Vol. 00 no. 00 2009
Pages 1–11

Finding All Steady States in Biological Regulatory Networks

Ferhat Ay, Fei Xu, and Tamer Kahveci

Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611
{fay, feixu, tamer}@cise.ufl.edu

ABSTRACT

Motivation. Computing the long term behavior of regulatory and signaling networks is critical in understanding how biological functions take place in organisms. Steady states of these networks determine the activity levels of individual entities in the long run. Identifying all the steady states of these networks is difficult as it suffers from the state space explosion problem.

Results. In this paper, we propose a method for identifying all the steady states of regulatory and signaling networks accurately and efficiently. We build a mathematical model that allows pruning a large portion of the state space quickly without causing any false dismissals. For the remaining state space, which is typically very small compared to the whole state space, we develop a randomized algorithm that extracts the steady states. This algorithm estimates the number of steady states, and the expected behaviors of individual genes and gene pairs in steady states in an online fashion. Also, we formulate a stopping criteria that terminates the randomized algorithm as soon as user supplied percentage of the results are returned with high confidence. Finally, in order to maintain the scalability of our algorithm to very large networks, we develop a partitioning-based estimation strategy. We show that our algorithm can identify all the steady states accurately. Furthermore, our experiments demonstrate that our method is scalable to virtually any large real biological network.

Availability. All the code developed for this paper is available at http://bioinformatics.cise.ufl.edu/palSteady.html

1 INTRODUCTION

Analyzing biological networks is essential in understanding the machinery of living organisms which has been a main goal for scientists. We use the term *biological regulatory networks (BRN)* to combine gene regulatory networks and signal transduction pathways. To capture the biological meaning of BRNs, it is necessary to characterize their long term behavior. A common way to achieve this is to identify the *steady states* of the dynamic system defined by BRNs. A *state* shows the activity levels of genes at a certain time point. The state of a BRN can change over the time due to internal regulations or external stimulants. We say that a state of a BRN is steady if that state will periodically be revisited forever once the BRN reaches to that state (see Figures 2 (a) and (b)). In other words, the steady states represent the possible long term outcomes of the corresponding BRN. Identification of steady states is a crucial problem and is useful in many applications such as the treatment of various human cancers [12] (e.g. leukemia, glioblastoma) and genetic engineering [21]. *In this paper, we consider the problem of identification of all steady states in BRNs.*

The naive approach to this problem is to exhaustively search the state space for the steady states. However, the number of possible states of a BRN is exponential in the number of its genes. For a BRN with n genes, the state space size is 2^n for the boolean state model. Therefore, exhaustive methods are computationally infeasible for even moderate sized BRNs. To address this problem, some existing methods use finite-state Markov chains [10], binary decision diagrams (BDD) [9], constraint programming [6], probabilistic boolean networks [24] and relational programming [22]. Orthogonal to the selection of the above method, there are two options for modeling the state transitions. These are *synchronous* and *asynchronous* models and both are used in the literature. Synchronous model assumes that the activity levels of all the genes change simultaneously. Hence, the next state is deterministically decided by the current state. On the other hand, asynchronous model assumes that the time is divided into small enough intervals, such that only one gene can change its state at a time and state change is equally likely for all genes. The advantages/disadvantages of these models together with their effect on running time of steady state identification algorithms are discussed in several papers in the literature [1, 2, 8]. Here, we use the asynchronous model in our algorithm discussion since it allows the identification of an important type of steady states, namely cyclic steady states. However, it is important to point out that our method is independent of the choice of state transition model.

Garg *et al.* [9] proposed one of the most recent methods to identify the steady states of BRNs. They use the boolean network model for BRNs. They represent the state space using the BDD data structure. Their algorithm, however, fails to classify a significant percentage of the states correctly. Here, we present a brief example on a real sub-network given in Figure 1 to illustrate the problems of this method of Garg *et al.* (Detailed evaluation of this method is in Section 4.2.) For this 6 gene sub-network of human p53 signaling pathway, Garg *et al.*'s method finds four steady states. The sets of active genes in these steady states

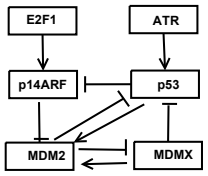


Fig. 1: A portion of p53 signaling pathway. p53 pathway plays an important role in the origination of different cancer types. Pointed arrow heads represent positive regulation (activation) and arrow heads with bars represent negative regulation (inhibition).

are: {MDM2, MDMX}, {MDM2}, {MDMX} and ∅. In order to evaluate the accuracy of their results, we surveyed the literature on p53 network and extracted its steady states. We observed that the first three states reported above by their algorithm are not steady (i.e. they are false negatives). Also, their method fails to find the steady states corresponding to oscillation(s) between p53 and MDM2 levels (false negatives) that are observed in the literature [4, 14]. Finally, the running time of their method grows quickly as the complexity or the size of the underlying BRN increases.

Our Contributions: In this paper, we develop an algorithm that identifies all the steady states of BRNs accurately and efficiently. We use the boolean network model proposed by Kauffman et al [13] due to its parallelism with the level of currently available data and its successful applications on real data sets [7, 16]. We build a hypothetical state transition graph using the interactions in a BRN. We develop a mathematical model that classifies each state into one of the three classes, namely Type 0, Type 1 and Type 2. Type 0 and Type 2 states are guaranteed to be steady and transient (i.e. not steady), respectively. Type 1 states can be either one. We use BDDs to partition the states into these classes efficiently. To further classify the Type 1 states, we develop a randomized algorithm. While sampling, we calculate the estimators for the number of steady states, expected steady state distribution of individual genes and joint-steady state distributions of gene pairs. We calculate a stopping criteria from the statistical information of explored states. This criteria allows early termination of sampling when the user defined percentage of steady states are found with high confidence. To make our algorithm scalable to large BRNs, we incorporate a partitioning strategy. Our partitioning strategy works for both disconnected and weakly connected sub-networks without losing steady state information. In summary, our technical contributions are:

• We build a mathematical model for pruning a very large portion of state space quickly without any information loss.

• We develop a randomized algorithm that computes estimators for the number of steady states and the fraction of individual genes and gene pairs being active in these states in an online fashion. Our algorithm guarantees to find all the steady states after sufficient number of iterations.

• We develop a partitioning-based strategy to utilize the weak connectivity of BRNs.

The organization of the rest of this paper is as follows: Section 2 presents the background information. Section 3 discusses our methods. Section 4 presents the experimental results. Section 5 concludes the paper.

