RESEARCH ARTICLE

An Efficient Steady-State Analysis Method for Large Boolean Networks with High Maximum Node Connectivity

Changki Hong[1], **Jeewon Hwang**[1], **Kwang-Hyun Cho**[2], **Insik Shin**[1]*

1 School of Computing, KAIST, Daejeon, Korea, **2** Department of Bio and Brain Engineering, KAIST, Korea

* insik.shin@cs.kaist.ac.kr

Citation: Hong C, Hwang J, Cho K-H, Shin I (2015) An Efficient Steady-State Analysis Method for Large Boolean Networks with High Maximum Node Connectivity. PLoS ONE 10(12): e0145734. doi:10.1371/journal.pone.0145734

Editor: Attila Csikász-Nagy, King's College London, UNITED KINGDOM

Received: July 31, 2015

Accepted: December 8, 2015

Published: December 30, 2015

Data Availability Statement: All relevant data are within the paper and its Supporting Information files.

Funding: This work was supported in part by IITP (B0101-15-0557) and NRF funded by the Korea Government, the Ministry of Science, ICT & Future Planning (2013M3A9A7046303, 2014R1A2A1A10052404 and 2015M3A9A7067220). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing Interests: The authors have declared that no competing interests exist.

Abstract

Boolean networks have been widely used to model biological processes lacking detailed kinetic information. Despite their simplicity, Boolean network dynamics can still capture some important features of biological systems such as stable cell phenotypes represented by steady states. For small models, steady states can be determined through exhaustive enumeration of all state transitions. As the number of nodes increases, however, the state space grows exponentially thus making it difficult to find steady states. Over the last several decades, many studies have addressed how to handle such a state space explosion. Recently, increasing attention has been paid to a satisfiability solving algorithm due to its potential scalability to handle large networks. Meanwhile, there still lies a problem in the case of large models with high maximum node connectivity where the satisfiability solving algorithm is known to be computationally intractable. To address the problem, this paper presents a new partitioning-based method that breaks down a given network into smaller subnetworks. Steady states of each subnetworks are identified by independently applying the satisfiability solving algorithm. Then, they are combined to construct the steady states of the overall network. To efficiently apply the satisfiability solving algorithm to each subnetwork, it is crucial to find the best partition of the network. In this paper, we propose a method that divides each subnetwork to be smallest in size and lowest in maximum node connectivity. This minimizes the total cost of finding all steady states in entire subnetworks. The proposed algorithm is compared with others for steady states identification through a number of simulations on both published small models and randomly generated large models with differing maximum node connectivities. The simulation results show that our method can scale up to several hundreds of nodes even for Boolean networks with high maximum node connectivity. The algorithm is implemented and available at http://cps.kaist.ac.kr/~ckhong/tools/download/PAD.tar.gz.

Introduction

Modeling of biological systems as a network of interacting components has received increasing attention in various areas, such as computational and systems biology since it allows to analyze biological phenomena systematically at various scales including molecular and cellular levels [1]. Boolean networks (BNs) have been widely used among various network models because BNs are relatively simple and efficient to model large networks [2–4]. The BN is a discrete model of biological system that comprises of a number of nodes and corresponding update rules. Each node represents a gene and takes on a value of 1 or 0, meaning that the gene is expressed or unexpressed, respectively. Each update rule represents interactions between genes. The state of a gene at a given time step is determined by its update rule and the state of its input genes at the previous time step. In synchronous BNs, the states of all nodes are updated simultaneously at each time step, and it directly induces global state transitions. An important characteristic of BNs is that any sequence of consecutive global state transitions eventually converges to either a single state (*i.e., steady state*) or a cycle of states (*i.e., cyclic attractors*).

It is important to identify steady states to have a proper understanding of biological systems because steady states often convey biological implications. Specifically, steady states capture long-term behaviors of biological systems: they are closely related to different cell types or states (*e.g.,* growth, differentiation, and apoptosis) [5–7]. This motivated several meaningful case studies of using steady-state analysis. For example, the identification of steady states has been playing a crucial role in treatment of various human cancers such as breast cancer, and leukemia [8, 9]. Additionally, the steady-state analysis can successfully explain the flower morphogenesis of Arabidopsis thaliana [10], the differentiation process of T-helper cells [11], the mechanism of T cell receptor signaling [12], the cell cycles of yeast types [13, 14], and the express patterns of Drosophila melanogaster segment polarity genes [15]. Theoretically, steady states can be detected by exhaustively exploring all the global states of a BN. However, it becomes too memory- and time-consuming even for a small BN with n nodes since 2^n global states need to be examined in total. Indeed, it has been proved that the problem of finding fixed points in a Boolean network is NP-hard [16]. Hence, it is essential to develop an efficient algorithm to detect steady states while avoiding such state space explosion.

Algorithms for the problem of finding steady states have been extensively studied in the past decade [17–30]. A common approach is to convert explicit state transitions to implicit representations: decision diagram (DD), and propositional logic formula (SAT). In algorithms based on DD [18–21], a DD represents a Boolean update rule. Then, by combining the DD representations of all the Boolean update rules, the problem of finding steady states becomes a search problem in the larger DD. This limits DD-based algorithms to small BNs with about 100 nodes. The SAT-based algorithms [22–25], however, do not suffer from the potential space explosion of DDs, but most of them have focused on large BNs with low maximum node connectivity (*i.e.,* maximum indegree, K). For example, Tamura *et al.* [22] proved that the problem of detecting steady states of a BN with K can be transformed to $(K + 1)$-SAT problem. Those algorithms take advantage of modern success of SAT solvers [31], which enable to scale up to hundreds of nodes for BNs with $K < 2$ [24]. However, in case of BNs with higher Ks, a state explosion still occurs because no efficient SAT solvers are known for the $(K + 1)$-SAT problem with $K \geq 2$, which is a well-known NP-complete problem [32]. This limits the SAT-based algorithms to Boolean networks with low Ks only.

To expand the range of feasible BNs, partitioning-based steady states detection algorithms have been published recently [25, 33]. The key idea of those approaches is to reduce the basic unit of steady-state analysis. For example, Guo *et al.* [25] partition the given network into smaller blocks based on a strongly connected component (SCC). Steady states are independently detected in each block by applying a SAT solver, and then combined to construct the steady states of the original BN. Therefore they achieved a better scalability on randomly generated BNs with $K \leq 3$. The scalability gained in principle is, however, unlikely to happen in the models of realistic biological processes. This is because K of these models is known as orders of magnitude higher than the average indegree [34]. For BNs with such high K, the size of the largest SCC is too large to be analyzed within a reasonable timing. Several studies have discovered that the currently available protein interaction networks have a SCC connecting the vast majority of the proteins [35–37]. It is also reported that the currently available metabolic and signal transduction networks are connected, with 50 to 60% of the nodes forming the largest SCC [38, 39]. Thus, the partitioning strategy based on SCC is not suitable in the real world, and so a better partitioning method is necessary.

