultaneously*, that is, current state values of all variables are used to simultaneously compute next state values. This simulation scheme is, therefore, deterministic, as for each state, there is only one possible next state that can be computed according to element update rules. However, in order to model stochasticity which plays an indispensable role in biological systems, in this work we use the *random sequential* update scheme from DiSH, in which, at a given simulation step, a randomly selected element is updated according to its update rule. For example, if the initial state of our example model above is 110 ($A = 1, B = 1, C = 0$), the simultaneous scheme will always lead to 001 as next state, while the random sequential scheme will lead either to state 010 (when element A is selected for update), to 100 (when element B is selected for update), or to 111 (when element C is selected for update). Therefore, the trajectories that elements follow from initial state to a given final state can vary in the random sequential case.



(a)



(b)



(c)

Figure 1: Discrete modeling approach and a comparison between different simulation schemes : (a) An influence map $G(V, E)$ and update functions for a small toy model of three elements A, B, C, they together forms an executable discrete model $M(V, F)$; (b) An STG of the toy model created for simultaneous and random sequential update schemes [5]; (c) Trajectories obtained from simulation for the toy model elements A, B, C, when the model is simulated using simultaneous (red line) and random sequential (black line) approach, from initial state $(A, B, C) = (1, 1, 0)$.

In Figure 1(b), we show the state transition graph (STG) for our example model. The red arrows in the figure indicate state transitions when the simultaneous update scheme is used, and black arrows indicate state transitions in the case of random asynchronous update scheme. In Figure 1(c), we show trajectories for elements A, B, and C obtained from simulation, from the initial state $(A, B, C) = (1, 1, 0)$ at the beginning of simulation, for 15 simulation steps. The red lines in the figure are simulation trajectories when the model is simulated using simultaneous approach, while the black lines show average trajectories obtained using the random sequential simulation scheme. Since the random sequential scheme returns trajectories that can vary for the same initial state, average trajectories are obtained by simulating the model multiple times from the initial state, and computing average element values at each simulation time step across all simulation runs. Further details about the simulation schemes can be found in [5].

2.3 A DEPENDENT MULTI-VALUED PROBABILISTIC BOOLEAN NETWORK

The Probabilistic Boolean Network (PBN) modeling approach is proposed in [2] to study the randomness of biological networks, from a perspective that is slightly different from the one described in Section 2.1, Section 2.2 and applied in [5]. Instead of assembling models using the information collected from experts or from literature[13], many previous studies have been done to estimate the structure of gene regulatory networks from gene expression data[9, 10]. In the latter case, the authors adopted the idea that one deterministic logic rule per gene may cause incorrect estimation results when inferring rules from gene expression measurements, as these measurements are sometimes noisy and the data size is not sufficient[2]. Therefore, they introduced a new model class called Probabilistic Boolean Networks (PBN). All model elements are assigned a random update rule from several candidate Boolean functions (defined as *predictors*) according to a pre-defined distribution. The way to select a set of predictors for a given model element is to employ the Coefficient of Determination[14], which is a method used on the element expression data samples.

The basic idea of PBNs is to accommodate more than one possible functions for each model element, that is, for each element x_j, there is a set of update rules $F_j = \{f_j^{(k)}\}, k = 1, 2, ..., l(j)$ where each $f_j^{(k)}$ is a possible update rule determining the value of element $x_j, j = 1, 2, ..., n$ and $l(j)$ is the number of candidate update rules of x_j. A realization of the PBN at a given time t, $R(t)$, is the choice of predictors for all model elements, which forms a BN at that time point. It's not hard to find that there are at most $N = \prod_{j=1}^{n} l(j)$ realizations. In general, the simulation of a PBN is the simultaneous state transition according to the realization $R_s(t), s = 1, 2, ..., N$ at time t. A PBN is defined to be independent if the choice of predictors for all model elements are independent from each other, in which the number of realizations reaches the maximum $N = \prod_{j=1}^{n} l(j)$. The number of realizations in a dependent PBN might decrease as the affection between the choice of predictors can prevent some realizations from happening.

Considering the discrete modeling approach and the random-order sequential update scheme we discussed in Section 2.2, our model can be viewed as a dependent simultaneous PBN such that there are two candidate update rules per element, i.e. (1) regulated by itself as buffer; (2) regulated by its pre-defined rule. The dependence within the model lies in the fact that if any model element follows Rule (2), all the other model elements will be updated according to their corresponding Rule (1). Therefore, the number of realizations in this dependent simultaneous PBN is n rather than 2^n. Taking the model in Figure 1 as an example, this dependent simultaneous PBN consists of three elements $V = \{A, B, C\}$, and the function sets $F = \{F_A, F_B, F_C\}$, where $F_A = \{f_A^{(1)} = A, f_A^{(2)} = B \cdot C\}, F_B = \{f_B^{(1)} = B, f_B^{(2)} = \neg B\}, F_C = \{f_C^{(1)} = C, f_C^{(2)} = A + C\}$. There are three realizations, that's $R_1 = \{f_A^{(2)}, f_B^{(1)}, f_C^{(1)}\}, R_2 = \{f_A^{(1)}, f_B^{(2)}, f_C^{(1)}\}, R_3 = \{f_A^{(1)}, f_B^{(1)}, f_C^{(2)}\}$ shown in the Table 1.

Finding the steady state distribution using PBN is an interesting question to address as the steady states (defined as *attractors*) often represent a cell type that has been reach under a particular scenario. The steady-state question can be addressed through the study of the state transition matrix A of the underlying Markov chains. The procedure to find A is described as follows: for each realization R_s as a Boolean Network (BN), we will have a one-step state transition binary matrix A_s, for each row (current state) in A_s, we compute n times to decide the next state (i.e. where to assign 1), the other entries in that row will be

Table 1: Truth tables of three PBN realizations of the toy example model

ABC	R_1	R_2	R_3
000	000	010	000
001	001	011	001
010	010	000	010
011	111	001	011
100	000	110	101
101	001	111	101
110	010	100	111
111	111	101	111
$p(R_s)$	1/3	1/3	1/3

automatically assigned 0. Since there are N realizations, we sum up all realizations weighted by their BN probabilities $p(R_s)$ to obtain the average state transition matrix. Thus, for an independent PBN, the computation complexity will be $O(n{\cdot}2^n{\cdot}N) = O(n{\cdot}2^n{\cdot}2^n) = O(n{\cdot}2^{2n})$, while for the discrete modeling approach we use together with the random-order sequential update scheme, in each row in a certain realization R_s, we only need to compute once to decide the next state. Also, there are only $N = n$ realizations in total. Therefore, the computation complexity of obtaining the state transition matrix A under discrete modeling approach is reduced to $O(1 \cdot 2^n \cdot N) = O(1 \cdot 2^n \cdot n) = O(n \cdot 2^n)$.

Another property of the state transition matrix A under discrete modeling approach is that it contains more zero entries compared to the state transition matrix under PBN modeling. For example, the state transition matrix of the toy example model in Figure 1 is

$$A = \begin{bmatrix} 2/3 & 0 & 1/3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2/3 & 0 & 1/3 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 2/3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 1/3 \\ 1/3 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 0 \\ 0 & 1/3 & 0 & 0 & 0 & 1/3 & 0 & 1/3 \\ 0 & 0 & 1/3 & 0 & 1/3 & 0 & 0 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 1/3 & 0 & 2/3 \end{bmatrix}$$

where the rows correspond current states, and the columns correspond next states, and states are ordered by $000, 001, 010, 011, 100, 101, 110, 111$. The entries denote the probabilities.

In linear algebra, sparse matrices tend to be nonsingular and therefore, don't possess eigenvectors which means that we cannot find a universal steady state distribution for the model. However, we can still group some states together to form a nonsingular group-based transition matrix and maintain the nonsingularity within the group as well. For example, we can rearrange the order of these eight states to $000, 010, 100, 110, 001, 011, 101, 111$ and obtain transition matrix as

$$A_{new} = \begin{bmatrix} 2/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 2/3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 1/3 \\ 0 & 0 & 0 & 0 & 2/3 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/3 & 1/3 & 0 & 1/3 \\ 0 & 0 & 0 & 0 & 1/3 & 0 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 2/3 \end{bmatrix}$$

and group states $(000, 010)$, states $(001, 011, 101, 111)$ together as shown in Figure 2.

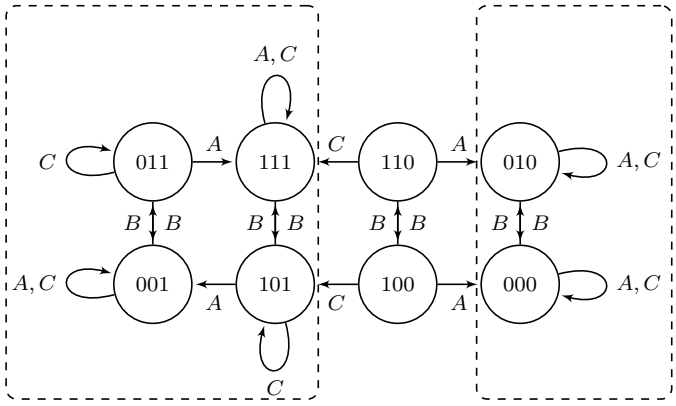


Figure 2: A grouped state transition diagram shows the steady state distribution is highly dependent on initial states

Now if we order the groups as $(000, 010), 100, 110, (001, 011, 101, 111)$, we can construct the new grouped state transition matrix A^*_{new} and inside-group state transition matrixes G_{left}, G_{right} as

$$A^*_{new} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1/3 & 1/3 & 0 & 1/3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, G_{left} = \begin{bmatrix} 2/3 & 1/3 & 0 & 0 \\ 1/3 & 1/3 & 0 & 1/3 \\ 1/3 & 0 & 1/3 & 1/3 \\ 0 & 0 & 1/3 & 2/3 \end{bmatrix}, G_{right} = \begin{bmatrix} 2/3 & 1/3 \\ 1/3 & 2/3 \end{bmatrix}$$

$A^*_{new}, G_{left}, G_{right}$ are all non-singular matrixes and possess a steady state distribution. Thus, due to bit changes occurring locally in random sequential scheme, if model starts within any state in the right dashed box $000, 010$, it will be trapped and form a Markov chain within the box, this Markov chain has a steady state distribution accordingly. The case will be the same when the model starts within any state in the left dashed box $001, 011, 101, 111$. In other words, instead of universal steady state distributions, we shall obtain initial-state-dependent steady state distributions, which is more informative for biology studies.

3.0 METHODOLOGY

In this chapter, we include the details of the methods that we have developed within our sensitivity analysis framework. We outline in Figure 3 the flow diagram of the framework.