2 BACKGROUND

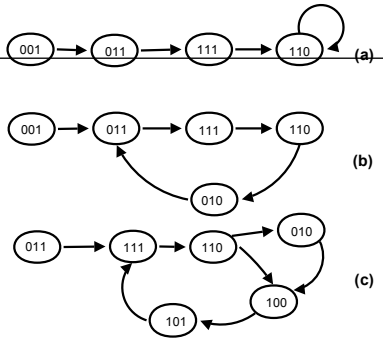
In order to find the long term behavior of BRNs, a dynamic model should be used to express the state changes. Here, we describe how the dynamic behavior of BRNs are modeled. First, we define a *state* and a *steady state* of a BRN. Then, we discuss the existing methods and models for identification of steady states in dynamic biological networks.

Assume that a BRN has n genes. A *state* of this BRN consists of activity levels of individual genes. We represent the state of the ith gene by $X_i = 1$ (active) or $X_i = 0$ (inhibited). The state of that BRN is denoted as $[X_1 X_2 \cdots X_n]$. A set of genes alters the state of another gene set if the first set has activators or the suppressors of the genes in the second set. Figure 2 shows how the state of a BRN changes for three hypothetical BRNs.

A *steady state* in the state transition graph is a state in which the system stabilizes. This stability is achieved at a state either if none of the genes issue a change in their activity levels or if there is only one possible chain of state transitions from the current state which deterministically leads to the same state after a certain number of state transitions. In other words, once the state transitions lead to a steady state the probability of revisiting this state through a fixed sequence of states is 1. Thus, the steady states reflect the long term behavior of the biological system. We call all the remaining states *transient*. For example, consider the states of the BRN in Figure 2(a). Here, the state $[110]$ is steady as the BRN remains in that state forever once it reaches to that state. The BRN shown in Figure 2(b) has four steady states, namely $[011]$, $[111]$, $[110]$ and $[010]$. This is because the BRN deterministically visits these states in cycle forever once it reaches to one of them. We call these type of steady states "*cyclic steady states*". On the other hand, all the states in Figure 2(c) are transient. The reason is that once the state $[110]$ is reached, it is impossible to know which path will be taken before revisiting $[110]$.

Existing work on steady state identification: The trivial way to identify all the steady states is to analyze the whole state transition graph exhaustively. However, the size of this graph grows exponentially with the number of genes in the BRN. Therefore, enumeration of all the states is not practical and efficient methods are necessary.

A number of methods have been developed to find the steady states of BRNs. Devloo *et al.* [6] used generalized logical analysis [29] together with constraint programming for this purpose. They introduced an image function to traverse the state transition graph by utilizing the logical parameters described in Snoussi *et al.*[26]. However, their algorithm does not work for networks containing multiple arcs of same weight from the same origin. Since networks with multiple arcs of same weight from the same origin are common [17], this limitation restricts the applicability of



their algorithm. Using the *Probabilistic Boolean Network Model*, Shmulevich *et al.*[25] calculated the steady state probabilities of genes via random gene perturbations. However, they don't distinguish between different interaction types such as activations and inhibitions. Ignoring the type of interactions in the modeling phase degrades the accuracy of their algorithm from the very beginning. Shlomi *et al.*[23] used linear programming to extract the steady states after gene perturbations. However, their method cannot find the cyclic steady states in BRNs. Recently, Schaub *et al.*[22] proposed an extension of boolean network model called *Qualitative Networks* and analyzed signaling networks by symbolic methods. They identified the infinitely visited states of multicellular systems by using relational computation tools. However, their idea of using partitioned transition relations to speed up the algorithm is not valid for single cell model.

To the best of our knowledge, the most recent algorithm for identification of all steady states of biological regulatory networks is proposed by Garg *et al.*[9]. They used boolean network model to represent the regulatory networks and BDD data structure for the representation of its state space. This method, however, incurs both false positives and false negatives (see the example in Section 1). Furthermore, its running time increases quickly as the complexity and the size of the BRN increases. We evaluate this algorithm in detail in Section 4.2.

3 METHODS

This section discusses our algorithm for identifying all the steady states of BRNs. Section 3.1 describes the mathematical model for expressing the states of a BRN. Section 3.2 discusses the strategy we use to split the search space into three subspaces. Section 3.3 presents our randomized algorithm for extracting steady states from the third subset above. Section 3.5 discusses our partitioning strategy to accelerate our algorithm.

3.1 State Transition Model

In order to identify the steady states of a BRN, we first need to build a mathematical model that explains its states and how the network moves from one state to another.

Let $X_i(t) = true/false$ denote the state of the ith gene at time t. Here *true* denotes that ith gene is "active" and *false* denotes that it is "inactive". We use X_i instead of $X_i(t)$ for simplicity wherever appropriate.

We summarize the interactions that determine the next state of the ith gene from the activity values at time t as follows. The ith gene will be inhibited if at least one of its suppressors is active. If all the suppressors of the ith gene are inactive and at least one of its activators is active, then it becomes active in the next time step. The assumption that inhibitors are stronger is due to the binding mechanism of inhibitors. In all other situations the state of the ith gene remains unchanged. The following equation summarizes this process:

$$X_i(t+1) : (X_i(t) \vee p_A(t)) \wedge \neg p_S(t) \quad (1)$$

In this equation, the symbols $\vee$ and $\wedge$ denote the logical "AND" and "OR" operators, $p_A(t)$ and $p_S(t)$ represent predicates for the activators and the suppressors of the ith gene at time t, respectively. We compute these predicates as $p_A(t) = \bigvee_{j \in A} X_j(t)$ and $p_S(t) = \bigvee_{j \in S} X_j(t)$, where A and S are the sets of indices for activators and the suppressors of the ith gene.

EXAMPLE 1. *Consider the BRN in Figure 1. Assume that E2F1 and ATR are the only active genes at some point in time. E2F1 will activate p14ARF and ATR will activate p53. When p53 becomes active, it will suppress p14ARF even when E2F1 remains active. p53 will also activate MDM2, whose activity will then suppress p53.* □

It is worth mentioning that our model uses boolean values for the states of the genes since it is successfully used in the literature for BRNs [7, 16]. The method we develop in this paper is independent of this choice. It can be used for categorical state model, where the state of a gene can have more than two activity levels (e.g. low, medium, high activity) and more complex predicates for interactions.

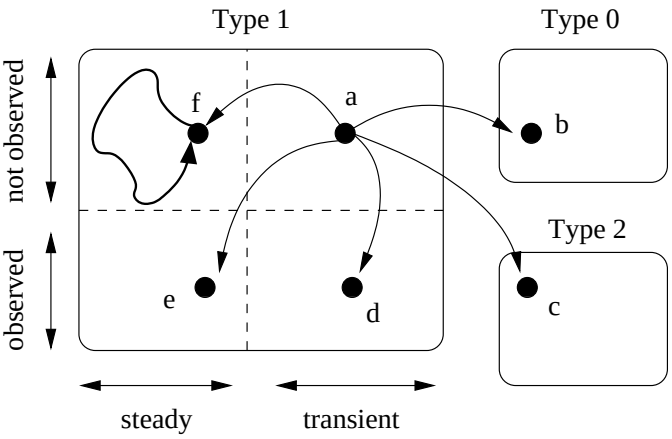


Fig. 3: Summary of the traversal process for a randomly picked state from unobserved Type 1 states. The unobserved state is either a steady state as shown with f or a transient state as shown with a. In the latter case, the traversal may take different paths before labeling a as transient. In both cases, all the states on traversed paths are also labeled as steady or transient.