Then, a natural question is what the smallest possible unit for partitioning is. In this manuscript, we aim to discover the smallest partition of the network to utilize a SAT solver in the most efficient way. To this end, this paper proposes an optimal partitioning algorithm based on the *minimum essential block* (MEB). Then, we build upon the idea of the MEB-based network partition into subnetworks. Each subnetwork is guaranteed to be the smallest both in the size (*i.e., n*) and the maximum indegree (*i.e., K*) such that any deletion of nodes or edges from the subnetwork cannot correctly determine the steady states of the network. Thus, the proposed MEB-based partition guarantees to maximize the performance of a SAT solver while ensuring to determine all the steady states reliably. We implemented an experimental tool and compared with other state-of-the-art methods [24, 25]. For the models of real biological processes acquired from literature [9–15], the runtime of the proposed algorithm performs about three times better than others in average. Since the models consist of several tens of nodes only, we randomly generated larger networks to evaluate how well our approach scales up. In comparison to the other methods, our method performs favorably on the large network with high K, and reliably finds all steady states even for networks with up to $n = 1000$ and $K = 5$.

The rest of this paper is organized as follows: The Methods section describes the model, definitions and the proposed algorithm. In the Results section, we demonstrate the correctness and efficiency of our method. The Discussion section sums up the results of the study.

Methods

The Boolean network and attractors

A *Boolean network* (BN) $G = \langle V, F\rangle$ consists of a set V of n nodes and a set F of Boolean update rules, where

$$\begin{aligned} V &= \{v_1, v_2, \ldots, v_n\} \\ F &= \{f_1, f_2, \ldots, f_n\}. \end{aligned}$$

Each node $v_i \in V$ has its associated state $s_i(t) \in \{0, 1\}$ at any time instant t and a Boolean update rule f_i. The Boolean update rule f_i: $\{0, 1\}^{k_i} \rightarrow \{0, 1\}$ determines the state value of v_i (*i.e.*, $s_i(t)$) according to the state of k_i neighbor nodes incoming to v_i at previous time instant $t - 1$. Here, k_i is the number of inputs for v_i, and called *indegree* of v_i. The maximum indegree of a Boolean network is defined as $K = \max\{k_i\}$. The states of all nodes (*i.e.,* global state) are updated simultaneously at each time instant. We denote the global state of the Boolean network at time t by a vector $S(t) = \langle s_1(t), s_2(t), \ldots, s_n(t)\rangle$. Hence, the synchronous updating of all nodes directly induces global state transitions as follows:

$$F(S(t)) = S(t + 1)$$

The state transition represents the dynamics of the network. One of the main characteristics of Boolean network dynamics is that for any initial state condition, the system eventually converges to *attractors*. Consecutive global state transitions $S(t)$, $S(t + 1)$, . . ., $S(t + p)$ are called an attractor with period p if $S(t) = S(t + p)$. An attractor with period $p = 1$ is called a *steady state* (*i.e.*, a *fixed point*), and an attractor with a period $p > 1$ is called a *cyclic attractor*.

An efficient steady-state identification method through partitioning

The state space of a BN grows exponentially with the number of nodes in the network. It becomes more complex as the maximum indegree increases. In this paper, we divide the given network into multiple smaller subnetworks to decrease the computation complexity. We then independently analyze each subnetwork to find *local* steady states by applying a SAT solver. Finally, we compose the local steady states to construct the steady states of the overall network. Note that the steady states detected in each subnetwork are called local steady states to distinguish them from the steady states of the given original BN.

Intuitive idea. The steady states identified through partitioning should be identical to the steady states detected directly from the original BN. This and the following section present how our partition-based method can be applied to identify steady states in a complete and correct manner. For ease of discussion, we first describe an intuitive idea of how we divide a network into two subnetworks and how we combine the steady states of them to construct the steady states of the overall network. This discussion will be extended to include the case of multiple subnetworks.

Let us assume that the given BN G is partitioned into two blocks P and Q as shown in Fig 1A. Block Q depends on block P through an out-going edge from node c_p. Thus, state transitions of Q are regulated by the values of c_p at each time instant. We define such c_p as a control node of Q. If there exists a steady state of G, say a, P will eventually converge to a_p (*i.e.*, a local steady state of P, which comprises the part of the steady state a of G) at some time instant t_k regardless of Q. From the time instant t_k, the *fixed* value of c_p of a_p keeps controlling state transitions of Q, then Q also arives at a_q (*i.e.*, a local steady state of Q, which comprises the part of the steady state a of G). Finally, the composition of a_p and a_q can safely construct a.

Based on the aformentioned observations, we present two-block partitioning and composing methods that correctly determine steady states of the given network G. Once G is partitioned into two blocks P and Q, we construct two subnetworks based on these blocks. Here, it is worth noting that P can be independently analyzed to find its local steady states regardless of Q. Block Q, however, cannot be correctly analyzed by itself since it depends on a control input (*i.e.*, c_p) from outside. We thus propose to construct a new subnetwork Q' for block Q such that $Q' = Q \cup \{c_p\}$. For the steady-state analysis of Q', the value of c_p is assumed to be fixed to 0 or 1, and controls state transitions of Q. We then join each local steady state of P with that of Q' by using the state of c_p as a *glue* between them. That is, we compose the two local steady states into the steady state for G if c_p has the same value in both subnetworks.

In the case that there also exists a set of out-going edges from node c_q of Q as shown in Fig 1B, we construct two subnetworks P' and Q', where $P' = P \cup \{c_q\}$ and $Q' = Q \cup \{c_p\}$. For each subnetwork, local steady states are computed under the assumption that the value of its input is fixed and controls state transitions of the subnetwork. Thus, composing two local steady states that have the same values of c_p and c_q in both subnetworks can reliably construct the steady states.