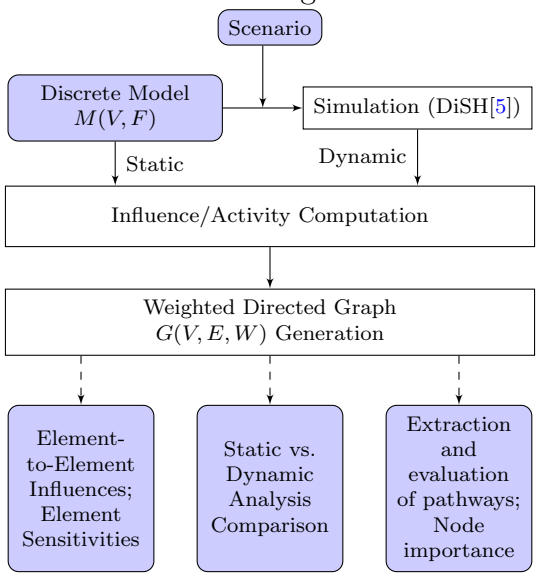


Figure 3: Flowchart diagram of our sensitivity analysis framework

We use as inputs the model $M(V, F)$, that is, defined sets V and F, and a scenario under which the model will be analyzed. The model definition is sufficient for the static sensitivity analysis (described in detail in Section 3.1, 3.2, 3.3), while the scenario definition is required for the dynamic sensitivity analysis (see Section 3.4). To obtain model trajectories for the dynamic sensitivity analysis, we run simulations using DiSH simulator[5]. Later in Chapter 4, we compute element influence based on Binary Decision Diagrams, implemented with CUDD package[15]. With the computation results, we extend discrete model $M(V, F)$ to weighted directed graph $G(V, E, W)$ and apply this graph to several studies in Chapter 5.

3.1 ELEMENT INFLUENCE

For a given set of model elements $\mathbb{V} = \{x_1, x_2, ..., x_N\}$, we are interested in computing a sensitivity of element x_j to changes in the value of element x_i, where $i, j \in \{0, 1, ..., N\}$, and $i \neq j$. To find the sensitivity of function $x_j = f_j(x_1, x_2, ..., x_N)$ to element x_i, we need to calculate the partial derivative of function f_j with respect to x_i. Since in this work we are focusing on logical models with Boolean variables and Boolean functions, the partial derivative is defined as an exclusive OR (XOR) of the co-factors of f_j with respect to x_i[1]:

$$\begin{aligned} \frac{\partial f_j}{\partial x_i} &= (f_j|_{x_i=0}) \oplus (f_j|_{x_i=1}) \\ &= f_{ji}(x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_N) \end{aligned} \tag{3.1}$$

In other words, $\frac{\partial f_j}{\partial x_i}$ does not depend on x_i, and can depend on any other model element $x_k \in V, k \neq i$, which is determined by the x_j's update function, f_j. Therefore, to find whether model element x_j can be influenced by x_i, we need to identify all possible values of vector $(x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_N)$, for which the partial derivative $\frac{\partial f_j}{\partial x_i}$ is true (i.e., equal 1). It can be seen from Equation (3.1), that the partial derivative will be equal 1 iff, for given values of $x_k, (k = 1, 2, ...i-1, i+1, ..., N)$, functions $f_j|_{x_i=0}$ and $f_j|_{x_i=1}$ have different values, which means that in such cases, function f_j changes when x_i changes. This is consistent with our intuition of x_j being sensitive to x_i, or we can say that there exist conditions under which model element x_i can influence model element x_j.

The influence/activity of element x_i in function f_j is defined as

$$\alpha_i^j = \alpha_i^{f_j} = \mathbb{E}(\frac{\partial f_j}{\partial x_i}) \tag{3.2}$$

This definition quantitatively describes the relationship between the regulator x_i and the regulated element x_j. Since the partial derivative $\frac{\partial f_j}{\partial x_i}$ itself is a Boolean function with only two possible values $\{0, 1\}$, we can use the expectation $\mathbb{E}(\frac{\partial f_j}{\partial x_i})$ as the probability that the change in the model element x_i flips the value of the function f_j (i.e., x_j), and hence, the influence (activity) α_i^j can vary between 0 and 1.

3.2 ELEMENT SENSITIVITY

The average sensitivity of a function f_j (or, of a element x_j) equals the sum of the activities of all its regulators:

$$s^{f_j} = \sum_{i=1}^{k_j} \alpha_i^{f_j} \tag{3.3}$$

On one hand, an element's sensitivity summarizes all the influences of its regulators, thus serving as an essential property of elements. On the other hand, it is important to note that element sensitivity is also dependent on the connectivity K (number of its regulators). In general, the more regulators one element has, the less is the influence of each of its regulators. Thus, the element sensitivity (sum in equation (3.3)) does not necessarily increase when the connectivity K grows. As shown in [1] and [8], if an element's average sensitivity is greater than 1, this property is critical in leading to instability of the model, and it enables the perturbation to propagate out of control. Other research[7] has also shown that, even with the same element sensitivity, unbalanced influence distribution of its regulators can make certain elements behave more stable and robust than elements with balanced regulator influence distribution.

3.3 STATIC ANALYSIS

We can denote the state of a Boolean model with N elements as an $N \times 1$ vector $\mathbf{x} = (x_1, x_2, ..., x_N)$, where each element $x_i \in \{0, 1\}$. If we assume that all possible states of the model are equally distributed, the influence of element x_i in the regulation of x_j can be expressed as

$$\alpha_i^j = \alpha_i^{f_j} = \mathbb{E}(\frac{\partial f_j}{\partial x_i}) = \frac{1}{2^{k_i}} \sum_{\mathbf{x}} \frac{\partial f_j(\mathbf{x})}{\partial x_i} \tag{3.4}$$

Thus, the larger the influence is, the more element x_j is sensitive to element x_i. As can be seen from equation (3.4), the sensitivity of model elements to changes in values of other

elements is determined by the fixed set of element update rules. In other words, the static sensitivity analysis approach relies solely on the update functions in the model, it assumes that the states of the system are uniformly distributed, and does not take into account dynamic element trajectories.

3.4 DYNAMIC ANALYSIS

The assumption that the states of the system follow a uniform distribution is usually not true in biology. Some states may never occur, or are not possible in living organisms. There are two important aspects of these systems that should be accounted for when conducting sensitivity analysis:

(1) The information about the system that is available is usually *not sufficient to derive the exact state distribution.*

(2) Depending on the scenario, that is, initial values, inputs, and perturbations, the *distribution of states varies.*

To tackle the challenge (1) above, we estimate the distribution of states through simulations. We can simulate the model for a pre-determined number of steps, and we use element trajectories that we obtain through simulations to derive the distribution of each model state. To tackle the challenge (2) above, we conduct simulations for all the initial states that are of interest for studying a particular system.

As described in Section 2.2, the choice of a simulation scheme determines whether we need to obtain a single or multiple trajectories from simulation for each initial state. When we use simultaneous simulation scheme, one trajectory is sufficient to compute the distribution of states in one scenario. This is due to the fact that the simultaneous scheme is deterministic, and thus, each state has only one next state, which is uniquely determined by model update functions. If the simulation is run on the same model for the same scenario multiple times, using random sequential simulation approach, the trajectories obtained will vary for most elements due to the stochasticity in the simulation approach. Although the number of transient states between the initial and the final state is finite, and the number of possible

next states from each model state is finite, the overall number of possible trajectories grows exponentially. Thus, we use a sample of all possible trajectories, by defining a number of times that the simulation is run for a particular scenario. Previous work has shown that even a smaller number of trajectories is sufficient to capture an average behavior for a given scenario[12], and therefore, we use the data from the sample simulation runs to estimate the state distribution in each scenario.

The dynamic trajectories obtained from simulation are highly dependent on the initial state, and therefore, the distribution from the sample trajectories will be different for different initial states. In the dynamic sensitivity analysis approach, the activity of element x_i in function f_j is defined by taking into account the occurrence probability, $p(\mathbf{x})$, of the state $\mathbf{x}$.

$$\alpha_i^{f_j} = \mathbb{E}(\frac{\partial f_j}{\partial x_i}) = \sum_{\mathbf{x}} \frac{\partial f_j(\mathbf{x})}{\partial x_i} p(\mathbf{x}) \tag{3.5}$$

$\mathbb{E}$ is the expected value taken with respect to $p(\mathbf{x})$. Note that it is possible that the non-zero activity under static analysis $\alpha_i^{f_j}$ is turned off to zero under dynamic analysis (i.e. $\alpha_i^{f_j} = 0$) since the states in which x_j is affected by the change in x_i may never occur in these dynamic trajectories.

4.0 COMPUTATION

4.1 BINARY DECISION DIAGRAMS-BASED INFLUENCE COMPUTATION

As shown in equation (3.4) and equation (3.5), the core part of influence computation is $\frac{\partial f_j}{\partial x_i}$, which is defined as an XOR of the co-factors of f_j with respect to x_i, i.e. $(f_j|_{x_i=0}) \oplus (f_j|_{x_i=1})$. It was proposed in [16] to study Boolean variable influence using a discrete N-dimensional cube representation. The basic idea is: 2^{n-1} minterms of $f_j|_{x_i=0}$ and $f_j|_{x_i=1}$ are listed, they count the number of different f values with respect to the same minterm and normalize the number by 2^{n-1}. Table 2 gives an example of the influence computation of element a on function $f = ab + a'c + bc'd$.

Although this idea is straightforward and efficient on the low size Boolean vector, it fails to address the problem of computing influence under non-uniform state distribution. Also, it becomes exponentially complex as the size of input Boolean vector increases.

Table 2: A traditional method to compute element influence using truth table

b, c, d	$f\|_{a=0}$	$f\|_{a=1}$	Different?
000	0	0	No
001	0	0	No
010	1	0	Yes
011	1	0	Yes
100	0	1	Yes
101	1	1	No
110	1	1	No
111	1	1	No

The Binary Decision Diagram (BDD) structure is a tree data structure that has been demonstrated to reduce computational complexity when manipulating with and evaluating

Boolean functions[17]. Let us assume that a given function f depends on a set of Boolean variables $x_1, ..., x_N$. To evaluate the function, given the values of these N variables, we start from the root node. In a decision tree, variables are ordered such that the tree can be traversed from its root node to the leaf nodes following this order. At the root node of a BDD tree, i.e. the first variable in the order, for example, x_1, there are two sub-trees, one for the case when $x_1 = 0$ (dashed line), and one where $x_1 = 1$ (solid line). Each of these two sub-trees is now a new BDD, and we can evaluate the next variable that is at the root of these two sub-trees. At the leaves of a binary decision tree, there are two nodes, 0 and 1, which represent the value of the function. Given the values of variables, the function BDD can be traversed following these values, and the value of the function is given by the value at the leaf node. In addition, we allow redundant evaluation of Boolean variables to be omitted, and allow sharing of identical sub-trees. Example BDD is given in Figure 4(a).