An important observation is that, even though the next state of the ith gene is deterministically calculated, there can be multiple next states for the whole network since we use asynchronous model. A state of a given BRN is defined by the states of individual genes. Let $u = [X_1 \cdots X_n]$ denote a state of the network. The network can move from state u to state $v = [X_1 \cdots X_{i-1} \neg X_i X_{i+1} \cdots X_n]$ only if the ith gene is one of the genes that can have a state change. Individual genes that can issue a state change at a given state determines the possible next states of the network.

We model the changes in the states of a BRN using an abstract graph representation. In this graph, each vertex corresponds to a possible state of the BRN. Thus, if there are n genes in a BRN,

then the corresponding graph contains 2^n vertices. There is an edge from vertex u to vertex v, if it is possible to change the state of the BRN from the state represented by u to the state represented by v by only changing the state of a single gene. There can be up to $n2^n$ edges between these states. This graph is hypothetical as we use it only for building our mathematical model. We never materialize this exponential sized graph in our method.

We classify the vertices of this graph into three classes based on the number of their outgoing edges. Figure 2 provides visual examples for all three state types listed below:

- *Type 0:* The vertices that do not have any outgoing edges (except self cycles). These vertices correspond to steady states as the state of the network cannot change once one of them is visited. (Figure 2(a))
- *Type 2:* The vertices that have two or more outgoing edges. These vertices represent transient states. (Figure 2(c))
- *Type 1:* The vertices that have exactly one outgoing edge. These are the tricky vertices. The states for these vertices can be steady or transient. (Figure 2(b) and 2(c))

In the following section, we describe the method we use for the segregation of states into the above three types.

3.2 Segregation of States using BDDs

As we discussed in the previous section, we never generate the state transition graph of the input network. A simple observation on our state transition model allows us to segregate the states without this materialization. This segregation results in not only the immediate identification of all non-cyclic steady states, but also eliminates a huge portion of states by classifying them as transient. For instance, for *the pyk2 network* with 24 genes and $16,777,216$ (2^{24}) possible states, our segregation method classifies $114,300$ ($\approx 2^{17}$) states as Type 0 (i.e. steady) and $16,086,712 (\approx 2^{24})$ states as Type 2 (i.e. transient) in only 0.08 seconds. Remaining 576,204 states are labeled as Type 1. Thus, we need to explore only a small fraction of the state space (which still can be large for practical purposes).

Here, we describe how we construct the BDDs for all Type 0 states and all Type 1 states, namely Z_0 and Z_1. We first define a predicate that will be handy in this discussion.

$$C_i : X_i(t+1) \oplus X_i(t) \tag{2}$$

Here, $\oplus$ denotes the logical "XOR" operator. C_i evaluating to true at time t means that gene i will change its state from X_i to $\neg X_i$ at time $t+1$. Otherwise, it preserves its current state. The following equations, show the formulas of BDDs representing Type 0 and Type 1 states:

$$Z_0 : \bigwedge_i \neg C_i \quad \text{and} \quad Z_1 : \bigvee_i (C_i \wedge (\bigwedge_{j \neq i} \neg C_j).$$

$Z_0 =$ "True" represents the states that do not satisfy any of the C_i conditions (i.e. none of the genes change state). The states in $Z_1 =$ "True" satisfy exactly one of the C_i conditions (i.e. exactly one gene changes state). The states which are not included in the two BDDs above are called Type 2 and they are all transient states since they have more than one possible successor states. The BDD for these states can be constructed similarly. However, we eliminate these states since they do not reflect the long term behavior of the system. By doing this without materialization, we quickly reduce the state space of the problem to a significantly smaller one. In the next section, we describe how we decide the steadiness of Type 1 states.

3.3 Extracting Cyclic Steady States

We have shown that we can identify steady states corresponding to Type 0 vertices and transient states corresponding to Type 2 vertices efficiently. In this section, we develop a randomized algorithm that identifies the steady states of Type 1. We call these states "cyclic steady". An example for this is the cycle of four states in Figure 2(b).

Our randomized algorithm traverses the Type 1 vertices and reports the ones corresponding to steady states. After traversing a portion of the vertices, we estimate the total number of steady states, the probability of each gene being active and the joint probability of gene pairs being co-expressed in steady states. This is desirable because the user can decide to stop the traversal when sufficient number of steady states are discovered with high confidence (e.g. when 90% of the steady states are reported with 95% confidence). This kind of online querying is useful especially when the underlying BRN is very large and the space of Type 1 states is too big to completely traverse.

Algorithm 1 briefly describes how we traverse the Type 1 states. Next, we elaborate on different steps of this algorithm.

Algorithm 1 RANDOMIZED TRAVERSAL OF TYPE 1 STATES

1. Randomly get an unobserved vertex from the Type 1 set.
2. Follow the out going edge to traverse the graph until seeing one of the following vertices
   (i) A vertex that is labeled as transient or steady in previous iterations.
   (ii) A vertex that is traversed in this iteration.
3. Label all the traversed vertices as transient or steady and update the estimators.
4. Stop if number of steady states observed so far is *sufficient*.

Step 1. We obtain a random seed state among the untraversed satisfying assignments of the BDD for Type 1 states. We do this by traversing the BDD from root node to the leaf level. At each step of the traversal, we randomly pick a child node of the currently visited node. The probability of moving to a child is equal to the number of states under the subtree of that child divided by the number of states under the subtree of the parent node. Each level of the BDD decides the state of one gene. Thus, by randomly branching to a child node we choose the state of a gene randomly from the satisfying assignments of the BDD. When we reach the leaf level of the BDD,

the states of all the genes are determined and hence, our seed state for the whole BRN.

Step 2. Once we choose an unobserved Type 1 state, the next step is to understand whether or not we can reach to a new steady state from this state. To do this, we traverse the state transition graph starting from this vertex by following the edges.

Since the chosen state is of Type 1, by definition, it has only one outgoing edge. Thus, we can easily find the next state as the state that satisfies the transition condition. We continue traversal by applying the same principle. Figure 1 of the Supplementary Materials summarizes the possible cases that can occur during this traversal. Totally there are four different cases. Starting from an unobserved state if we traverse one of the following three paths then all the states visited on this path are transient:

- A path ending in a Type 0 state
- A path ending in a Type 2 state
- A path ending in a state that is observed in previous iterations

Notice that all three cases correspond to Step 2(i) of our traversal method. The next case produces both cyclic steady and transient states.