Partitioning and correctness. The two-block partitioning and composing methods presented in the previous section can be extended towards partitioning of a BN into multiple blocks. Let us consider the given BN $G = \langle V, F \rangle$ with n nodes, and V is partitioned into m-

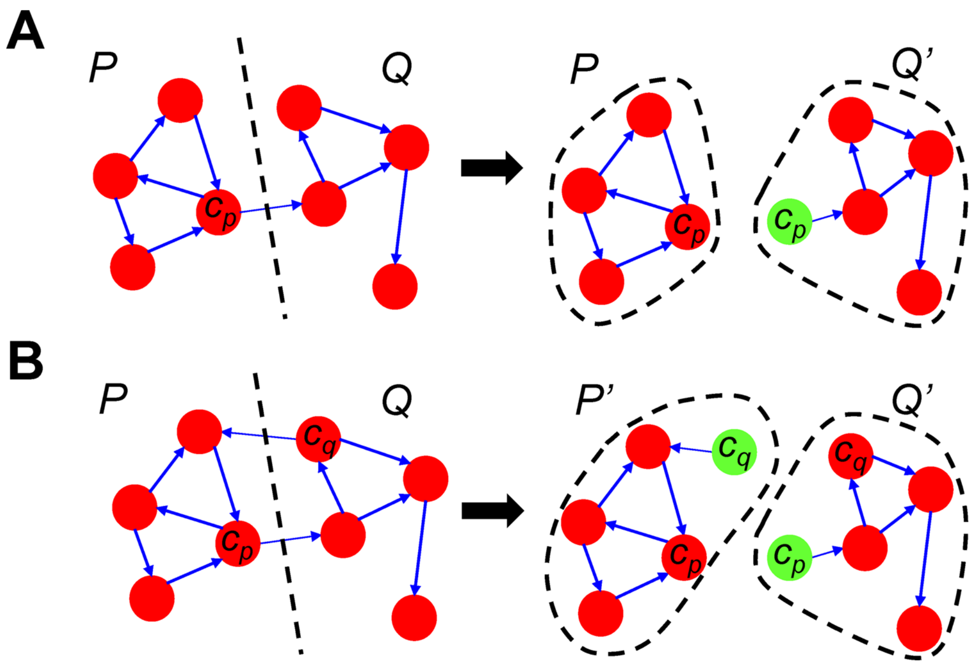

Fig 1. An intuitive idea of partitiong a Boolean network into two blocks and composing steady states found independently. A: A Boolean network *G* can be partitioned into two blocks *P* and *Q* which are connected by c_p. Then, we construct two subnetworks *P* and *Q*′ where $Q' = Q \cup \{c_p\}$, and c_p is used to combine the local steady states found in each subnetwork. B: When there also exists a set of edges from c_q of *Q*, we construct two subnetworks *P*′ and *Q*′ where $P' = P \cup \{c_q\}$ and $Q' = Q \cup \{c_p\}$. Both c_p and c_q are used to combine the steady states of subnetworks.

doi:10.1371/journal.pone.0145734.g001

number of blocks as follows:

$$V = \{v_1, v_2, \ldots, v_n\} = V_1 \cup V_2 \cup \ldots \cup V_m$$

where V_i is a proper subset of *V*, $V_i \cap V_j$ is empty for $i \neq j$. Each block V_i has incoming edges from outside of the block, and the source nodes of these edges can be interpreted as inputs for each block. Let us denote the set of inputs of the block V_i as C_i. Then, the subnetwork G_i is constructed to be used to reveal local steady states as follows:

$$G_i = \langle V_i \cup C_i, F_i \rangle$$

where F_i is a set of Boolean update rules associated to the nodes of V_i. It is worth noting that the Boolean update rules for nodes in C_i are ignored since they are regarded as inputs to V_i. But, a fixed state value (*i.e.,* 0 or 1) for each node in C_i controls state transitions of the block V_i, and thus input nodes are also called as control nodes.

The proposed composing method can also be generalized to the cases of combining local steady states of multiple subnetworks. Before we describe the composition procedure, we define $SV(a_k, T)$ as a subvector extracted from a steady-state vector, a_k, according to a set of target output variables, *T*. For example, let us assume that a steady state vector, $a_k = \langle 0, 1, 0, 1, 1 \rangle$, is ordered as $(v_1, v_2, v_3, v_4, v_5)$. The subvector extracted from a_k with respect to the targets, $\{v_2, v_3, v_4\}$, (*i.e., SV*$(\langle 0, 1, 0, 1, 1 \rangle, \{v_2, v_3, v_4\})$) is thus $\langle 1, 0, 1 \rangle$. Then, we define the composition of two local steady states from distinct subnetworks. Let us assume that $G_1 = \langle V_1 \cup C_1, F_1 \rangle$ and $G_2 =$

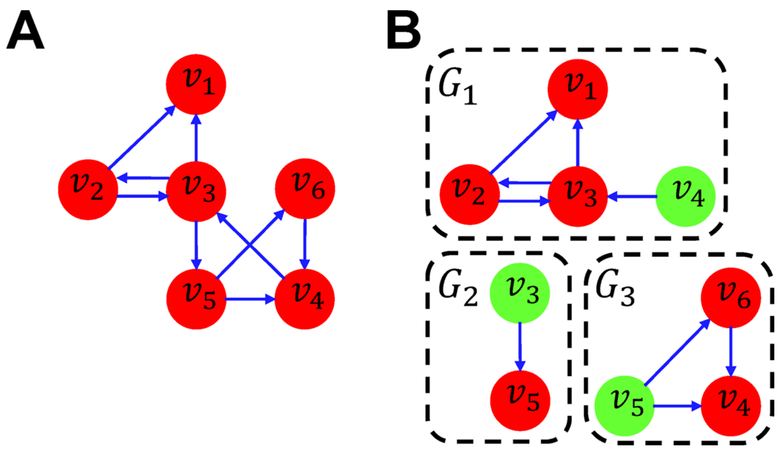

Fig 2. A division into three subnetworks. A: A Boolean network model *G*. B: A division into three subnetworks (*e.g., G*$_1$, *G*$_2$, and *G*$_3$). Input nodes are depicted as green nodes.

doi:10.1371/journal.pone.0145734.g002

$\langle V_2 \cup C_2, F_2 \rangle$ are the two subnetworks, and

$$\begin{aligned} V_1 \cup C_1 &= \{v_1, \ldots, v_k, v_{k_1+1}, \ldots, v_{k_1+l}\} \\ V_2 \cup C_2 &= \{v_1, \ldots, v_k, v_{k_2+1}, \ldots, v_{k_2+m}\} \end{aligned}$$

where $v_1, \ldots, v_k$ are the common components in two subnetworks. If two local steady states (*e.g.*, a_1 and a_2 for G_1 and G_2, respectively) meet the following condition,

$$SV(a_1, \{v_1, \ldots, v_k\}) = SV(a_2, \{v_1, \ldots, v_k\})$$

then a_1 and a_2 can be composed (*i.e.*, $a_1 \times a_2$) as follows:

$$a_1 \times a_2 = a_1 + SV(a_2, \{v_{k_2+1}, \ldots, v_{k_2+m}\})$$

where + operator represents the vector concatenation. All the local steady states (*i.e.*, $a_i \in A_1$ and $a_j \in A_2$) satisfying the condition are composed (*i.e.*, $a_i \times a_j$) as described above. The resulting intermediate set of combined local steady states is then composed with local steady states of the next subnetworks in the same manner until the last subnetwork is reached. The entire procedure guarantees the steady states identified through partitioning to be identical to the steady states detected directly from the original BN (for a formal proof, see S1 Text).