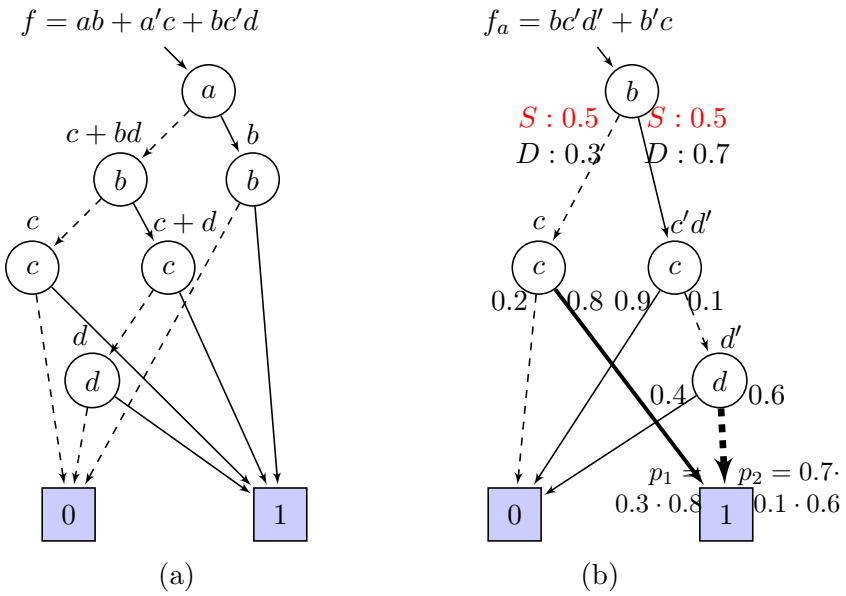


Figure 4: A binary decision diagram sample and BDDs-based influence computation method : (a) BDD of boolean function $f = ab + a'c + bc'd$ in the testing order of a, b, c, d; (b) To compute the influence of a in f, α_a^f, we construct the BDD of $f_a = b'c + bc'd'$

In the following, we propose our BDD-based influence computation method. As shown in equation (3.1), Boolean difference for function f_j, $f_{ji}(x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_N)$, is a new Boolean function with $n-1$ input variables. We can illustrate this new function f_{ji} in the form of a BDD, and using the diagram compute the expectation as in equation (4.1):

$$\begin{aligned} \mathbb{E}(\frac{\partial f_j}{\partial x_i}) &= \mathbb{E}(f_{ji}(x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_N)) \\ &= Pr(\boldsymbol{x} : f_{ji}(\boldsymbol{x}) = 1) \end{aligned} \tag{4.1}$$

We will use an example function $f = ab + a'c + bc'd$ to further explain our BDD-based method. In order to compute $\alpha_a^f = \mathbb{E}(\frac{\partial f}{\partial a})$, we should first construct $f_a = (f|_{a=0}) \oplus (f|_{a=1}) = (c+bd) \oplus b = b'c + bc'd'$. Note that we can directly read the co-factors from Figure 4(a) as the left child node and right child node of a. Then the BDD of f_a is constructed in Figure 4(b). Thus, equation (4.1) (i.e. $Pr(b, c, d : f_a(b, c, d) = 1)$) is the sum of probabilities of all paths ending at leaf "1" (shown as bold lines in Figure 4(b)). As each of these bold lines represents a path (or a set of paths) constituted from the root f_a to the leaf "1", we obtain

$$Pr(\boldsymbol{x} : f_{ji}(\boldsymbol{x}) = 1) = \sum_m p_m \tag{4.2}$$

where each joint probability p_m is computed through the path according to a given distribution. For the static sensitivity analysis approach described in Section 3.3, this will be a uniform distribution, shown with red numbers in Figure 4(b), that is, $p_1 = 0.5 \cdot 0.5, p_2 = 0.5 \cdot 0.5 \cdot 0.5$. For the dynamic sensitivity analysis approach, the distribution will most probably not be uniform, and will depend on a particular scenario that is studied. An example of a probability distribution for the dynamic case, p_1 and p_2, is shown with black numbers in Figure 4(b)).

As the number of bold lines ending in leaf node 1 is much smaller than 2^{n-1}, we greatly reduce the time and space complexity compared to the method shown in Table 2. Moreover, we can safely apply this method to dynamic sensitivity analysis such that these joint probabilities p_m are computed via conditional chain rules. In other words, if we associate each edge in the BDD with a probability, our problem of adding up joint probabilities p_m is converted into a traversal problem going from leaf "1" to root node f_a.

4.2 AN IMPROVED ROBDD-BASED INFLUENCE COMPUTATION

The efficiency of a BDD-based approach, in terms of reducing both the space and time complexity, is highly dependent on the order of evaluating variables within the BDD. Naturally, variables with high influence should be evaluated first to derive the decision tree towards high unbalance so that many redundant testing will be omitted. Therefore, Reduce Ordered Binary Decision Diagrams (ROBDDs) are proposed based on a fixed ordering of the variables and have the additional property of being reduced. It has been shown in [18] that building the ROBDD of a boolean function is NP-complete. Thus, once we obtain a ROBDD, we shall fully utilize it and manipulate operations on it rather than building another diagram.

Supposing that the variable evaluation order is given by $x_1 < x_2 < ... < x_{n-1} < x_n$, and we denote the two outgoing edges of a node v as $low(v)$ and $high(v)$, and the element(variable) name of node v as $var(v)$. A BDD is reduce ordered BDD[18] if

- **(uniqueness)** no two distinct nodes u and v have the same variable name and low-successor and high-successor, i.e.,

$$var(u) = var(v), low(u) = low(v), high(u) = high(v) \implies u = v \tag{4.3}$$

- **(non-redundant tests)** no node v has the same low-successor and high-successor, i.e.,

$$low(v) \neq high(v) \tag{4.4}$$

To compute the influence of all regulators in a Boolean function f, we follow two steps given an available ROBDD of function f:

(1) Find the conditional node influence: Recall in Equation (3.1) and Equation (3.2), we use α_i^f to denote the *influence of element* x_i *on function* f, which is the expectation of XOR between two co-factors of f with respect to x_i.

However, in a ROBDD, there might be many distinct nodes $x_{i_1}, ..., x_{i_{N(i)}}$ representing the same testing variable x_i. For each distinct node, we define the *conditional influence of node* $x_{i_m} (1 \leq m \leq N(i))$ *in function* f as the influence of x_i in function f given the input values of $x_1, x_2, ..., x_{i-1}$ that constitute the path from the root to the node x_{i_m}, where $N(i)$ is the number of distinct nodes testing the same variable x_i, that is:

$$
\begin{aligned}
\alpha^f_{i_m} &= \alpha^f_i|_{\{(x_1,x_2,...,x_{i-1}):\text{paths}(f\to x_{i_m})\}} \\
&= \mathbb{E}(\frac{\partial f}{\partial x_i}|\{(x_1,x_2,...,x_{i-1}) : \text{paths}(f \to x_{i_m})\}) \\
&= \mathbb{E}(f(x_i=0) \oplus f(x_i=1)|\{(x_1,x_2,...,x_{i-1}) : \text{paths}(f \to x_{i_m})\}) \\
&= \mathbb{E}(low(x_{i_m}) \oplus high(x_{i_m}))
\end{aligned}
\tag{4.5}
$$

Supposing the diagram in Figure 5 is a ROBDD of $f = ab + a'c + bc'd$. For convenience, we use the footnote to denote all the non-leaf nodes as a, b_1, b_2, c_1, c_2, d. For example, we can write the conditional influence of node b_2 in function f given that $a = 1$, as $\alpha^f_{b_2} = \mathbb{E}(\frac{\partial f}{\partial b_2}|a = 1) = \mathbb{E}(f(b = 0, a = 1) \oplus f(b = 1, a = 1)) = 1$

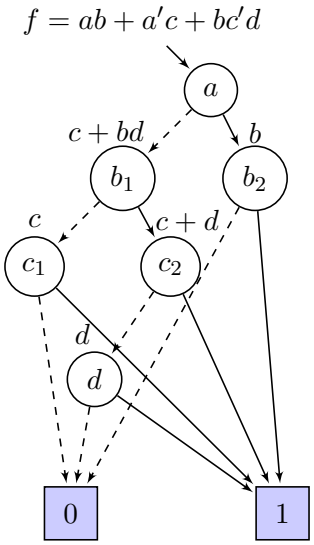


Figure 5: A ROBDD shows its power to compute conditional node influence

A tricky problem is what if there are multiple paths from root node to a certain non-leaf node in the diagram. In that case, the condition $\{(x_1, x_2, ..., x_{i-1}) : \text{paths}(f \to x_{i_m})\}$ should be conjunctions of several vectors $(x_1, x_2, ..., x_{i-1})$ that form paths towards node x_{i_m}.

As also shown in the last row of Equation (4.5), we can easily convert the problem of finding the *conditional influence of node* x_{i_m} *in function* f to the problem of finding the expectation of XOR between $low(x_{i_m})$ and $high(x_{i_m})$, which uses the similar idea in Section 4, but has much smaller size compared to the $n - 1$ size of $f|_{x_i=0}$ and $f|_{x_i=0}$.

(2) Traverse backward to obtain element influence: Recall the knowledge of conditional expectation $E(X) = \sum_{i=1}^{n} E(X|Y = y_i)Pr(Y = y_i)$. It's straightforward for us to obtain the *element influence* in terms of *conditional node influence* as follows,

$$\alpha_i^f = \sum_{m=1}^{N(i)} \alpha_{i_m}^f Pr\{(x_1, x_2, ..., x_{i-1}) : \text{paths}(f \to x_{i_m})\} \tag{4.6}$$

Note that in the formula of conditional expectation, events set $Y = y_i$ has to be mutually exclusive and collectively exhaustive. Recall the Taylor expansion of a boolean function

$$\begin{aligned} f &= x_1 f_{x_1=1} + x_1^{'} f_{x_1=0} \\ &= x_1 x_2 f_{x_1=1,x_2=1} + x_1 x_2^{'} f_{x_1=1,x_2=0} + x_1^{'} x_2 f_{x_1=0,x_2=1} + x_1^{'} x_2^{'} f_{x_1=0,x_2=0} \\ &= x_1 x_2 ... x_n f_{x_1=1,x_2=1,...,x_n=1} + ... + x_1^{'} x_2^{'} ... x_n^{'} f_{x_1=0,x_2=0,...,x_n=0} \end{aligned} \tag{4.7}$$

Thus when testing x_i, there should be 2^{i-1} distinct nodes in a complete expanding tree, that's $N(i) = 2^{i-1}, \sum_{m=1}^{m=2^{i-1}} Pr\{(x_1, x_2, ..., x_{i-1}) : \text{paths}(f \to x_{i_m})\} = 1$. But uniqueness endorses that nodes with the same testing variable $var()$ and the same outgoing edges $low()$ and $high()$ have been merged so that $N(i) < 2^{i-1}$ and now the number of paths going to x_{i_m} is more than 1. Thus we guarantee these conditional nodes are mutually exclusive nodes.

Moreover, non-redundant tests indicates that in ROBDD, there are some hidden nodes $x_{i_{hidden}}$ with $low(x_{i_{hidden}}) = high(x_{i_{hidden}})$ so that their values have no effect in determining the value of f, i.e., $\alpha_{i_{hidden}}^f = 0, \alpha_{i_{hidden}}^f \cdot Pr\{(x_1, x_2, ..., x_{i-1}) : \text{paths}(f \to x_{i_{hidden}})\} = 0$. These nodes are omitted in the ROBDD, and the corresponding terms are omitted in our formula given in Equation (4.6). Thus we also guarantee these conditional nodes are collectively exhaustive nodes.

Equation (4.6) therefore proves true for us to obtain element influence from conditional node influence. The probability $Pr\{(x_1, x_2, ..., x_{i-1}) : \text{paths}(f \to x_{i_m})\}$ can be addressed using the same idea as Equation (4.2) either under static or a dynamic distribution.

In summary, with the method proposed in this section, for example, we can achieve computing the influence of a, b, c, d in function f using $f_a(b, c, d), f_{b_1}(c, d), f_{b_2}(), f_{c_1}(), f_{c_2}(d), f_d()$, rather than having to construct $f_a(b, c, d), f_b(a, c, d), f_c(a, b, d), f_d(a, b, c)$ as stated in Section 4.1.