- A path leading to a cycle of states visited in current observation

In this case, we label all the states on the cycle as steady and the other states on the path as transient.

Step 3. At each iteration of the while loop, we traverse a path in the state transition graph and label each state on this path as transient or steady. We name the set of vertices visited in each such traversal as an observation. Using these observations, we develop estimators for the total number and the "*profile*" of steady states. The profile of the steady state is the vector where the ith entry is the expected fraction of the steady states at which the ith gene is active. For example, if the second entry of the profile is 0.95, it means that we expect that the second gene is active in 95% of the steady states. We also compute the estimators for the joint expression (co-expression) fractions of gene pairs. Computing these estimates is important as they can lead to early prediction of the steady state profile and early termination of the algorithm.

Here, we describe in detail the calculation and the analysis of the estimator for the total number of cyclic steady states. First of all, we prove that it is an unbiased estimator. Then, we discuss how to minimize the variance of this estimator. For the other estimators we only give the formulations.

First, let us introduce some notation we use throughout this section:

- N_0, N_1: Number of Type 0 and Type 1 states, respectively.
- $O_i = (s_i, t_i)$: ith observation. s_i and t_i are the number of observed steady and transient states traversed in this observation.
- S_i, T_i, U_i: Total number of observed steady states, observed transient states and unobserved states after first i observations, respectively.

From the definitions above, we can calculate $U_i = N_1 - S_i - T_i$, $S_i = \sum_{j=1}^{i} s_j$ and $T_i = \sum_{j=1}^{i} t_j$. Now, we introduce a 0/1 random variable B_i for each observation O_i. We simulate our sampling by assuming at any time one and only one of the B_i's can be 1. In other words, $E[B_i B_j] = 0$ for any $i \neq j$. Notice that $E[B_i] = E[B_i^n] = \frac{s_i+t_i}{N_1}$ for observation O_i. We formulate the estimator of the total number of Type 1 steady states at the ith iteration as:

$$F_i = \sum_{k=0}^{i} B_k s_k \frac{N_1}{s_k + t_k} \tag{3}$$

LEMMA 1. *The estimator F_i is an unbiased estimator.*

Proof: We prove this by showing the expected value of F_i is equal to the total number of Type 1 steady states. Taking expectations of both sides and replacing $E[B_k]$ with $\frac{s_k+t_k}{N_1}$:

$$E[F_i] = E[\sum_{k=0}^{i} B_k s_k \frac{N_1}{s_k + t_k}] = \sum_{k=0}^{i} E[B_k s_k \frac{N_1}{s_k + t_k}] = \sum_{k=0}^{i} s_k$$

□

After defining the estimator, the next step is to calculate its variance.

LEMMA 2. *The variance of F_i is*

$$Var[F_i] = \sum_{j=0}^{i} s_j^2 (\frac{N_1}{s_j + t_j}) - [\sum_{j=0}^{i} s_j]^2.$$

Proof: We know that, $Var[F_i] = E[F_i^2] - E^2[F_i]$. We first compute F_i^2.

$$F_i^2 = \sum_{j=0}^{i} B_j s_j \frac{N_1}{s_j + t_j} \sum_{k=0}^{i} B_k s_k \frac{N_1}{s_k + t_k}$$

$$= \sum_{j \neq k} B_j B_k s_j s_k (\frac{N_1}{s_j + t_j})(\frac{N_1}{s_k + t_k}) + \sum_{j=0}^{i} B_j^2 s_j^2 (\frac{N_1}{s_j + t_j})^2$$

When we take the expected value of F_i^2 the first term cancels since $E[B_j B_k] = 0$ for any $i \neq j$. Hence, the variance of F_i can be computed as:

$$Var[F_i] = E[F_i^2] - E^2[F_i] = E[\sum_{j=0}^{i} B_j^2 s_j^2 (\frac{N_1}{s_j + t_j})^2] - (E[F_i])^2$$

$$= \sum_{j=0}^{i} s_j^2 (\frac{N_1}{s_j + t_j}) - [\sum_{j=0}^{i} s_j]^2$$

□There are many ways to build an estimator from F_js. However, it is desirable to build an estimator with a small variance as it converges to true solution faster. Following Lemma builds the estimator with minimum variance.

LEMMA 3. *The estimator that has the smallest variance is*

$$T = \sum_j \frac{1}{\sum_i \frac{1}{V_i} V_j} F_j$$

Proof: Now, we discuss how we combine the estimators $F_1, F_2, ..., F_n$ with variances $V_1, V_2..., V_n$ to minimize the overall variance of our estimation. In other words, we want to find the weight parameters $\gamma_1, \gamma_2, ..., \gamma_n$ such that $\sum \gamma_i = 1$ and the variance of the estimator for total number of steady states of Type 1 is minimized. Let us denote this new estimator as $T = \sum \gamma_i F_i$. Then,

$$Var(T) = \sum \gamma_i^2 V_i$$

Mathematically, our aim is to minimize $\sum \gamma_i^2 V_i$ given $\sum \gamma_i = 1$. We formulate this problem by using Lagrange Multiplier as follows:

$$L = \sum \gamma_i^2 V_i - \lambda(\sum \gamma_i - 1)$$

Taking derivative of both sides with respect to each γ_i, we get the equations:

$$2\gamma_i V_i - \lambda = 0, \lambda = \frac{1}{\sum \frac{1}{2V_i}}$$

Solving these equations we get the γ_i values that minimizes the $Var(T)$ as:

$$\gamma_j = \frac{1}{\sum \frac{1}{V_i} V_j}$$

Thus, by using the value of γ_is we find that the estimator with smallest variance is

$$T = \sum_j \frac{1}{\sum \frac{1}{V_i} V_j} F_j$$

□

Next, we give the formulations of the estimators for the fractions of each gene and each gene pair being active in steady states. First, we formulate our estimator for the fraction of a gene being active in cyclic steady states. Assume that the number of steady states at the ith observation in which the kth gene is active is $n_{k,i}$. An estimator for the kth gene after the ith iteration is then :

$$G_{k,i} = \sum_{j=1}^{i} n_{k,j}/S_i \tag{4}$$

Let $n_{a\leftrightarrow b,i}$ denote the number of steady states in which gene a and gene b are both active or both inactive after the ith observation. We calculate the estimator of joint probability of two genes having the same activity level at a steady state as:

$$J_{a\leftrightarrow b,i} = \sum_{j=1}^{i} n_{a\leftrightarrow b,j}/S_i \tag{5}$$

The probability of a gene being active in steady states is important in characterizing the overall steady state behavior of this gene. We found that some genes can be active in more that 90% of the steady states, whereas some others show activation in only less than 10% of these states. The joint probability of two genes being active together is called co-expression. Co-expression of gene pairs are crucial in functional annotation of genes [24]. Therefore, the estimators we calculate above are useful in interpreting the co-expressed genes in BRNs. We discuss their biological significance in experimental results section.