As an example, let us consider that a BN $G = \langle V, F \rangle$ with six nodes in Fig 2A, where the Boolean update rule *F* consists of six corresponding Boolean update rules as follows:

$$\begin{aligned} f_1 : s_1(t+1) &= s_2(t) \wedge s_3(t) \\ f_2 : s_2(t+1) &= s_3(t) \\ f_3 : s_3(t+1) &= s_2(t) \vee s_4(t) \\ f_4 : s_4(t+1) &= s_5(t) \leftrightarrow s_6(t) \\ f_5 : s_5(t+1) &= s_3(t) \\ f_6 : s_6(t+1) &= \neg s_5(t) \end{aligned}$$

where $\wedge$, $\vee$, $\leftrightarrow$, $\neg$ denote logical AND, OR, EQUIVALENCE, NOT operations, respectively. Let us assume that *V* is partitioned into three blocks (*i.e.*, $V = V_1 \cup V_2 \cup V_3$) as shown in Fig 2B as red nodes:

$$V_1 = \{v_1, v_2, v_3\}, V_2 = \{v_5\}, V_3 = \{v_4, v_6\}.$$

Then, their inputs are $C_1 = \{v_4\}$, $C_2 = \{v_3\}$ and $C_3 = \{v_5\}$. Finally, three subnetworks (*i.e.,* G_1, G_2 and G_3) can be constructed as follows:

$$G_1 = \langle V_1 \cup C_1, F_1\rangle, G_2 = \langle V_2 \cup C_2, F_2\rangle, G_3 = \langle V_3 \cup C_3, F_3\rangle$$

$$\begin{aligned} F_1 \quad &: \quad \begin{cases} s_1(t+1) = s_2(t) \wedge s_3(t) \\ s_2(t+1) = s_3(t) \\ s_3(t+1) = s_2(t) \vee s_4(0) \end{cases} \\ F_2 \quad &: \quad \{\, s_5(t+1) = s_3(0) \\ F_3 \quad &: \quad \begin{cases} s_4(t+1) = s_5(0) \leftrightarrow s_6(t) \\ s_6(t+1) = \neg s_5(0) \end{cases} \end{aligned}$$

where $s_i(0)$ represents a fixed control value of the input node v_i, which is set to 0 or 1 depending on the initial state $S(0)$. When the state vectors of G_1, G_2, and G_3 are ordered as (v_1, v_2, v_3, v_4), (v_3, v_5), and (v_4, v_5, v_6), respectively, then the local steady states are detected as follows:

$$\begin{aligned} A_1 &= \{\langle 0,0,0,0\rangle, \langle 1,1,1,0\rangle, \langle 1,1,1,1\rangle\} \\ A_2 &= \{\langle 0,0\rangle, \langle 1,1\rangle\} \\ A_3 &= \{\langle 0,0,1\rangle, \langle 0,1,0\rangle\} \end{aligned}$$

where A_1, A_2, and A_3 are the sets of local steady states for G_1, G_2, and G_3, respectively. Firstly, A_1 and A_2 are composed by using the common component v_3 in G_1 and G_2 as the glue between them. As the result, a set of combined local steady states is $\{\langle 0, 0, 0, 0, 0\rangle, \langle 1, 1, 1, 0, 1\rangle\}$ ordered as $(v_1, v_2, v_3, v_4, v_5)$. The set is then composed with A_3. Finally, two steady states (*i.e.,* $\{\langle 0, 0, 0, 0, 0, 1\rangle, \langle 1, 1, 1, 0, 1, 0\rangle\}$) are identified for G as the result of composing all the local steady states, which are the same as the ones detected without partitioning.

The efficient network partition. In previous sections, we have discussed how we can reliably construct steady states of a given network through partitioning. However, there may exist a number of different partitions available for the given network. In this section, thus, we will discuss how to find the best partition of the network so that the cost to identify all local steady states is minimized. In the proposed approach, we use a SAT solver that is known to be efficient to find steady states of a network. Yet, the complexity of SAT solvers increase exponentially with the number of nodes (*i.e., n*) for BNs with $K \geq 2$ [24]. Hence, it is computationally more efficient when we partition each block to be smaller in n and lower in K.

To utilize a SAT solver in the most efficient way, this paper proposes an optimal partitioning algorithm based on the *minimum essential block* (MEB). Given a BN $G = \langle V, F\rangle$ with n nodes, such a partition can be constructed by dividing V into n blocks. Thus, each block V_i consists of only a single corresponding node v_i. As an example, the MEB-based network partition of the BN G of Fig 2A is composed of six blocks such as $V_1 = \{v_1\}$, $V_2 = \{v_2\}$, $V_3 = \{v_3\}$, $V_4 = \{v_4\}$, $V_5 = \{v_5\}$ and $V_6 = \{v_6\}$. As shown in Fig 3, six subnetworks are then constructed based on the blocks as described in the previous section. It is worth noting that each subnetwork is the minimum both in n and K because even if we remove just a single node or an edge from any single subnetwork, the induced alterations of Boolean update rules cause to break the correctness of the algorithm. Therefore, the proposed MEB-based partition guarantees to maximize the

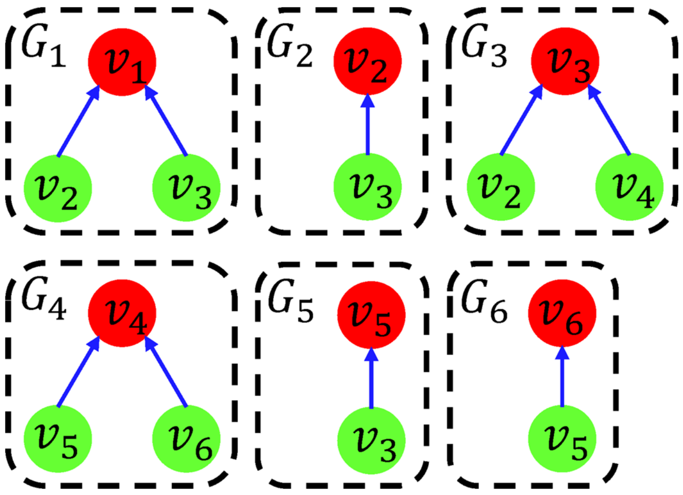

Fig 3. The MEB-based network partition. For the network G in Fig 2A, the MEB-based network partition consists of six blocks each of which has a single node (depicted as a red node). Each subnetwork G_i is constructed based on each block. Input nodes are depicted as green nodes.

doi:10.1371/journal.pone.0145734.g003

performance of a SAT solver while ensuring to determine steady states reliably. A formal proof is given to show that the proposed partition is the best network partition (see S2 Text). The pseudocode of the partitioning algorithm is given in Algorithm 1.