5.0 APPLICATION

5.1 PATHWAYS EXTRACTION AND EVALUATION

In studying biological systems, we are especially interested in the response of elements to perturbations, and furthermore, our goal is often to lead the model into a desired state via least number of interventions. Therefore, it becomes critical to develop methods to extract most influential or most active pathways from one model element (source) to another model element (target). Once we get these important pathways, we can easily control the model by tuning system input and by deciding whether to toggle elements during the transient process.

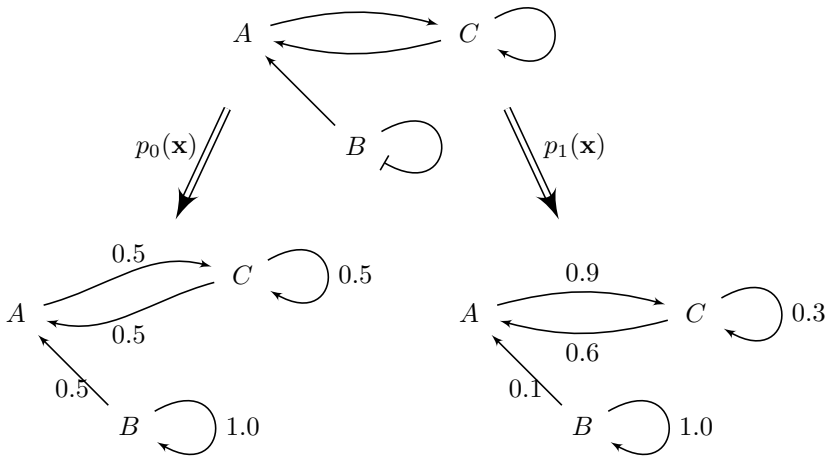


Figure 6: With the influence computation, weighted directed graphs are generated from the influence map according to state distributions

Within our sensitivity analysis framework, we are now able to extend the model interaction graph $G(V, E)$ to weighted directed graph $G(V, E, W)$ by adding weights $w_{ij} = \alpha_i^j$ to the directed edges pointing to x_j from x_i. Here α_i^j can be obtained either from static analysis as Equation (3.4) or dynamic analysis as Equation (3.5). Figure 6 shows two possible weighted directed graphs of previously discussed toy model. As can be seen, for a pre-defined discrete model $M(V, F)$, the interaction map $G(V, E)$ is fixed, while the weighted directed graph $G(V, E, W)$ can be varying with respect to $p(\mathbf{x})$, of the state $\mathbf{x}$, which is determined by simulation scenario.

As the weights associated with each edges represent the magnitude of influence, it is natural to say pathways with all high influence edges are more influential and active. For a weighted directed graph generated from a real biology influence map of huge size(e.g., Figure 7), it's quite important for us to choose a efficient and complete algorithm to find all the pathways connecting a given source and target node. To guarantee the completeness, we choose BFS algorithm in our model to explore pathways. We also keep track of the visited set for each path to avoid cycles. Among these pathways, we need a ranking method to evaluate them for biological study.

Suppose there is a regulatory pathway $\mathcal{P} = \{x_1, ..., x_K\}$, where x_1 is the input(source node) and x_K is the output(target node). We assign a score $S_\mathcal{P}$ to this pathway, which is defined as the sum of log of the activity of each input/output pair along the pathway:

$$S_\mathcal{P} = \sum_{i=1}^{K-1} log(\alpha_i^{f_{i+1}}) = \sum_{i=1}^{K-1} log(\alpha_i^{i+1}) \tag{5.1}$$

It's not hard to find that $e^{S_\mathcal{P}}$ is just the multiplication of activities along the pathway, and this multiplication result essentially reflects the propagated probability effect if we assume state distribution independence across different levels. The probability multiplication gives us a magnitude of how possible the source node affects the target node in the long-run style. The higher is the score of one pathway, the more dominant this pathway can be. Equation (5.1) is proposed to understand the influence propagation from the perspective of probability. We will refine the algorithm in Chapter 7 to other forms for other applications.

For example, in Figure 7, we'd like to extract all the pathways from *TCR* to *FOXP3*, we can obtain four non-cycle pathways as

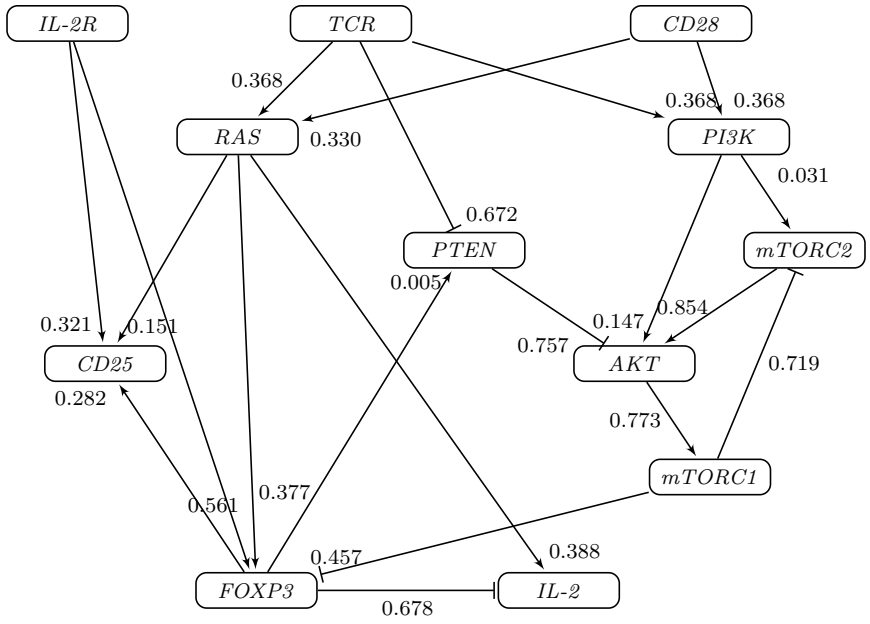


Figure 7: A weighted directed graph $G(V, E, W)$ of a real biological model

Pathway 1: $(TCR \rightarrow RAS \rightarrow FOXP3)$

Pathway 2: $(TCR \rightarrow PTEN \rightarrow AKT \rightarrow mTORC1 \rightarrow FOXP3)$

Pathway 3: $(TCR \rightarrow PI3K \rightarrow AKT \rightarrow mTOCR1 \rightarrow FOXP3)$

Pathway 4: $(TCR \rightarrow PI3K \rightarrow mTORC2 \rightarrow AKT \rightarrow mTOCR1 \rightarrow FOXP3)$

And pathway $(TCR \rightarrow RAS \rightarrow FOXP3)$ has the highest score according to Equation (5.1).

5.2 NODE IMPORTANCE

With the score of each pathway, we can find out which nodes play the most important role in regulation from a source node to a target node.There are two aspects that decide the importance of a node. Firstly, we look at the number of occurrences of the node in all possible pathways that link the source and the target. The more it occurs, the more important the node is to this regulatory relationship. Secondly, we take into account the score of each pathway $S_{\mathcal{P}}$. Each occurrence is weighted by $e^{S_{\mathcal{P}}}$. In other words, an node that

shows up in a high-scoring pathway should be more important than those showing up in a low-scoring pathway. Thus we define the importance of node v in the relationship $x \to y$ as

$$\mathcal{I}_v^{x \to y} = \sum_{p \in \mathcal{P}} e^{S_{\mathcal{P}}} \cdot \mathbb{I}(v \in p) \tag{5.2}$$

$\mathbb{I}()$ is an indicator function which returns 1 if the statement is true, and 0 otherwise. When $S_{\mathcal{P}}$ is defined as Equation (5.1), node importance can be viewed as the expected number of occurrence of the element v in the $x \to y$ relationship.

For example, in Figure 8, we show nodes(as dashed nodes) which can be possibly visited when exploring all non-cycle pathways from *TCR* to *FOXP3* and separate their non-cycle interactions. As can be seen, although *RAS* occurs in the pathway with highest score, its role is not so important as *AKT* and *mTORC1* which occur almost in every pathway. According to Equation (5.2), the importance of these six dashed nodes is ranking as (*AKT*=*mTORC1*>*RAS*>*PTEN*>*PI3K*>*mTORC2*), indicated by the thickness of node border in Figure 8.

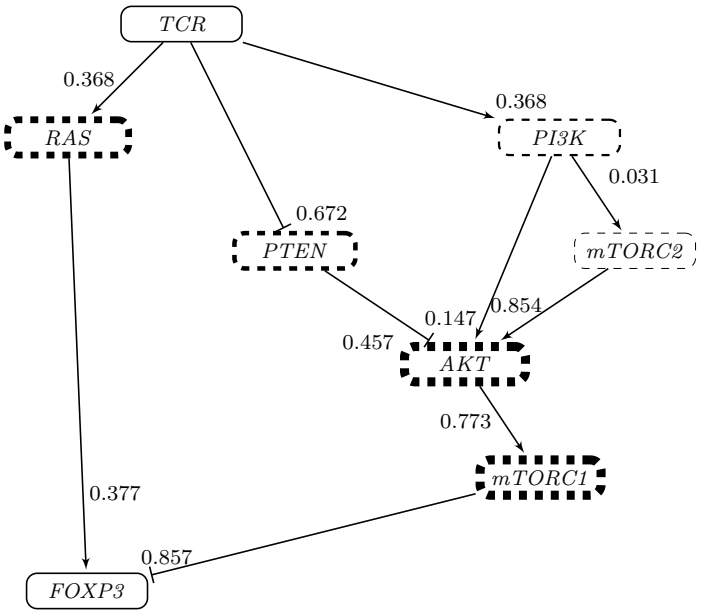


Figure 8: Finding out the node importance in a weighted directed graph

5.3 SELF-INFLUENCE: AN INDICATOR OF LOOP FEEDBACK

As discussed in Section 5.1, we avoid to expand any repeated nodes to reduce the computation complexity. However, biology models are always affected by many feedback to regain the stability, giving up all repeated nodes cuts the opportunities to look into the details of loop feedback. We may recover these missing information with *self-influence*. Equation (3.4) and Equation (3.5) give the computation of the influence x_i in x_j, we can even perform this computation between all pairs of model elements and therefore a $n \times n$ matrix $I : I_{ij}$ is constructed such that $I_{ij} = \alpha_i^j$. If x_i isn't a regulator of x_j, $\alpha_i^j = 0$.

Looking into the structure of influence matrix I, it's possibly not symmetric and has zero entries on the diagonal except these elements which are regulated by themselves. But we can go one step further to $I^2, I^3, ...$, which gives the propagated influence within two direct regulations, three direct regulations or more. It's not surprised that the number of non-zero diagonal entries will increase as we multiply I by itself to obtain $I^2, I^3, ...$ These non-zero entries represents the influence in itself, indicating the affection of loop feedback.

The self-influence can be used to answer two types of questions:

(1) For a certain length feedback, supposing I^k, which element is most influential by itself? The solution is just the element with highest diagonal entry in I^k. In both biology control and circuit design, loop feedback is typically time sensitive, the longer time one feedback signal takes, the less effective it will be. So we can safely limit our research scope within feedback of a certain length. These top elements with highest diagonal entries in I^k serve as indicator how the loop effect distributes across the model space, which helps biologist isolate parts of model for further study.