Step 4. When our randomized algorithm finishes traversing all Type 1 states (steps 1 to 3), it finds all the steady states. However, in some applications it might be sufficient to find a predetermined percentage of steady states. We develop statistical criteria to be able to terminate the algorithm quickly after a sufficient portion of the Type 1 states are explored. Our method still guarantees that the desired percentage of the results are found with high confidence. More precisely, when the user supplies a parameter α (e.g. 0.9), we compute a confidence $c \in [0, 1]$, at each iteration such that *"at least $\alpha \times 100$ percent of the steady states are found with probability at least c"*. This is desirable as the user can terminate the loop when c is large enough for the underlying application.

Now, let us describe how the stopping criteria works. Let A^* denote the actual number of total Type 1 steady states. If we have known the value of A^* we could have stopped sampling with a confidence value of $c = 1$ when $A^* < S_i + \frac{(1-\alpha)(N_0+S_i)}{\alpha}$ is satisfied. That is the time when we are sure that $\alpha \times 100$ percent of the steady states are already reported. Since we do not know A^* in advance, we use the information gathered from observed portion of states. We compute A_i which denotes the minimum number of total steady states that needs to be present for our method to decide to continue traversal.

$$A_i = S_i + (1 - \alpha)(N_0 + S_i)/\alpha \tag{6}$$

Trivially, if $A_i > U_i + S_i$ we just stop sampling with 100% confidence as even if all the unobserved states were to be steady, the reported ones constitute at least $\alpha \times 100$ percent of total steady states. Otherwise, we calculate the confidence value in ith iteration as the probability that we would have observed at least S_i steady states in our observations so far if there were A_i unobserved steady states. Formally, we compute the confidence as:

$$C(A_i) = \sum_{k=S_i}^{S_i+T_i} [\binom{S_i + T_i}{k} q_i^k (1 - q_i)^{S_i+T_i-k}] \tag{7}$$

q_i in Equation 7 represents the percentage of steady states if there were A_i steady states in Type 1 states (i.e. $q_i = \frac{A_i}{N_1}$). The inner term of the summation represents "The probability of getting exactly k steady states from S_i+T_i currently observed states if the probability of a state being steady is q_i".

Lemma 4 shows that, the confidence value reported when we stop sampling is never an over estimation. In other words, the stopping criteria does not cause any false dismissals.

LEMMA 4. *The confidence value given in Equation 7 by using A_i does not lead to false dismissal.*

Proof: Here, we have three cases to consider:

- *Case1 :* $(A^* > A_i)$
  Then, $q_* = \frac{A^*}{N_1} > \frac{A_i}{N_1} = q_i$. Since the confidence value is calculated as the area under the right hand side of a binomial probability distribution function (i.e. inverse CDF), c value will be larger for a larger value of q. Hence, $C(A^*) > C(A_i)$. That means whenever we stop sampling the confidence we report is conservative.
- *Case2 :* $(A^* = A_i)$
  Trivially, $C(A^*) = C(A_i)$ when we terminate the sampling.
- *Case3 :* $(A^* < A_i)$
  This case implies that we overestimated the total number of steady states in confidence value calculation. Only thing that can happen in such a case is that our method continues sampling when it does not need to. Since the actual number of steady states are less than what we have guessed, when we decide to stop we have already sampled at least as many steady states as we needed.

□

Corollary 1 follows from Lemma 4.

COROLLARY 1. *Our algorithm guarantees to find all the steady states when the confidence value reaches* 1.0. □

3.4 Partitioning of independent sub-networks

An interesting case happens when the BRN can be partitioned into disconnected subnetworks. Finding the steady states in the entire BRN from the results of the individual parititions is relatively easy in this case. When there are no activations or inhibitions between two or more sub-networks of the input network, the steady states of a partition can be determined independent of the other partitions. In the case of BRNs, that type of partitioning is usually applicable [11]. Assume that we have k independent partitions for an input BRN. We know that Type 0 states have no outgoing edges and Type 1 steady states have only one outgoing edge which is a part of a cycle. Therefore, the combination of two Type 0 steady states from two partitions results in a Type 0 steady state for the union of these partitions. This is also true for multiple partitions. On the other hand, combining a Type 1 steady state of a partition with a Type 0 steady state of another partition we get a Type 1 steady state for the union of these partitions. However, combining two Type 1 steady states produces two outgoing edges from combined state. Hence, all such states are transient. Clearly, any other type of state combination results in transient states.

Let, $\Theta_1, \Theta_2, \cdots, \Theta_k$ and $\Omega_1, \Omega_2, \cdots, \Omega_k$ denote the number of Type 0 and Type 1 steady states in each partition. Using the above observations on combining the steady states, we formulate the total number of Type 0 and Type 1 steady states (Θ and Ω respectively) of the actual network as:

$$\Theta = \prod_{i=1}^{k} \Theta_i \quad \text{and} \quad \Omega = \sum_{i=1}^{k} [\Omega_i \prod_{i \neq j} \Theta_j].$$

3.5 Further Partitioning the BRN

The initial segregation of states and the use of stopping criteria for sampling accelerate our algorithm significantly. Since the state space grows exponentially, further speed up might be necessary for large networks. For this purpose, we propose a network partitioning-based strategy to increase the scalability of our method. Especially, when the number of Type 1 states and the number of cyclic steady states are large, partitioning the network before applying the above steps can improve the running time significantly. In here, we discuss how we combine the results gathered from these partitions to construct overall steady states of the actual network without loss of any information.

Combining the results from different partitions is rather easy when the partitions are independent of each other (i.e. when they are disconnected). We leave that case to the Supplementary Materials and discuss the harder case when the partitions weakly depend on each other (i.e. there are a small number of interactions between different partitions). We use The Boost Graph Library functions [28] to identify the two components of the network that have the least number of interactions between each other. When necessary, this division can be repeated recursively. In here, we explain our method on a network that is divided into two subnetworks which are connected by only one interaction of activation type. The discussion can be generalized to partitions connected by more than one interactions. However, the complexity of combining the results of two partitions can increase exponentially with the number of interactions between them. Therefore, it is important to use partitioning of this type for only weakly dependent sub-networks. The good news is that, the weakly dependent subnetworks are commonly observed in real BRN data due to the sparsity of interactions. For instance, the average number of interactions per gene is only 1.12 in average in the real networks we use in our experiments. We discuss the applicability of our partitioning strategy on real data sets in Section 4.2.