Algorithm 1 Pseudocode of the network partitioning algorithm

```
Input: a Boolean network, $G = \langle V, F \rangle$ with $n$ nodes
Output: an optimal network partition, $G_{min} = [G_1, G_2, \ldots, G_n]$
1: function BEST-PARTITION (BN $G = \langle V, F \rangle$)
2:   $G_{min} \leftarrow [\ ]$
3:   for $i = 1$ to $n$ do
4:     $V_i \leftarrow \{v_i\}$, $F_i \leftarrow \{f_i\}$
5:     for each node $v_k \in V$ do
6:       if $v_k$ is an input node of $v_i$ then
7:         $V_i \leftarrow V_i \cup \{v_k\}$
8:       end if
9:       $G_i \leftarrow \langle V_i, F_i \rangle$
10:      $G_{min} \leftarrow [G_{min}, G_i]$
11:    end for
12:  end for
13:  return $G_{min}$
14: end function
```

The efficient steady-state composition algorithm. The overall performance of our algorithm can be further improved when we compose each set of local steady states in a proper order. This is because composing the local steady states in a wrong order may lead to the generation of many redundant intermediate composed steady states which will be discarded later on. If we consider our previous example, we compose local steady states in the order of (A_1, A_2, A_3) resulting in two steady states, $\{\langle 0, 0, 0, 0, 0, 1\rangle, \langle 1, 1, 1, 0, 1, 0\rangle\}$. The composition order (A_1, A_3, A_2) can also construct the same steady states, but redundant intermediate composed steady states are generated during the composition. That is, the composition A_1 and A_3 results in $\{\langle 0, 0, 0, 0, 0, 1\rangle, \langle 0, 0, 0, 0, 1, 0\rangle, \langle 1, 1, 1, 0, 0, 1\rangle, \langle 1, 1, 1, 0, 1, 0\rangle\}$, which include two redundant ones. Recall that the composition order (A_1, A_2, A_3) does not generate any redundancy in intermediate composed steady states as shown in the previous example. Thus, the best composition order is the one that generates the least redundant intermediate composed steady states in the course of identification of all steady states. It is, however, generally complicated to find such an optimal composition order in advance since it is often sophisticated to predict which local steady states will be identified in each subnetwork. We thus provide a heuristic algorithm that determines an efficient composition order. The pseudocode of the algorithm is given in Algorithm 2. At each step, the algorithm chooses a subnetwork whose combination with the current combined subnetwork generates the largest number of common nodes between them. The local steady states of the chosen subnetwork are then determined to be composed next. It is worth noting that our composition algorithm composes two local steady states only if the state values of common nodes between them are the same. Thus, it is more likely to filter out redundant intermediate composed steady states when composing with local steady states whose corresponding subnetwork has more common nodes than others at each step. The pseudocode of the composition algorithm is given in Algorithm 3.

Algorithm 2 Pseudocode for determining an order of composing local steady states

```
Input: an optimal network partition, $G_{min}$ = [$G_1$, $G_2$, ..., $G_n$]
Output: a composition order, $O_c$
1: function DETERMINE-COMPOSITION-ORDER (list $G_{min}$)
2:    $begin \leftarrow arg\max_i \{|V_i|\}$          /* $V_i$ of $G_i \in G_{min}$ */
3:    $O_c \leftarrow$ [ begin]
4:    $V_{curr} \leftarrow V_{begin}$
5:    while |$O_c$| = $n$ do
6:       $next \leftarrow arg\max_{i \neq j} \{|V_{curr} \cap V_i|\}$     /* $j \in O_c$ and $V_i$ of $G_i \in G_{min}$ */
7:       $V_{curr} \leftarrow V_{curr} \cup V_{next}$
8:       $O_c \leftarrow$ [ $O_c$, next]
9:    end while
10:   return $O_c$
11: end function
```

Algorithm 3 Pseudocode of the composition algorithm

```
Input: an optimal network partition, $G_{min}$, a list of local steady states,
$A_{loc}$, and a composition order, $O_c$
Output: a set of composed steady states, $A_{comp}$
1: function ATTRACTOR-COMPOSITION (list $G_{min}$, list $A_{loc}$, list $O_c$)
2:    $s \leftarrow O_c[0]$
3:    $A_{comp} \leftarrow A_s$, $V_{comp} \leftarrow V_s$          /* $A_s \in A_{loc}$ and $V_s$ of $G_s \in G_{min}$ */
4:    for each $i \in O_c$ do
5:       $C \leftarrow V_{comp} \cap V_i$
6:       $A_{temp} \leftarrow \emptyset$
7:       for each $a_p \in A_{comp}$ do
8:          for each $a_q \in A_i$ do
9:             if $SV(a_p, C) = SV(a_q, C)$ then
10:               $A_{temp} \leftarrow A_{temp} \cup \{a_p \times a_q\}$
11:            end if
12:         end for
13:      end for
14:      $A_{comp} \leftarrow A_{temp}$
15:      $V_{comp} \leftarrow V_{comp} \cup V_i$
16:   end for
17:   return $A_{comp}$
18: end function
```

Time complexity analysis. Finding steady states of Boolean networks has been proved as NP-hard by Akutsu *et al.* in 1998 [16]. Thus, it is not plausible that the steady-state identification problem can be solved efficiently (*i.e.,* polynomial time) in all cases. However, it is possible to develop algorithms that are fast in the average case, and thus this manuscript studies an algorithm that is efficient in many practical cases.

The proposed algorithm has three main parts as follows: (1) partition the Boolean network, (2) find all local steady states for each subnetwork, and (3) combine the local steady states to the steady states of the overall network. Comparing to (2), the complexity of (1) is negligible as the computational cost increases polynomially with the number of nodes. The worst case time complexity of (1) is $O(n^2)$ because each subnetwork is constructed by checking n nodes to find all the control nodes of the corresponding block, where n is the size of the network. For (2), on the other hand, the worst case time complexity depends on the performance of a SAT solver, but the time complexity for the $(K + 1)$-SAT problem with $K \geq 2$ (*i.e.,* corresponds to the BN with $K \geq 2$) has been proved as NP-complete. Thus, in general, it is very hard to predict which network instances are going to be hard to solve without actually attempting to solve it. The worst case time complexity of (3) is $O(2^n)$ if the number of steady states of a BN is 2^n. However, we may be less interested in such networks as most of biologically meaningful dynamics evolve into relatively small number of steady states. The mean number of attractors of BNs is shown to be proportional to $\sqrt{n}$ [1]. In average cases, the time complexity of (3) is bounded by $O\left(n^{\frac{3}{2}}\right)$ (for details, see S3 Text).

In summary, the efficiency of the proposed algorithm is mostly determined by the performance of a SAT solver which depends both on n and K. Since the proposed MEB-based partition minimizes the cost of (2), the proposed algorithm can efficiently detect steady states except in rare cases. The theoretical analysis results are also supported by the simulation results as will be discussed in Results section.