(2) For a certain element x_i, which loop feedback should be considered first? We may first compare the i-th entry in $I, I^2, I^3, ...$ to find the largest one, supposing it's $I^{sol(i)}$, this tells that feedback of length $sol(i)$ is quite important to element x_i, we then head to find such kind of feedback in the model. Recall that we avoid repeated elements in BFS algorithm, we may replace it with a smarter method in which we assign every element x_i with a threshold $sol(i)$ to denote the maximum times of duplicated visits. This includes loop when extracting the pathways as well as limit the computation complexity.

For example, for the model in Figure 7, the influence matrix is given by

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.321 & 0 & 0 & 0.561 & 0 \\
0 & 0 & 0 & 0.368 & 0.368 & 0.672 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.330 & 0.368 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.151 & 0 & 0 & 0.377 & 0.388 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.031 & 0 & 0.147 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.757 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.854 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.773 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.719 & 0 & 0 & 0 & 0.457 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.005 & 0 & 0.282 & 0 & 0 & 0 & 0.678 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

where the rows and columns correspond the elements ordered by *IL-2R*, *TCR*, *CD28*, *RAS*, *PI3K*, *PTEN*, *mTORC2*, *CD25*, *AKT*, *mTORC1*, *FOXP3*, *IL-2*.

Within loops of 6 interactions or less, *PTEN*, *mTORC2*, *AKT*, *mTORC1* and *FOXP3* have non-zero self-influence, of which *AKT* and *mTORC1* have the highest self-influence. This results correspond two important system loops (*AKT*, *mTORC1*, *mTORC2*) and (*AKT*, *mTORC1*, *FOXP3*, *PTEN*).

For elements *AKT*, which loop is more important? We can just compare the ninth entry(AKT) among $I, I^2, I^3, ...I^Z$, we find that the ninth entry of I^3, I^4 and after I^6 is non-zero. Moreover, the ninth entry of I^3 is greatest among them, which indicates that (*AKT*, *mTORC1*, *mTORC2*) is dominant feedback. If we allow this loop when extracting the pathways, the extraction results will be more reasonable.

6.0 CASE STUDY: T-CELL DIFFERENTIATION

T cells, one of two primary types of lymphocytes, play central roles in cell-mediated immunity, which does not involve antibodies, but rather involves the activation antigen-specific cytotoxic T-lymphocytes and the release of various cytokines in response to an antigen. There are several subsets of T-cells and each one has a distinct function in T-cell mediated immunity.

Generally speaking, T cells can be differentiated into two subsets: (1) regulatory cells (T_{reg}), which mainly suppress T-cell mediated immunity and reduce the damage caused by autoimmune response; (2) helper cells(T_H), which assist other lymphocytes in the mediation of immune response. Previous research[19] has shown that these two types of T-cells are distinguished by different element expressions in the molecular level. For example, in T_{reg} type, the transcription factor forkhead box P3 *FOXP3* is expressed and Interleukin-2 *IL-2* is inhibited, while in T_H type, *FOXP3* is inhibited and *IL-2* is activated.

In [12], the circuitry that controls the differentiation of T cells is modeled using the logical modeling approach. Some model elements are implemented as discrete (not Boolean variables) with values $\{0, 1, 2\}$, to denote absence, low activity and high activity of the element, respectively. Three discrete levels can be encoded with two Boolean variables in order to use a logical model. For example, the T-cell receptor (*TCR*) is modeled with two variables, *TCR_LOW* and *TCR_HIGH*, such that:

$TCR{=}0 : (TCR\_LOW{=}0, TCR\_HIGH{=}0)$;

$TCR{=}1 : (TCR\_LOW{=}1, TCR\_HIGH{=}0)$;

$TCR{=}2 : (TCR\_LOW{=}0, TCR\_HIGH{=}1)$;

In this model, the input nodes are *TCR_LOW*, *TCR_HIGH*, *CD86*, *IL-2 EX*, $TGF{-}\beta$ and *PI3K*. The output nodes are: *FOXP3*, *IL-2*, *mTOCR1*, *mTOCR2* and *PTEN*.

As we have mentioned in Section 3.4, scenarios have to be defined to conduct dynamic sensitivity analysis. We are particularly interested in three scenarios: high-dose scenario with initial value TCR=2; low-dose scenario with initial value TCR=1; toggle scenario with initial value TCR=2 but regulated down to TCR=0 after short time.

6.1 ELEMENT-LEVEL ANALYSIS: ELEMENT SENSITIVITY

For the T cell model, the interaction map $G(V, E)$ is fixed, while the weighted directed graph $G(V, E, W)$ is varying with respect to state distribution, which is determined by simulation scenario. Under different scenarios, elements have different sensitivities.

Figure 9 shows the sensitivity of 55 elements under four different scenarios, where these elements in the T cell differentiation are sorted alphabetically, four different scenarios are the static analysis, and the dynamic analysis with high-dose, low-dose and toggle scenario.

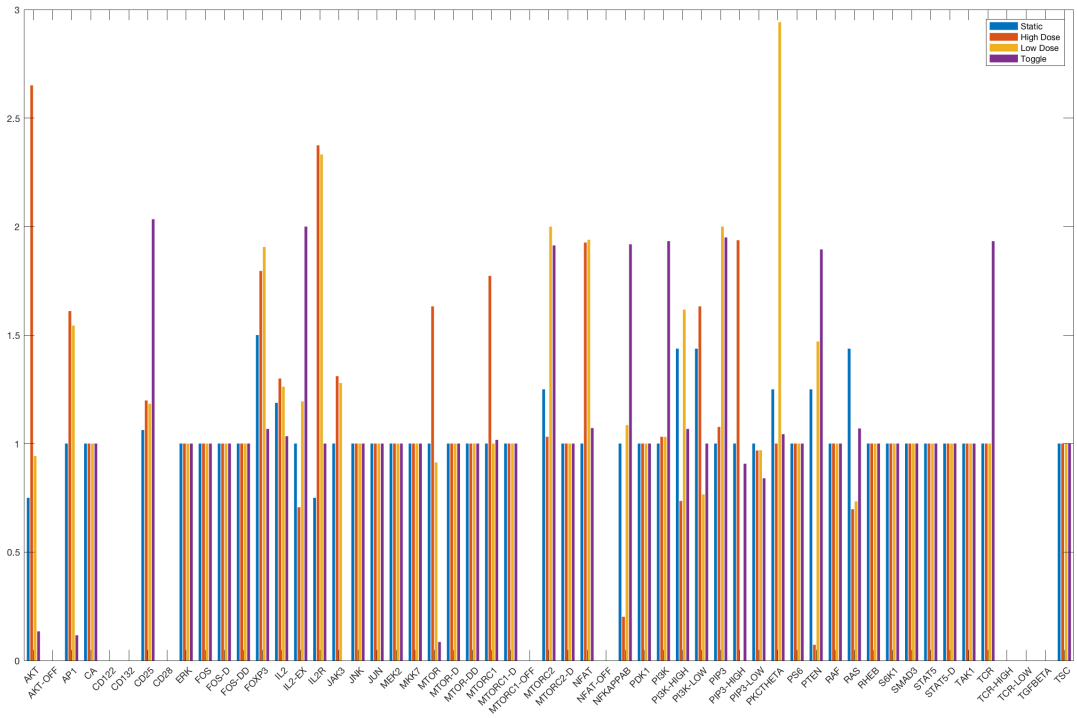


Figure 9: Sensitivity distribution of all elements in T cell model under four scenarios

In all scenarios, we can see that there are 9 elements having zero sensitivity, which corresponds to the system input elements and some specific parametric elements that are not regulated by others.

The result for static analysis shows less variance, ranging from 0 to 1.5. Most elements have sensitivities less than 1, indicating that this network follows a stable and ordered structured behavior[1]. However, the behavior of elements under dynamic analysis shows greater variance, ranging from 0 to 2.94. Some elements under dynamic analysis have sensitivities much greater than 1 (e.g. element *PKCTHETA* under low-dose scenario), which could lead to local instability.

Top three elements that behave differently under different scenarios are *AKT*, *PIP3*, *PKCTHETA*, most of which are cell type markers. They play dominant roles in the whole network.

6.2 INTERACTION-LEVEL ANALYSIS: ELEMENT INFLUENCE

Apart from neighboring elements, we are also interested in element-to-element influence of two arbitrary elements. To achieve that, we first need to find all the pathways from one element to the other, then summarize all the pathway effects. As defined in equation (5.1), the score of the pathway $S_{\mathcal{P}}$ reflects the propagated probability. If we add the scores of all possible pathways from one element to the other, this summation shows the overall influence that the source element has on the target element. Figure 10 shows the element-to-element influence matrix under four scenarios, where rows correspond to the source elements and columns correspond to the target elements.

Note that the color bar in the top-left corner is ranging from 0 to 1, while others are ranging from 0 to 2. In general, element influence under three dynamic scenarios are greater than the element influence under the static analysis. For one reason, under the static analysis, the long-run element-to-element influences are quite sensitive to the length of pathways since the edge weight α under static analysis is relatively small. For another reason, dynamic scenarios are obtained from real biological observations and thus show stronger homogeneity.

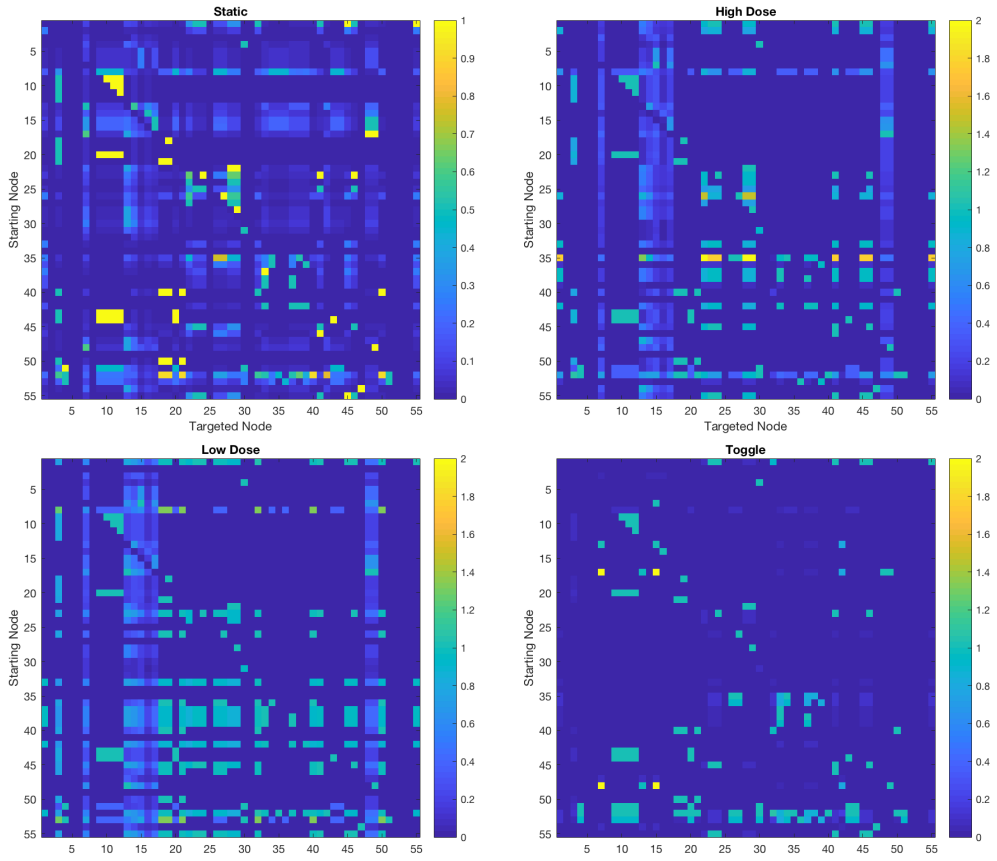


Figure 10: Element influence matrix of T cell model under four different scenarios

Additionally, it is worthwhile to note that compared to static analysis, the entries in element influence matrix of three dynamic scenarios are distributed with high unbalance (i.e. with wild variance range and high deviation), especially that of toggle case. As the system with unbalanced influence distribution is said to behave stable and robust[7], the results in Figure 10 inspire that we may toggle some nodes within the transient process for the purpose of driving system to certain states and maintaining stability as well.