Let us turn our attention back to the case with two sub-networks connected by a single activation. Let P and R denote these partitions and $A_p \in P$ and $B_r \in R$ be the two genes such that A_p activates B_r (i.e. $A_p \rightarrow B_r$). Define another partition R' such that $R' = R \cup \{A_p\}$. We first identify all the steady states of each of these three partitions using our algorithm. The tricky part is to combine the steady states of these three partitions to find all the steady states of the original network without losing any information. We do this as follows. We first pick one steady state, say u_P, from the solution space of partition P. Depending on the type of u_P, we have three different cases to consider:

- *Case1 :* u_P *is of Type 0.* In this case, each steady state of partition R' can be combined with u_P to create a combined steady state. If A_p is active in state u_P we combine u_P with only the steady states of R' that have gene A_p in active state. The case when A_p is inactive is combined similarly.
- *Case2 :* u_P *is of Type 1 and the state of* A_p *is fixed in the steady state cycle that contains* u_P. If gene A_p is inactive in u_P, we combine it with only the Type 0 steady states of partition R. When A_p is always zero in a cyclic steady state of P, the activation

between A_p and B_r has no effect on the states of R and two partitions act independently. In the case when A_p is always active, we only combine u_P with Type 0 steady states of R which have A_p in active state.

• *Case3 : u_P is of Type 1 and the state of A_p is not fixed in the steady state cycle that contains u_P.* If gene A_p changes activity level (oscillates) in a steady state cycle with state u_P, then we can only combine u_P with the Type 0 steady states of partition R in which gene A_p has *no effect* on the state of gene B_r. The only case when A_p can have a state changing effect on B_r is when all the other activators and all inhibitors of gene B_r are in inactive state. Hence, we combine u_P with only the steady states of R in which at least one activator or an inhibitor of B_r is in active state.

4 RESULTS AND DISCUSSION

In this section, we evaluate the performance and accuracy of our algorithm through various experiments on real BRNs. We compare the significance of the steady states found by our algorithm to both experimentally observed and computationally predicted steady states. We measure the number of steady states observed, running time and the activity levels of individual genes and gene pairs in steady states.

Code: We implement our algorithm using C++. We use the BDD implementation of the BuDDy package [15] as our data structure. For partitioning the BRNs, we use Boost Graph Library [28]. We obtain the executable of the method proposed by Garg *et al.*[9] from the authors.

Dataset: We use the BRNs that are available from the literature. Also, we use the signaling pathways and gene regulatory networks available in the KEGG Pathway database [19].

Environment: We run all the experiments on a desktop computer running Ubuntu 8.04 with a 3.20 GHz processor and 2 GB of RAM.

4.1 Biological Significance

To evaluate the accuracy of the results reported by our algorithm, we compared the steady states that we found with the steady states that are reported in the literature. For this purpose, we use the genetic regulatory network that controls flower morphogenesis in *Arabidopsis thaliana* which has been studied in detail in several papers [17, 6].

The network shown in Figure 4 is taken from Mendoza *et al.* [17]. They analyzed this network using generalized logical analysis. They focused on two subnetworks, namely the subnetworks that contain the genes {AP3, PI} and {TFL1, LFY, AP1, AG}. They enumerated all the possible states of each subnetwork and exhaustively identified all their steady states. We use the vector notation to represent the activity levels of an ordered gene set. For instance, for a gene set of $\{g_1,\ g_2,\ g_3\}$ [010] represents the state when g_2 is active and g_1, g_3 are inactive. Using this notation, the two steady states found for {AP3, PI} are [00] and [11]. For the subnetwork of {TFL1, LFY, AP1, AG} they considered two cases depending on the activity of flower inhibitor gene EMF1. If EMF1 is active [1000] is found as the only steady state for {TFL1, LFY, AP1, AG} subnetwork. For the case when EMF1 is inactive, [0001] and [0010] are found as the two possible steady states. When identifying these steady states, they did not consider the effect of input genes (LUG, UFO and SUP) and the interactions between two subnetworks. They assumed independent behavior for two subnetworks and only transient activity for the three inputs. *However, these assumptions prevent them from combining the steady states correctly and cause loss of biologically meaningful steady states as we will discuss next.*

Assume that the states we give in vector notation will correspond to the states of the genes in the order EMF1, TFL1, LFY, AP1, AG, LUG, UFO, AP3, PI and SUP. The six steady states found by Mendoza *et al.* are described by using *ABC* combinatorial model proposed for flower morphogenesis [5] as follows: [0001000000] ⇒ *A function*, [0001000110] ⇒ *A and B functions*, [0000100110] ⇒ *B and C functions*, {[1100000000], [1100000110]} ⇒ *non-floral state* and [0000100000] ⇒ *C function*. Our algorithm identified the last four steady states when A function is not active. The two states with active genes of {AP1} and {AP1, AP3, PI} are not reported by our method. Indeed, there is a high parallelism between the activity levels of AP1 and LFY genes [3]. Additionally, they have a mutually reinforcing effect which is denoted by the positive feedback loop in Figure 4. Due to these reasons, the two states with active AP1 and inactive LFY proposed by Mendoza *et al.* for describing the states of genes when A function is active are not realistic. To express the states when only A function is active and when both A and B functions are active more realistically, we propose the two steady states [0011000001] and [0011000110] which are identified by our algorithm. The first one corresponds to high activity levels of AP1, LFY and SUP genes. This describes the case when only A function is active. The reason why SUP is active in this state is that it suppresses the activator effect of LFY on the genes related to B function. That way it keeps A as the only active function in this state. The second state we propose corresponds to the situation when SUP loses its inhibition effect on AP3 and PI. In that case, the active genes are AP1, LFY, AP3 and PI. The first two genes are responsible from A function and the last two are responsible from B function.

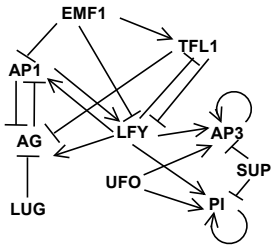


Fig. 4: Regulatory network of flower morphogenesis in *Arabidopsis Thaliana*. Pointed arrow heads represent activation and arrow heads with bars represent inhibition.

The two states that we propose above cannot be found by Mendoza *et al.*'s method. The reason is, when they separate the network into two parts they do not consider how LFY activity can lead to different transitions than the case when the network is complete. Since our algorithm takes only milliseconds for identifying all the steady states of this network, we did not partition the network. However, even if it was the case that we partitioned the network, our method would have assured that none of the steady states are lost by considering all the possible cases while combining the results of partitions as we described in Section 3.5.

We provide additional results on co-expression of genes to test the biological significance of the results in the Supplementary Materials.

4.2 Comparison with Existing Methods

In this section, we compare the performance of our method to that of Garg *et al.* [9]. We compared the number of steady states found by each algorithm and the running times for a number of real BRNs taken from KEGG. Table 1 reports the results for two different versions of their method and our method. We explain what we mean by different versions in the caption of this table.