Results

We have implemented an experimental tool PAD (Partition-based Attractor Detection tool) based on the presented algorithm. The pseudocode of our overall algorithm is given in Algorithm 4. To find the local steady states of each subnetwork, we use a SAT solving algorithm by Dubrova and Teslenko [24] based on MiniSAT SAT solver [31] as shown in lines 4 to 7 of Algorithm 4. PAD is available at http://cps.kaist.ac.kr/ ∼ ckhong/tools/download/PAD.tar.gz.

Algorithm 4 Pseudocode for the overall steady-state identification algorithm

```
Input: a Boolean network, $G = \langle V, F \rangle$ with $n$ nodes
Output: a set of steady states, $A$
1: procedure MAIN
2:    $G_{MIN} \leftarrow$ BEST-PARTITION($G$)
3:    $A_{LOC} \leftarrow$ [ ]
4:    for each $G_i \in G_{MIN}$ do
5:       $A_i \leftarrow$ MiniSAT($G_i$)
6:       $A_{LOC} \leftarrow$ [ $A_{LOC}$, $A_i$]
7:    end for
8:    $O_C \leftarrow$ DETERMINE-COMPOSITION-ORDER($G_{MIN}$)
9:    $A \leftarrow$ ATTRACTOR-COMPOSITION($G_{MIN}$, $A_{LOC}$, $O_C$)
10: end procedure
```

We analyzed over 3,600 BNs (*e.g.,* real biological process models [9–15], and the $N - K$ random BNs [3, 40]) to benchmark our method against others. The methods we used for the comparison were those with good computational efficiency: BNS (*i.e.,* the state-of-the-art SAT

Table 1. Simulation results for detecting steady states from real Boolean networks.

Models	# of nodes	K	# of steady states	Runtime (sec)		
				BNS [24]	ST [25]	PAD (Ours)
Mammalian cell [10]	10	6	1	0.003	0.006	0.001
Fission yeast [14]	10	6	13	0.003	0.004	0.003
Budding yeast [13]	12	6	7	0.009	0.009	0.005
T-helper cell [11]	23	5	3	0.003	0.004	0.001
T-cell receptor [12]	40	5	8	0.018	0.021	0.009
Drosophila melanogaster [15]	52	6	7	0.027	0.043	0.019
T-cell LGL leukemia signaling [9]	60	7	1	1.371	0.877	0.168

doi:10.1371/journal.pone.0145734.t001

solver-based attractor detection tool) [24] and ST (*i.e.,* the state-of-the-art partitioning-based attractor detection tool) [25]. It is important to mention that those methods can find not only the fixed points of BN, but also cyclic attractors of the network, which our method is not currently designed to do. There also exist other methods mainly focusing on finding fixed points of the network, but they do not provide their source code or any executable binaries [17, 29]. However, the reported timing of their methods grows exponentially with the number of nodes for BNs with $K = 3$. As we will see later, the timing of our method grows linearly for such networks, so it was not necessary to implement them to include in our benchmark. All the simulations were performed on a machine with Intel(R) Core(TM) i3-2120 CPU@3.30GHz 2-Core with 4GB memory running Ubuntu 12.04.

Results for Boolean networks models of real biological processes

We tested the proposed and other methods on seven Boolean network models of real biological processes: control of flower morphogenesis in the mammalian cell cycle regulation [10], fission yeast cell cycle regulation [14], budding yeast cell cycle regulation [13], T-helper cell differentiation [11], T-cell receptor signaling pathway analysis [12], Drosophila melanogaster segment polarity genes expression patterns prediction [15], and T-cell large granular lymphocyte (T-LGL) leukemia signaling [9].

The simulation results are summarized in Table 1. Columns 1, 2, 3 and 4 show the name of the model, the number of nodes (*i.e., n*), the maximum indegree (*i.e., K*) and the number of fixed points computed, respectively. In columns 5, 6 and 7, we show the runtime of BNS, ST, and PAD, respectively. As shown in Table 1, the models are all small, but have high *K*. The results indicate that the proposed tool performs better than BNS by 2.91 times, and ST by 3.27 times in average. Interestingly, BNS is faster than ST although ST partitions the given model to several blocks, and applies a SAT solver to each block while BNS applies a SAT solver without partitioning. A possible reason is that ST needs to compute strongly connected components (SCC) to partition the given BN based on those, and such overhead caused not to improve the performance of ST in the small networks. The partitioning method of the proposed algorithm is simpler than the one of ST, and also minimizes both *n* and *K* of each block. Therefore, the computational advantage gained by the proposed partition can dominate the overhead of the partitioning procedure, and so PAD outperforms both BNS and ST even in the small networks.

Results for random Boolean networks

The models in Table 1 consist of several tens of genes only, which is a typical size used in most published models today. It is known, however, that the number of genes involved in many processes of biological interest exceeds the size of these models by at least an order of magnitude [28]. The number of functionally relevant interactions between genes of these models, representing the links, is expected to be much more complex (*i.e.,* higher *K*) [25]. In order to evaluate how well our algorithm can scale up on larger and more complex examples, a large set of BN with different *K*s are randomly generated. We use the *BoolNet* package [41] in the *R* environment [42] to generate $N - K$ random BNs. To generate more biologically realistic random BNs, the parameters of *generateRandomNKNetwork* function are set to $K = 2$ to 5 and *topology* = *scale_free*, where *K* and *topology* represent parameters for the maximum indegree and the network structure of the BN, respectively. Note that many real biological networks have the scale-free property such that *K* of these models greatly exceeds the average indegree [34, 43, 44].

In the first simulation, we applied the algorithms to compute the median runtime per steady state for 3,600 randomly generated BNs of sizes between $n = 100$ and $n = 900$ nodes with $K = 2$ to $K = 5$. The timeout is set to 1000 seconds, and the results only include the data within the timeout as shown in Fig 4. Each dot is computed for 100 networks. As we can see in Fig 4B–