Intuitively, a high element influence indicates a strong interaction (in other words, an important pathway which dominates the regulation pairs). From Figure 10, we can easily find some strong element interactions such as (*PI3K* to *mTOR*) in high-dose scenario, (*CD28*, *TCR*) to (*JNK*, *JUN*, *MKK7*, *NFKAPPAB*, *PKCTHETA*, *TAK1*) in low-dose scenario.

6.3 SYSTEM-LEVEL ANALYSIS: PATHWAYS EXTRACTION

To get insight into global influences, we perform system-level sensitivity analysis with the system input as source and system output as target. Figure 11 shows all 306 pathways in a heatmap from *TCR_HIGH* to *FOXP3* under static analysis, where rows correspond the different pathways and columns are all elements ranked in the order of node importance (not in alphabetical order anymore). A pink block denotes presence in the pathway, while a green block denotes absence. In additional, rows have been clustered by their similarities.

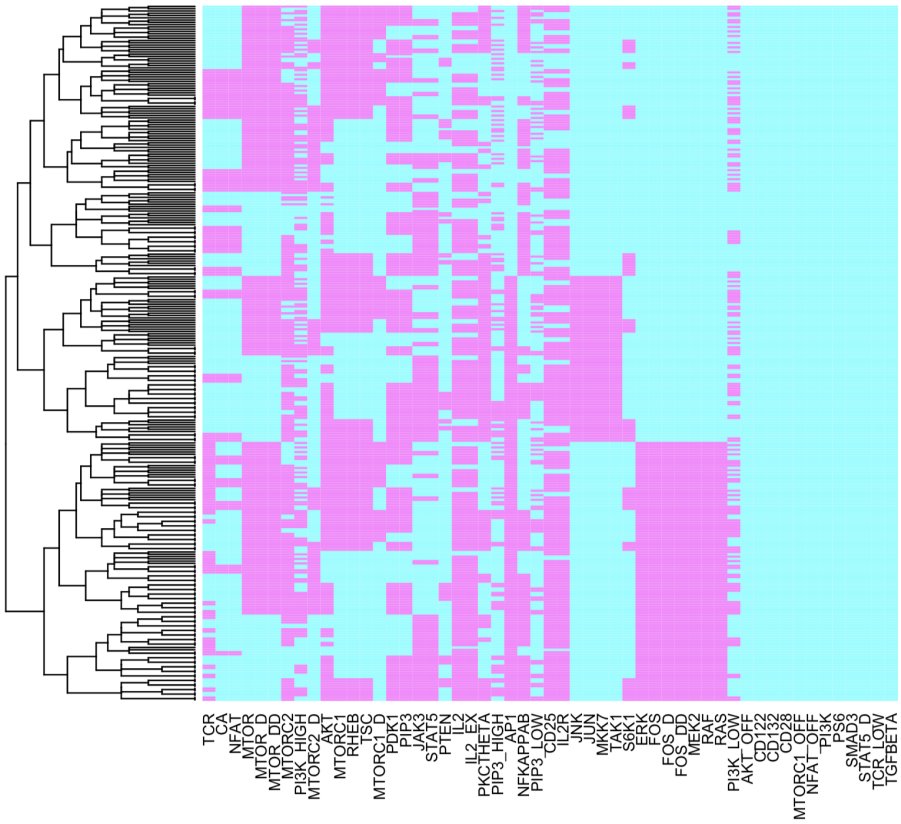


Figure 11: Pathways from *TCR_HIGH* to *FOXP3* under static analysis

We do not plot all the pathways (rows) by the order of pathway scores calculated in equation (5.1), instead we plot the rows to form cluster according to their Hamming distance so that we can easily recognize groups of patterns. The clustering information in heatmap helps detect blocks within the model, which can furthermore separate and simplify the

complex network. We should also note that, if we cluster these pathways according to node-importance-weighted Hamming distance rather than the naive Hamming distance, the heatmap will tell more about the pattern recognition priority.

6.4 SCENARIOS COMPARISON

Sensitivity analysis also offers us an opportunity to compare between different scenarios. This is mainly because some none-zero activities under static analysis $\alpha_i^{f_j}$ can be turned off to zero under dynamic analysis (i.e. $\alpha_i^{f_j} = 0$). Therefore, some active pathways in static analysis become inactive in dynamic scenario and regulations following these paths are no longer effective. To illustrate the difference, Figure 12 shows all 18 active pathways(with all non-zero activities) from *TCR_HIGH* to *FOXP3* under high-dose scenario, with columns representing elements in the same order as Figure 11.

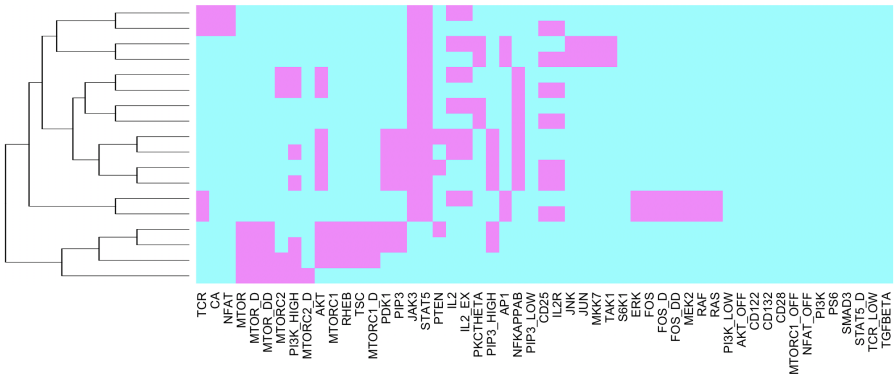


Figure 12: Pathways from *TCR_HIGH* to *FOXP3* under high-dose scenario

Obviously the number of active pathways decreases a lot, whereas the group of patterns is still easily detected. The large decrease in the number of active pathways is common in all connections between system inputs and outputs. Table 3 gives a detailed comparison among the numbers of active pathways under static analysis, high-dose scenario and low-dose scenario.

Table 3: Comparison among numbers of active pathways under different scenarios

From	*TCR_HIGH*		*IL-2 EX*		*CD86*	
To	*FOXP3*	*IL-2*	*FOXP3*	*IL-2*	*FOXP3*	*IL-2*
Static	306	196	44	155	214	133
High-dose	18	18	1	1	9	9
Low-dose	15	15	1	1	10	10

As shown in Table 3, the dynamic scenario shows its power in reducing the number of active pathways. More interestingly, we find that there is only one active pathway from *IL-2 EX* to *IL-2* under high-dose/low-dose scenarios(i.e. *IL-2 EX*→*JAK3*→*STAT5*→*FOXP3*→*IL-2*). Also, under high-dose and low-dose scenarios, the number of pathways to regulate *FOXP3* and *IL-2* are always the same, no matter what is the source node. This indicates that *FOXP3* exists almost everywhere in the regulation that control *IL-2*.

7.0 EXTRACTION ALGORITHMS

The algorithms proposed in Section 5.1 provide us an insight into the model from the perspective of probability. Since element influence is always less than 1, an accumulative algorithm to assign pathways scores is always giving preference to shorter pathways, whereas longer pathways are typically these pathways we are pretty familiar with, and meanwhile, there might be some hidden interactions within the shorter pathways. In addition, the purpose of extracting pathways from a model is not the same for all application occasions. In some circumstances, we are interested in structure-based pathways regardless of the transient states, in other circumstances, we may need to observe how different pathways couple each other.

Therefore, in this chapter we refine our extraction algorithms to some alternative methods and introduce cross validation to evaluate these extractions as well. Cross validation, on one hand, can help verify mathematically and biologically whether the extractions are valid and effective. On the other hand, with cross validation, we are able to identify which algorithms perform better for a certain application occasion.

7.1 FOUR EXTRACTION EVALUATION METHODS

Method 1 (Accumulative): as mentioned in Section 5.1 and Section 6.3, we can regard the element activity(influence) as the probability that the value of one element is flipped as its regulator's value changes. From this perspective, it's nature to assign pathway a score such that it equals the probability that the source element's flipping cause the target element's change. Thus for a regulatory pathway $\mathcal{P} = \{x_1, ..., x_K\}$, where x_1 is the input(source node) and x_K is the output(target node), we define the score of a pathway as Equation (7.1).

$$S_{\mathcal{P}_1} = \sum_{i=1}^{K-1} log(\alpha_i^{i+1}) \tag{7.1}$$

Method 2 (Normalized): Method 1 always prefers shorter pathways since shorter pathways have a higher end-to-end probability. It degrades longer pathways, thus disregards these long regulatory relationships. However, long regulatory pathways are these well-studied interactions and there might be some potential interactions added to these short pathways later. To address this, we normalize the above pathways scores by the length of the pathways. Thus we define the score of a pathway as Equation (7.2).

$$S_{\mathcal{P}_2} = \frac{\sum_{i=1}^{K-1} log(\alpha_i^{i+1})}{K} \tag{7.2}$$

Method 3 (Weighted Normalized): For a given source-target pair, considering the timing scale, pathways with high influence near the target element should be paid more attention than these pathways with high influence far away from the target node. So instead of simply normalizing the pathway scores by the length of pathway, we assign weights to the element-to-element activity (influence) along the pathway such that the closer element-to-element activity (influence) has a larger weight. Thus we define the score of a pathway as Equation (7.3), where we suppose weights w_i are linearly increasing from the source node to target node.

$$S_{\mathcal{P}_3} = \frac{\sum_{i=1}^{K-1} w_i log(\alpha_i^{i+1})}{\sum_{i=1}^{K-1} w_i} \tag{7.3}$$

Method 4 (Weighted Normalized with Eliminating Delay Variables): In the T cell model purposed by [12], there are some delay variables as buffers which are created to denote the different delay time occurred in the regulation. However, the element influences between delay variables is always 1 and count nothing towards the pathway scores, but in normalized method, we normalize the accumulative score by the total length, which prefers pathways with lots of delay variables. To address it, we add the feature of eliminating delay variables to Method 3 (Weighted Normalized). Thus we define the score of a pathway as Equation (7.4), where $\mathbb{I}(v_i \in DV)$ is an indicator function which returns 1 if the statement (node v_i is a delay variable) is true, and 0 otherwise.

$$S_{\mathcal{P}_4} = \frac{\sum_{i=1}^{K-1} w_i log(\alpha_i^{i+1}) \cdot \mathbb{I}(v_i \in DV)}{\sum_{i=1}^{K-1} w_i \cdot \mathbb{I}(v_i \in DV)} \tag{7.4}$$

7.2 CROSS VALIDATION OF PATHWAYS EXTRACTION

In order to verify whether the pathway extractions are valid and effective, and to get a better understanding about which pathways extraction method in Section 7.1 is better, we propose a way to validate these extractions which is called *cross validation*.