When we use the original BRNs as inputs, Garg *et al.*'s method crashed for most of the cases. We believe that, one of the reasons of this is that their method assumes that the input genes without self activations are always inactive. However, this is not the case in many real BRNs. Additionally, this assumption literally disconnects input genes from the network which results in a considerably smaller but not complete model of the real BRN. As a result, the method of Garg *et al.* misclassifies a significant portion of the steady states. An example for that case is *p53 network* given in Table 1. For this network, we report 9.4E+13 steady states, whereas their method could find only 1 trivial steady state which is when all genes are inactive. Thus, their assumption in modeling degrades the accuracy and the applicability of their algorithm significantly.

We tried to fix the above problem in Garg *et al.* by adding self activations to all the genes in the networks. Interestingly, the numbers of steady states for modified networks are significantly larger than the ones for the original versions. Also, we observed that their method does not crash for any of the modified input networks. However, it still incurs *false positives* and *false negatives*. The first row of the Table 1 provides an example when the modified version of Garg *et al.*'s method incurs false negatives (i.e. misses some of the steady states). The second row, T-Helper network, shows the case when it has false positives (i.e. transient states are classified as steady).

The running times in Table 1 indicate that, even when we do not use our partitioning strategy, our method runs significantly faster than the modified version of the method of Garg *et al.* for all the cases. It took around 1.5 minutes for our method to identify the steady states for the largest network (*MAPK*), whereas their algorithm could not find all the steady states in 10 days.

The last column in Table 1 reports the running times in bold for the cases when partitioning into independent sub-networks resulted in partitions having more than 20 genes. For these cases, we used our partitioning strategy of separating weakly-dependent components. For instance, for the *p38-MAPK* network with one connected component (26 genes) our algorithm took 21 minutes. When we divide the network into two components with 17 and 9 genes that are connected by only one edge, the running time dropped to 0.2 seconds.

The above results showed that our algorithm is more accurate and significantly faster than the method of Garg et al. Furthermore, our non-trivial partitioning strategy makes it scalable for even large networks. Therefore, we believe that our method can be used for accurate identification of all the steady states for even the large scale BRNs.

4.3 Accuracy of Estimators

To evaluate the quality of our sampling-based estimators, we measured their correctness and convergence rate. Correctness means that the estimates will eventually converge to the correct value. For the convergence rate, a good estimator should approximate the correct value after a small fraction of the state space is explored.

We use a portion of p53 network in this experiment. We measure the estimated number of steady states at which a gene is active for each gene at each iteration of our algorithm. Our algorithm traverses the entire space of Type 1 states in approximately 2,500 iterations for this network. Figure 5 shows the results for seven different genes. We plot these genes as they have different characteristics. They vary in the fraction of steady states in which they are active (e.g. CHK3 is active whereas p21 is suppressed in many steady states). The results show that our estimators converge to the correct ratio for all genes in less than 500 iterations. The rapid convergence suggests that our algorithm approximates the correct profile of gene levels without traversing the whole space of Type 1 states. Hence, our algorithm is also practical for large BRNs with high connectivity which is the case when partitioning the network becomes difficult.

4.4 Co-expression of gene pairs

As discussed in Section 3.3 of the paper, we calculate the fraction of steady states in which a two genes are in active state together. Biologically this fraction corresponds to the co-expression of the two genes. Revealing co-expressed genes has great significance in discovery of conserved genetic modules [27] and identification of differentially expressed genes [20].

In here, we compare the co-expression values for gene pairs found by our algorithm with the values reported in the gene co-expression database, COXPRESdb [18]. For this purpose, we use *Hedgehog signaling network* of *Homo Sapiens* given in KEGG. This network consists of 17 genes and hence, 136 possible gene pairs. We sorted the gene pairs according to their co-expression values in a decreasing order and compared our ordering with the one in COXPRESdb. We picked top 20 gene pairs from our list and searched for the indices of these pairs in the ordering of COXPRESdb. Here, we report the largest index, l, among these k indices for different values of k.

For $k = 1$ we have $l = 1$, which means that the highest co-expressed gene pair (GL1-SMO) in our ordering is also the top scoring pair in COXPRESdb. For $k = 5$ we have $l = 6$, meaning that the five gene pairs (GL1-SMO, GSK3B-FBXW11, RAB23-GAS1, GLI1-IHH and SUFU-SMO) with the highest ranks in our ordering are in between the top 6 pairs in the ranking of COXPRESdb. For the other values of $k = 10$ and $k = 15$, the l values are 16 and 35 respectively. Hence, the gene pairs reported by our method that are found to be active together in the steady states suggest that there is a co-expression between these two genes. This suggests that our algorithm is useful in predicting co-expression of genes in BRNs.

Table 1. The comparison of our algorithm with an existing method on eight different BRNs. [1] The method of Garg *et al.* when we use BRNs as they are given in KEGG database. [2] The method of Garg *et al.* when we add self activations for all the genes in original BRNs. [3] Our algorithm without partitioning the BRNs into weakly-connected sub-networks. [4] Our algorithm when we partition the weakly-connected sub-networks of size larger than 20. "-" indicates that the corresponding method crashed and "'>10 days' indicates that the method could not finish reporting all the steady states after 10 days.

	Network size		Number of Steady States			Total Running Time (seconds or days)			
Network	Genes	Edges	Garg *et al.*[1]	Garg *et al.*[2]	Our Algo.[3,4]	Garg *et al.*[1]	Garg *et al.*[2]	Our Algo.[3]	Our Algo.[4]
Hedgehog	17	11	-	616	11,700	-	4.79s	0.02s	0.02s
T-Helper	23	35	3	4,685	2,152	0.178s	723s	48s	**0.9s**
p38-MAPK	26	28	-	23,087	23,087	-	3.6E+4s	1.26s	**0.2s**
fmlp	27	31	-	81,864	81,864	-	4.8E+5s	1.3s	1.3s
Adipocytokine	35	35	40	>665,450	8.3E+6	0.05s	>10 days	125s	**0.4s**
GnRH	40	23	-	47,520	5.8E+7	-	9.8E+4s	0.09s	0.09s
p53	55	56	1	>545,807	9.4E+13	0.2s	>10 days	144s	**0.1s**
MAPK	114	55	-	>304,131	1.17E+30	-	>10 days	91s	**0.13s**

5 CONCLUSION

In this paper, we proposed a computational method that can identify all the steady states of BRNs accurately and efficiently. We built a mathematical model that can explain the changes in the state of BRNs. Using this, we employed BDDs to filter a large percentage of the state space that has only transient states. The BDDs allowed us to pick another subset in the state space which consists of all the Type 0 steady states. For the remaining state space, we developed a randomized traversal algorithm. This algorithm estimates the number of steady states and the expected behavior of individual genes and pairs of genes in the steady state in an online fashion. We formulated a stopping criteria that terminates the randomized algorithm as soon as user supplied percentage of the results are returned with high confidence. We also developed a partitioning-based strategy that makes our algorithm scalable to virtually any large network. Our experiments showed that, our method finds many steady states that cannot be found using existing methods. Furthermore, our algorithm runs significantly faster than the existing methods especially for large networks.