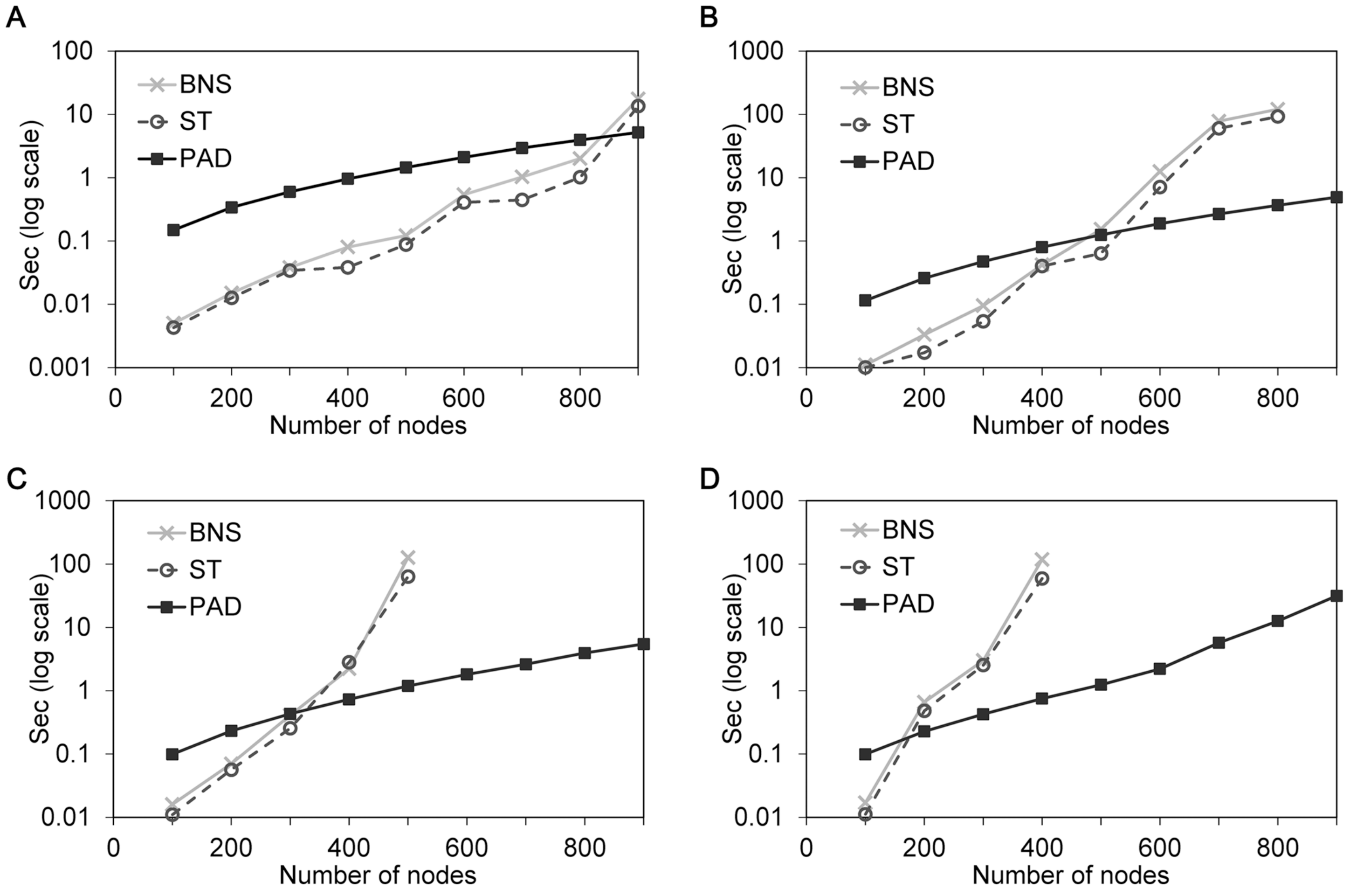

Fig 4. Runtime comparisions of algorithms. The simulation results for 3,600 randomly generated Boolean networks (BN) with maximum indegree *K*. Each dot is the median value computed for 100 networks. A: Median runtime of algorithms for BN with $K = 2$. B: Median runtime of algorithms for BN with $K = 3$. C: Median runtime of algorithms for BN with $K = 4$. D: Median runtime of algorithms for BN with $K = 5$.

doi:10.1371/journal.pone.0145734.g004

4D, the median runtime per steady state for BNS and ST grows exponentially with *n*. We can also see that all timings grow much faster for networks with increasing *K*. Interestingly, BNS performs almost the same as ST for networks with $K = 4$ or 5. We consider that the partitioning method implemented in ST focuses on reducing costs to find steady states in networks with relatively low *K*. The timing of our algorithm, however, grows linearly with *n* even for networks with high $K = 5$.

On the other hand, PAD performs slightly worse than other tools when networks have relatively low *K* and small *n*, as we can see in Fig 4. We consider that the results may be due to the fact that the proposed partitioning method is optimized to handle mainly large networks with high *K*. But, at the same time, more number of subnetworks can be constructed, which may induce an overhead during composing local steady states of them. Meanwhile, a SAT solver itself shows good scalability for networks with up to $n = 800$ nodes when $K = 2$ as shown in Fig 4A. Thus, we account for that the proposed fine-grained partition is over-optimized for such BNs and caused the overhead in the procedure of composition. The proposed partitioning method, however, enables each subnetwork to maintain small size and simple structure even for large BNs with high *K*s. It is worth noting that the cost to compute steady states of other methods grows exponentially with *n* and *K*. This is because their units of partitioning (*i.e.,* the given BN itself in BNS and the strongly connected component in ST) are still too large and complex for a SAT solver to compute steady states efficiently. Compared to those timings, the aforementioned overhead of our method is negligible. Therefore, the proposed fine-grained partition enables our method to be much more scalable than others.

In the second simulation, we compared the presented algorithm to other algorithms in terms of the number of successful terminations within 60 sec. We used the same random BNs as the ones generated in the first simulation. The results are shown in Fig 5. Each dot is computed for 100 networks. As we can see, for BNS the number of successful termination rapidly approaches 0 for $n > 200$. In comparison to BNS, the number of successful termination of ST approaches 0 slightly slower than BNS, but it rapidly reaches 0 when *K* increases. On the other hand, PAD is able to handle 40 percent of networks even for $n = 900$ and $K = 5$. The results show that increasing *K* has a dramatic effect on the sizes of feasible networks that can be studied. Especially for BNS and ST, further increasing *K* will have a much more dramatic effect. We consider that the larger (*i.e.,* $n \geq 1000$) and more complex (*i.e.,* $K \geq 5$) BNs would be the trend for real biological process models [25, 45, 46]. We claim that PAD would be more useful in those BNs because of the scalability although our method is not currently equipped to find cyclic attractors.

Discussion

Finding steady states is one of the key problems in the analysis of Boolean network models of biological processes. In this manuscript, we presented an efficient algorithm for the determination of steady states based on partitioning and a SAT solver. Our method ensures each block of the partition to be the minimal both in size (*i.e., n*) and the maximum indegree (*i.e., K*) such that maximizes the performance of a SAT solver to identify local steady states. The correctness of our proposed algorithm is also formally proved. We have used two types of networks for benchmark: published models for real biological processes, and randomly generated scale-free networks. Our simulation results show that the inherent overhead (*i.e.,* partitioning and composing costs) of our method is compensated by the efficiency of the provided partition, and thus our implementation tool, PAD, shows good scalability even on large random networks with high *K*s.

In 2007, Naldi *et al.* [20] proposed a steady-state analysis algorithm by using decision diagrams (DDs) each of which is constructed based on a Boolean function associated with a node.