We follow a basic idea that if a pathway (connecting a given source target pair under a certain scenario) is still ranking high in other source-target pairs or under other scenarios, it's a "good" pathway extraction.

The cross ranking of a pathway $v_1, v_2, v_3, ..., v_n$ in another source target pair(v_i, v_o) is computed as follows: compared to $v_1, v_2, v_3, ..., v_n$, we first find the most similar pathway $\mathcal{P}_k$ among all pathways $\{\mathcal{P}\}$ from v_i to v_o, the cross ranking is thus the ranking of $\mathcal{P}_k$ among $\{\mathcal{P}\}$. When measuring the similarities, we use editing distance as a reference.

We develop space cross validation to assess that extractions are globally valid in the regulations of different targeted nodes, and develop time cross validation to assess that extractions are globally effective starting from different initial states.

7.2.1 Space Cross Validation

From system input(*TCR_HIGH*, *CD28*, *IL2_EX*) to system output(*FOXP3*, *IL2*), we can extract six important pathways group(a set of most important pathways). A ranking matrix R_{ij} is created to denote the space cross validation of T cell model. The rows and columns correspond six source-target pairs (from *TCR_HIGH* to *FOXP3*, from *CD28* to *FOXP3*, from *IL2_EX* to *FOXP3*, from *TCR_HIGH* to *IL2*, from *CD28* to *IL2*, from *IL2_EX* to *IL2* respectively). The element R_{ij} denotes the average cross ranking percentage of important pathways group extracted from ith source-target pair in the jth source-target pair. The smaller these cross ranking percentages are, the more valid the pathways extraction is.

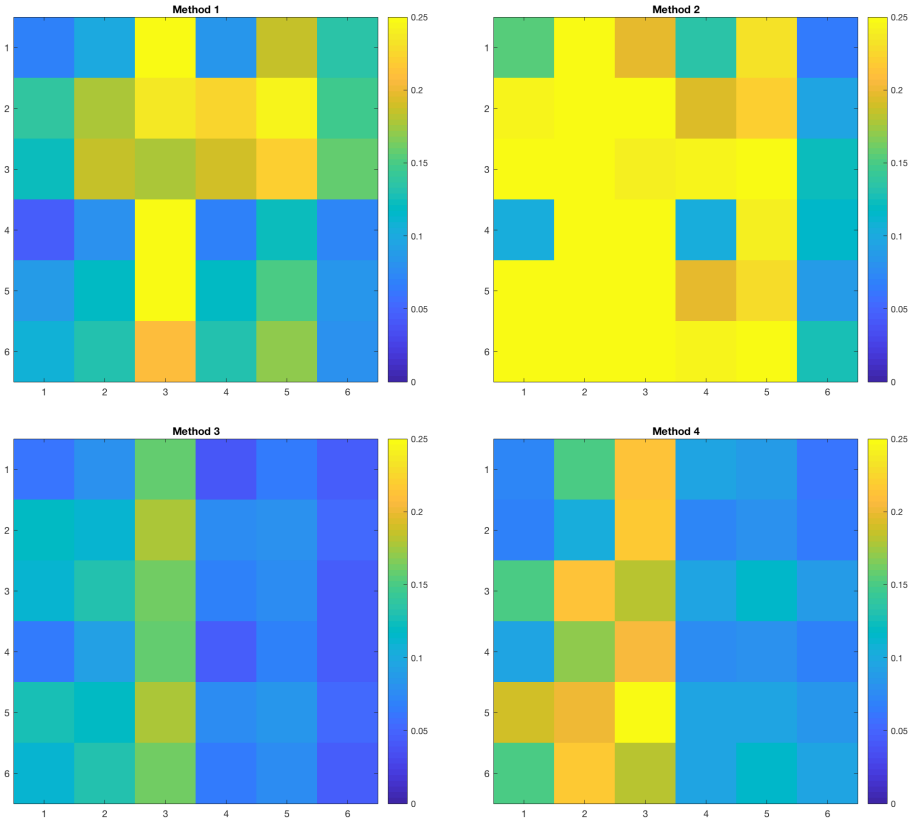


Figure 13: Space cross ranking percentage of six system source-target pairs following four proposed methods under static analysis

Figure 13 shows the space cross ranking percentage of six system source-target pairs following four proposed methods under the static analysis.

As shown in Figure 13, weighted normalized method(i.e. Method 3 and 4) perform better than others. Under the static scenario where all network states are assumed to be equally distributed, it's more reasonable to extract pathways biased towards these with higher activity(influence) near the targeted node. Comparing Method 3 with Equation (7.3) and Method 4 with Equation (7.4), we can also predict that these delay variables are necessary, partly due to the fact that delay variables are introduced to make the system behave in a scheduled manner and become closer to the biology observation.

As shown in Table 3, there is only one active pathways from *IL-2 EX* to *IL-2*(and to *FOXP3*) under high-dose and low-dose scenarios. When generating the space cross ranking of system source-target pairs, we just omit these two pairs and obtain Figure 14 and Figure 15, the rows and columns correspond four source-target pairs (from *TCR_HIGH* to *FOXP3*, from *CD28* to *FOXP3*, from *TCR_HIGH* to *IL2*, from *CD28* to *IL2*, respectively).

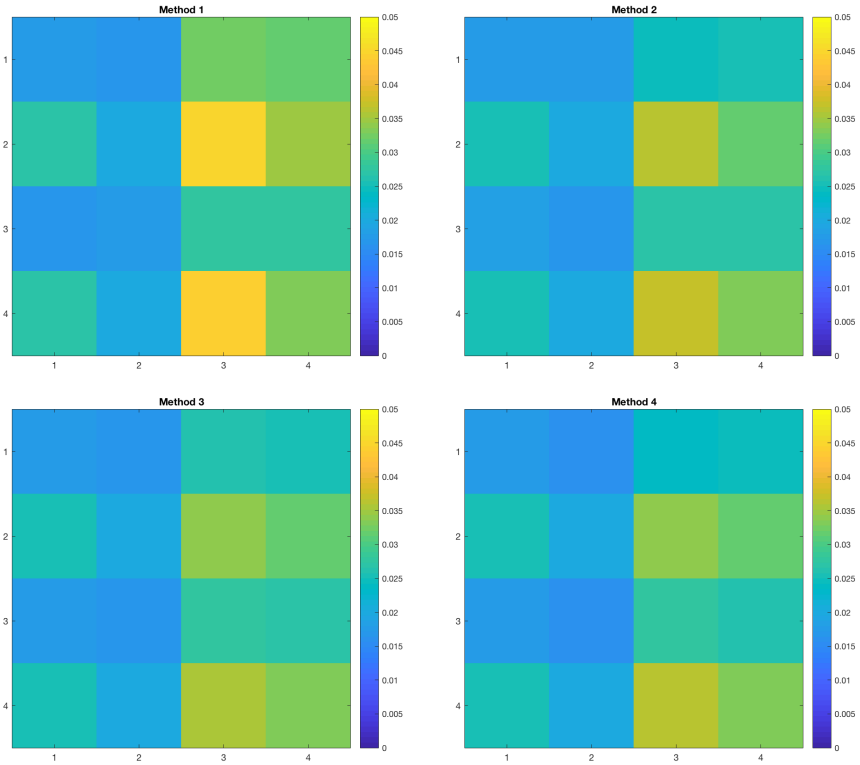


Figure 14: Space cross ranking percentage of four system source-target pairs following four proposed methods under high-dose scenario

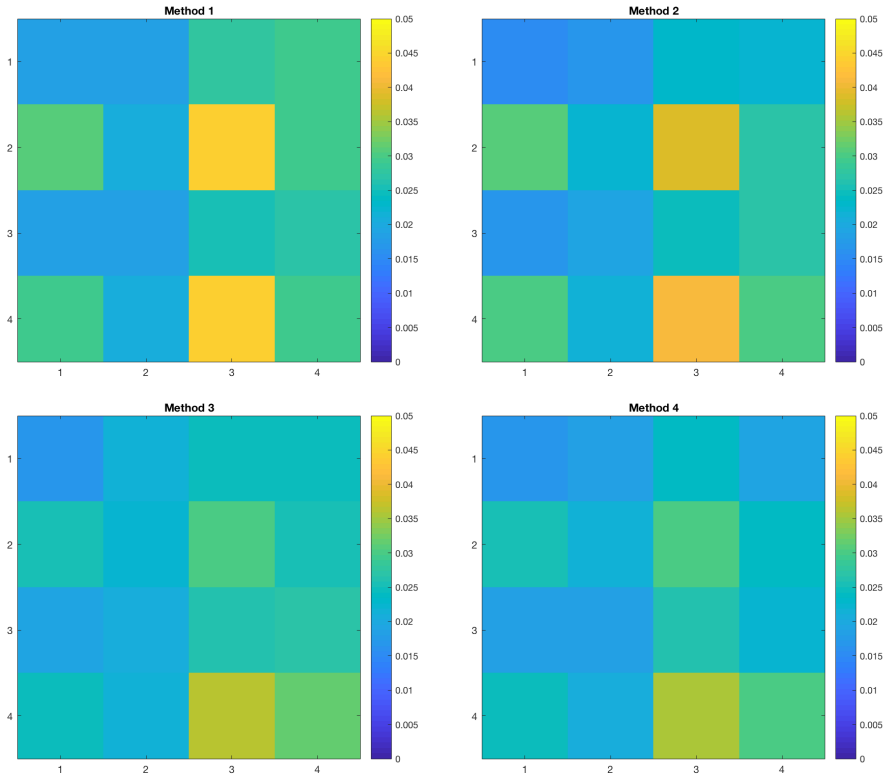


Figure 15: Space cross ranking percentage of four system source-target pairs following four proposed methods under low-dose scenario

As shown in Figure 14 and Figure 15, weighted normalized method(i.e. Method 3 and 4) always return a ranking percentage that is smaller than of other methods. This is consistent with our findings under static analysis in Figure 13.

It's worthwhile noting that all entries in matrices under dynamic scenarios(both high-dose and low-dose) are much smaller than entries in matrices under static analysis. The reason is that under dynamic scenarios, some activities(influence) are turned off to zero so some pathways become inactive and the regulation via these interactions is turned off. Thus, the total number of active pathways decreases and pathways extraction is biased towards these highly active pathways.

7.2.2 Time Cross Validation

In this section, we follow the same idea to verify whether one pathway extraction under a certain scenario is still effective under other scenarios. Similarly, we use the cross ranking percentage to measure the validness. As shown in Fig.6, we give the scenario cross validation under static, high-dose, low-dose and toggle scenarios of extractions following four proposed methods. For convenience, we choose to only plot the pathways extractions from *TCR_HIGH* to *IL2*.