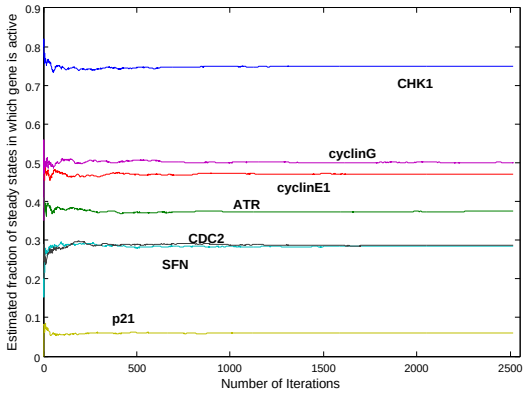


Fig. 5: Convergence of the estimators of the fraction of steady states that each gene is active.

REFERENCES

[1]I. Albert, J. Thakar, S. Li, R. Zhang, and R. Albert. Boolean network simulations for life scientists. *Source code for biology and medicine*, 3:16, 2008.

[2]R. Albert and HG. Othmer. The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in drosophila melanogaster. *Journal of Theoretical Biology*, 223(1):1–18, 2003.

[3]JL. Bowman, J. Alvarez, D. Weigel, EM. Meyerowitz, and DR. Smyth. Control of flower development in arabidopsis thaliana by apetala1 and interacting genes. *Development*, 119(3):721–43, 1993.

[4]A. Ciliberto, B. Novak, and JJ. Tyson. Steady states and oscillations in the p53/mdm2 network. *Cell Cycle*, 4(3):488–93, 2005.

[5]ES. Coen and EM. Meyerowitz. The war of the whorls: genetic interactions controlling flower development. *Nature*, 353(6339):31–7, 1991.

[6]V. Devloo, P. Hansen, and M. Labb. Identification of all steady states in large networks by logical analysis. *Bull Mathematical Biology*, 65(6):1025–51, 2003.

[7]L. Fangting, L. Tao, L. Ying, O. Qi, and T. Chao. The yeast cell-cycle network is robustly designed. *Proceedings of the National Academy of Sciences (PNAS)*, 101(14):4781–6, 2004.

[8]A. Garg, A. Di Cara, I. Xenarios, L. Mendoza, and G. De Micheli. Synchronous versus asynchronous modeling of gene regulatory networks. *Bioinformatics*, 24(17):1917–25, 2008.

[9]A. Garg, I. Xenarios, L. Mendoza, and G. De Micheli. An efficient method for dynamic analysis of gene regulatory networks. and in silico gene perturbation experiments. In *International Conference on Research in Computational Molecular Biology (RECOMB)*, pages 62–76, 2007.

[10]GD. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Markovian analysis of large finite state machines. *IEEE Transactions on Computer-Aided Design*, 15:1479–93, 1996.

[11]AM. Huerta, H. Salgado, D. Thieffry, and J. Collado-Vides. Regulondb: a database on transcriptional regulation in escherichia coli. *Nucleic Acids Research (NAR)*, 26(1):55–9, 1998.

[12]TR. Hupp, DP. Lane, and KL. Ball. Strategies for manipulating the p53 pathway in the treatment of human cancer. *Biochemical Journal*, 352:1–17, 2000.

[13]S. Kauffman. Homeostasis. and differentiation in random genetic control networks. *Nature*, 224(5215):177–8, 1969.

[14]G. Lahav, N. Rosenfeld, A. Sigal, N. Geva-Zatorsky, AJ. Levine, MB. Elowitz, and U. Alon. Dynamics of the p53-mdm2 feedback loop in individual cells. *Nature Genetics*, 36(2):147–50, 2004.

[15]J. Lind-Nielsen. Buddy - a binary decision diagram package. tech. rep, http://cs.it.dtu.dk/buddy. 1999.

[16]L. Mendoza. A network model for the control of the differentiation process in th cells. *Biosystems*, 84(2):101–14, 2006.

[17]L. Mendoza, D. Thieffry, and ER. Alvarez-Buylla. Genetic control of flower morphogenesis in arabidopsis thaliana: a logical analysis. *Bioinformatics*, 15(7):593–606, 1999.

[18]T. Obayashi, S. Hayashi, M. Shibaoka, M. Saeki, H. Ohta, and K. Kinoshita. Coxpresdb: a database of coexpressed gene networks in mammals. *Nucleic Acids Research (NAR)*, 36(Database issue), 2008.

[19]H. Ogata, S. Goto, K. Sato, W. Fujibuchi, H. Bono, and M. Kanehisa. Kegg: Kyoto encyclopedia of genes. and genomes. *Nucleic Acids Research (NAR)*, 27(1):29–34, 1999.

[20]MC. Oldham, S. Horvath, and DH. Geschwind. Conservation. and evolution of gene coexpression networks in human. and chimpanzee brains. *Proceedings of the National Academy of Sciences (PNAS)*, 103(47):17973–8, 2006.

[21]S. Ostergaard, L. Olsson, M. Johnston, and J. Nielsen. Increasing galactose consumption by saccharomyces cerevisiae through metabolic engineering of the gal gene regulatory network. *Nature Biotechnology*, 18(12):1283–6, 2000.

[22]MA. Schaub, TA. Henzinger, and J. Fisher. Qualitative networks: a symbolic approach to analyze biological signaling networks. *BMC Systems Biology*, 1(4), 2007.

[23]T. Shlomi, O. Berkman, and E. Ruppin. Regulatory on/off minimization of metabolic flux changes after genetic perturbations. *Proceedings of the National Academy of Sciences (PNAS)*, 102(21):7695–700, 2005.

[24]I. Shmulevich, ER. Dougherty, S. Kim, and W. Zhang. Probabilistic boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18(2):261–74, 2002.

[25]I. Shmulevich, ER. Dougherty, and W. Zhang. Gene perturbation and intervention in probabilistic boolean networks. *Bioinformatics*, 18(10):1319–31, 2002.

[26]EH. Snoussi and R. Thomas. Logical identification of all steady states: The concept of feedback loop characteristic states. *Bull Mathematical Biology*, 55:973–91, 1993.

[27]JM. Stuart, E. Segal, D. Koller, and SK. Kim. A gene-coexpression network for global discovery of conserved genetic modules. *Science*, 302(5643):240–1, 2003.

[28]H. Sutter and A. Alexandrescu. The boost graph library (bgl) http://www.boost.org/.

[29]R. Thomas. Logical analysis of systems comprising feedback loops. *Journal of Theoretical Biology*, 73(4):631–56, 1978.