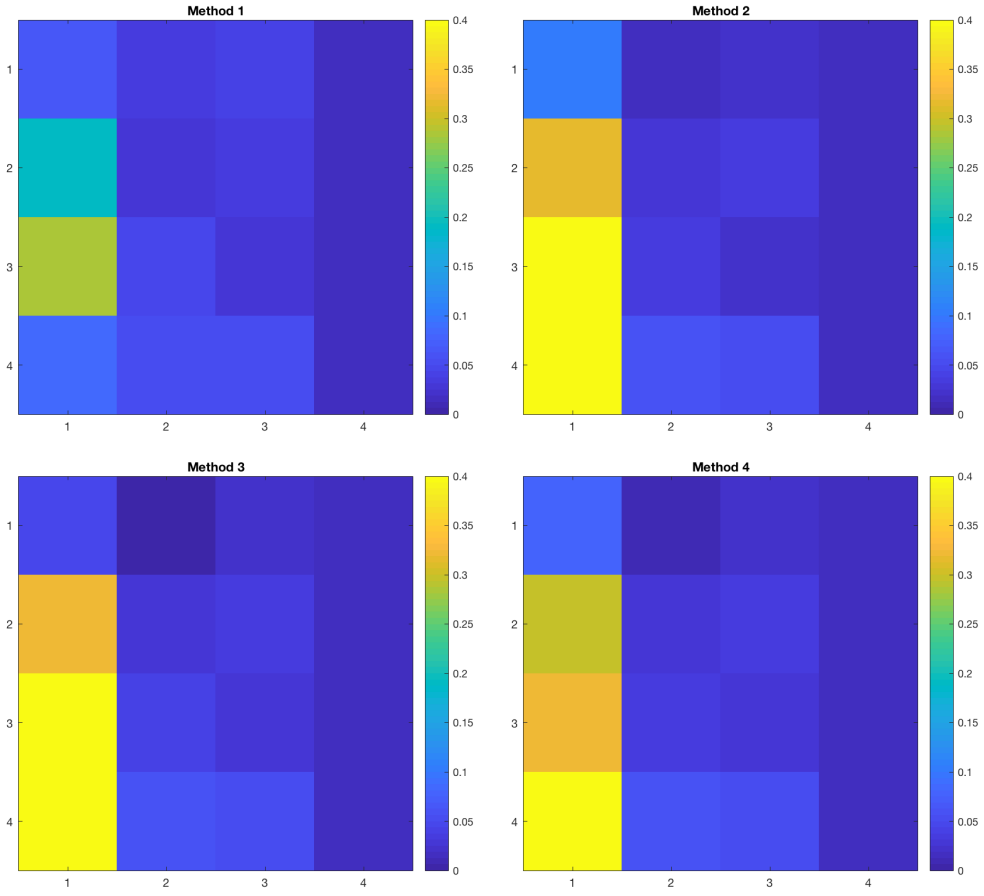


Figure 16: Time cross ranking percentage of extractions under different scenarios following four proposed methods

As shown in Figure 16, accumulative method performs best among these four methods, showing a high relatedness across four scenarios. That's to say, accumulative method is pretty good at extracting pathways which are required to be globally effective (no matter which state distributions the network follows). This also inspires us to apply accumulative extraction method to these application occasions where the network structure matters rather than the specific network states.

8.0 FUTURE WORK

8.1 POSSIBLE DEVELOPMENT

Almost all theoretical analysis of a biological model is aimed to learn the behavior of the model under perturbations and lead the model to certain states via least number of interventions. Within the sensitivity analysis framework, we can address these problems from the perspective of probability as follows. Given a perturbation on a certain element x_i, we study the dynamics of influence matrix and keep track of all elements which could be affected. The affection will either die out within several time steps(shown as the influence smaller than a threshold, pathway score less than a limit) or propagate to affect the long run behavior(shown as some obvious differences between steady state distribution vector). This is consistent with our previous finding in Section 2.3 that the steady state distribution vector under discrete modeling approach is highly dependent on initial states. After observing system behaviors under independent perturbations trial on each element, we shall assign each element a probability-based parameter vector(taking into account many factors such as element sensitivity, local influence, self-influence). With these parameters, we reduce the state intervention problem to a general ML(maximize likelihood) or MAP(maximize a posterior) problem.

As we have also mentioned in Section 2.3, the discrete modeling approach shows its advantage over PBN in terms of steady state distribution computation complexity. However, the complexity is still increasing exponentially as the model size goes up. An recent solution is studying the ergodicity of the underlying Markov chains and converting the distribution over space to time average. Recall in Section 2.2, we have defined scenario where perturbations could occur at a particular model element, at a specified time point. In other words, any

model element has some probability to flip its value at any time. This ensures that there is always a positive probability to pass from any state to any other state in one step, which is exactly the definition of ergodicity. Now the cost of computing steady state distribution is converted to the the cost of running time for which we should run the model simulation to obtain sufficient trajectories for time average. We have shown in [12], with discrete modeling approach, the running time before reaching an attractor is relatively smaller than other models. Together with the sensitivity analysis results and the framework of perturbations and interventions, we can even choose on purpose the initial states to shorten the simulation time.

8.2 CURRENT BOTTLENECK

The current bottleneck of our sensitivity framework lies in its generality to extend to any biological model. We have to address the following problems:

(1) The high computation complexity in influence calculation limits the size of models we can analyze, even with the help of ROBDD-based method shown in Section 4.2.

(2) The redundant searching in BFS when extracting all the pathways from a certain source and target element costs a lot. This will be more troublesome when the model size goes up. We'd like to adopt alternative algorithms or customize BFS to our requirements of both completeness and low complexity.

(3) The fact that dynamic sensitivity analysis shows great power in reducing the number of pathways is sometimes annoying since it possibly hide signaling pathways which are now dormant but will become active under other cases.

9.0 CONCLUSION

Understanding sensitivity is an important step to study system robustness against perturbations and adaptability to the environment. In this work, we propose a framework to study sensitivity via discrete modeling approach. Within the framework, we define element activity and sensitivity with respect to the state distribution of the modeled system. We perform both static and dynamic sensitivity analysis, the former assuming uniform state distribution, and the latter using a distribution estimated from stochastic simulation trajectories under a particular scenario. In addition, we also propose a Binary-Decision-Trees-based method to compute element influences. Within our sensitivity analysis framework, we add weights to interaction rules helps to identify key elements in the model, as well as dominant signaling pathways that determine the behavior of the overall model.

To the best of our knowledge, previous sensitivity analysis research did not focus on detecting crucial pathways (elements regulations sequence) in a complicated Boolean network. For a well-studied or informative regulatory model with a large number of nodes and complicated interactions, biologists are interested in extracting important pathways which can dominate the control on a targeted node. In this work, we also discuss how these pathways can be extracted with the help of sensitivity analysis. We then refine our pathways extraction by improving the sensitivity scores propagation algorithm. This ensures the balance between long regulation pathways and short ones and gives more flexibility to these algorithms for different application occasions. In order to evaluate the extracted pathways, we also develop cross validation to assess that extractions are "globally valid" in the regulations of different targeted nodes (validation in space) and are "globally effective" starting from different initial states (validation in time).

BIBLIOGRAPHY

[1] I. Shmulevich and S. A. Kauffman, "Activities and Sensitivities in Boolean Network Models," Physical Review Letters, vol. 93, no. 4, 2004.

[2] I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang, "Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks," Bioinformatics, vol. 18, no. 2, pp. 261-274, Jan. 2002.

[3] J. R. Faeder, M. L. Blinov, and W. S. Hlavacek, "Rule-Based Modeling of Biochemical Systems with BioNetGen," Methods in Molecular Biology Systems Biology, pp. 113-167, 2009.

[4] I. Shmulevich, E. Dougherty, and W. Zhang, "From Boolean to probabilistic Boolean networks as models of genetic regulatory networks," Proceedings of the IEEE, vol. 90, no. 11, pp. 1778-1792, 2002.

[5] K. Sayed, Y.-H. Kuo, A. Kulkarni, and N. Miskov-Zivanov, "DiSH simulator: Capturing dynamics of cellular signaling with heterogeneous knowledge," 2017 Winter Simulation Conference (WSC), 2017.

[6] I. Shmulevich, E. R. Dougherty, and W. Zhang, "Gene perturbation and intervention in probabilistic Boolean networks," Bioinformatics, vol. 18, no. 10, pp. 1319-1331, Jan. 2002.

[7] S. Kauffman, C. Peterson, B. Samuelsson, and C. Troein, "Genetic networks with canalyzing Boolean rules are always stable," Proceedings of the National Academy of Sciences, vol. 101, no. 49, pp. 17102-17107, 2004.

[8] "Analysis of random Boolean networks using the average sensitivity: Steffen Schober: Free Download and Streaming," Internet Archive, 02-Apr-2007. (Online). Available: https://archive.org/details/arxiv-0704.0197. (Accessed: 18-Mar-2018).

[9] W. Liu, H. Lahdesmaki, E. Dougherty, and I. Shmulevich, "Inference of Boolean Networks Using Sensitivity Regularization". EURASIP Journal on Bioinformatics and Systems Biology, pp. 1-12, 2008.

[10] X. Qian and E. R. Dougherty, "On the long-run sensitivity of probabilistic Boolean networks," Journal of Theoretical Biology, vol. 257, no. 4, pp. 560-577, 2009.

[11] A. Garg, K. Mohanram, A. D. Cara, G. D. Micheli, and I. Xenarios, "Modeling stochasticity and robustness in gene regulatory networks," Bioinformatics, vol. 25, no. 12, pp. i101-i109, 2009.

[12] N. Miskov-Zivanov, M. S. Turner, L. P. Kane, P. A. Morel, and J. R. Faeder, "The Duration of T Cell Stimulation Is a Critical Determinant of Cell Fate and Plasticity," Science Signaling, vol. 6, no. 300, May 2013.

[13] K. Sayed, C. A. Telmer, A. A. Butchy, and N. Miskov-Zivanov, "Recipes for Translating Big Data Machine Reading to Executable Cellular Signaling Models," Lecture Notes in Computer Science Machine Learning, Optimization, and Big Data, pp. 1-15, 2017.

[14] E. R. Dougherty, S. Kim, and Y. Chen, "Coefficient of determination in nonlinear signal processing," Signal Processing, vol. 80, no. 10, pp. 2219-2235, 2000.

[15] F. Somenzi, "CUDD: CU Decision Diagram package-release 3.0.0," University of Colorado at Boulder, 2015.

[16] J. Kahn, G. Kalai, and N. Linial, "The influence of variables on Boolean functions," [Proceedings 1988] 29th Annual Symposium on Foundations of Computer Science, 1988.

[17] S. Minato, N. Ishiura, and S. Yajima, "Shared binary decision diagram with attributed edges for efficient Boolean function manipulation," 27th ACM/IEEE Design Automation Conference.

[18] B. Bollig and I. Wegener, "Improving the variable ordering of OBDDs is NP-complete," IEEE Transactions on Computers, vol. 45, no. 9, pp. 993-1002, 1996.

[19] S. Sakaguchi, "Regulatory T Cells: History and Perspective," Regulatory T Cells Methods in Molecular Biology, pp. 3-17, 2011.