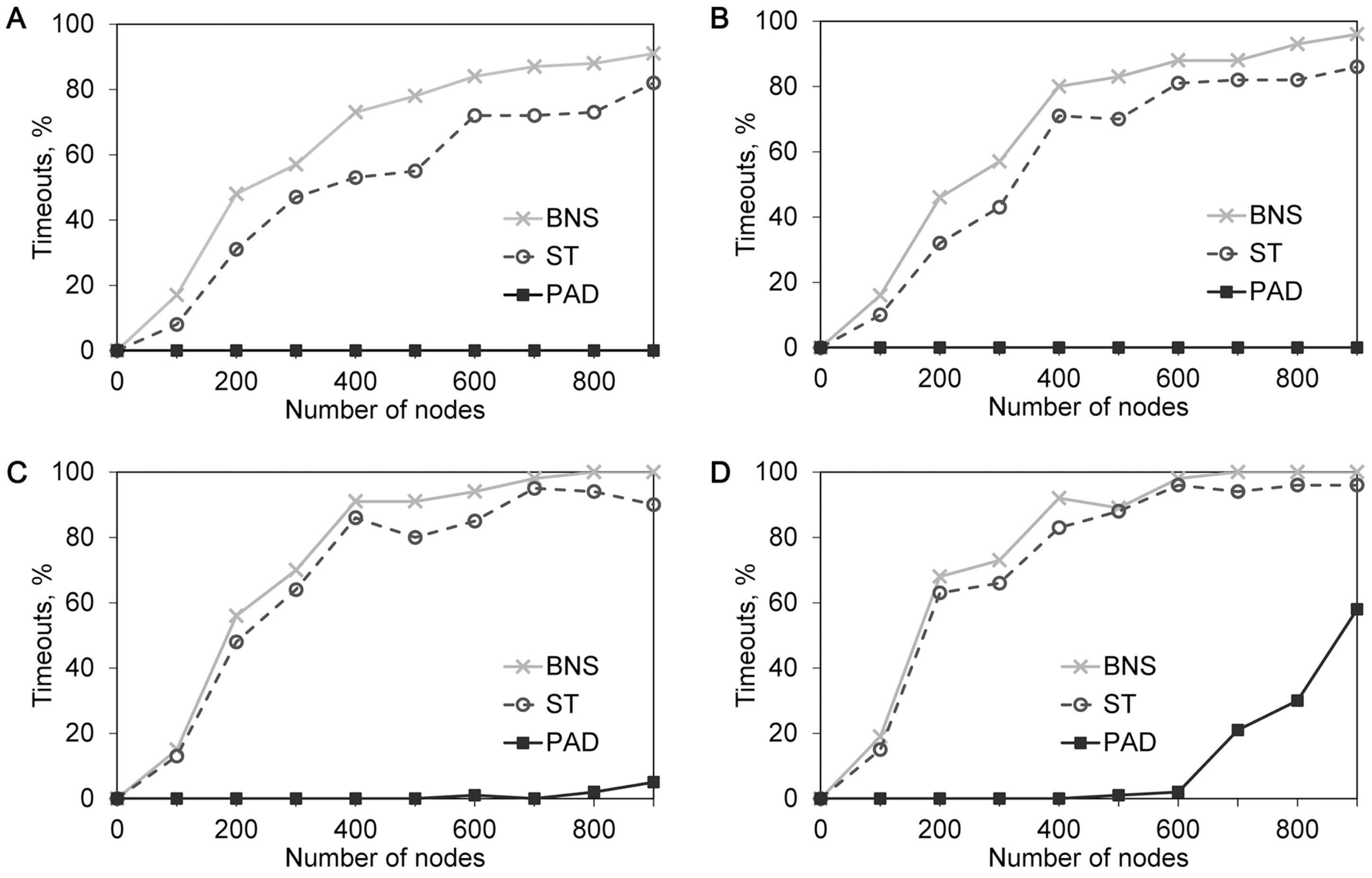

Fig 5. Scalability comparisions of algorithms. The simulation results for 3,600 randomly generated Boolean networks (BN) with maximum indegree *K*. Each dot is the percentage of timeouts computed for 100 networks. A: Timeout percentage of algorithms for BNs with *K* = 2. B: Timeout percentage of algorithms for BNs with *K* = 3. C: Timeout percentage of algorithms for BNs with *K* = 4. D: Timeout percentage of algorithms for BNs with *K* = 5.

doi:10.1371/journal.pone.0145734.g005

It is worth noting that each DD similarly corresponds to each subnetwork of our approach. Each DD (*i.e.,* local DD) is then composed in a pairwise manner, and the steady states of the given network are determined by checking the finally composed DD (*i.e.,* global DD). In the course of the composition, redundant intermediate composed local DDs can be temporarily generated, which will be discarded later on. Depending on the order of DD compositions, such redundant DDs can be significantly increased, thus making the intermediate combined DD become too memory-consuming. Such redundancy may be substantially reduced when composed in a good order, but they do not provide how to determine such an order. Thus, this limits the Naldi *et al.*'s algorithm [20] to deal with only small (*i.e.,* about a hundred nodes [47, 48]) Boolean networks like other DD-based algorithms [18–21]. Our partitioning method is similar to that of Naldi *et al.*'s study [20]. However, our steady-state detection algorithm is different from the Naldi *et al.*'s in two aspects as follows [20]: (1) instead of using DD, we use a SAT solver to analyze each subnetwork. (2) in contrast to Naldi *et al.*'s study [20], we provide an efficient composition order that reduces redundancy in intermediate steady states to be generated during composing local steady states. Thus, it results in a significantly smaller size of combined local steady states than that of its corresponding combined local DD as the composition steps continue. The above differences enable our composition method to have much smaller composition cost than the Naldi *et al.*'s [20]. Hence, this contributes to the enhanced scalability of our steady-state identification method although partitioning methods are similar. Here it should be noted that Naldi *et al.* provide a user-friendly GUI tool, GINsim, and it can handle multiple-valued networks too [20].

Zhao *et al.* [33] and Guo *et al.* [25] also exploited network partitioning methods to identify steady states of Boolean networks efficiently. As shown in the simulation results, however, the execution time of their methods grows exponentially with n and K. Specifically, for scale-free random networks with $K = 5$, they seemed not to terminate at all as n increases. The reason is that the size of the largest SCC is too large to be analyzed within a reasonable timing for such networks with relatively high K. In contrast to the results of Zhao *et al.* [33] and Guo *et al.* [25], PAD is scalable for the scale-free random Boolean networks up to several hundreds of nodes by favor of the MEB-based partition. The published Boolean network models analyzed in this manuscript have high K, but consist of several tens on nodes only. However, the size of published models is growing, and the largest SCC of such models connects the vast majority of nodes [35–39]. It is also reported that the maximum indegree of published models is orders of magnitude higher than the average indegree [34]. Thus, we believe that the demonstrated scalability for finding steady states will be a key functionality in any systems biology toolkit as more published large and scale-free networks become available.

Alongside the partitioning methods, network reduction methods are another direction of research in steady-state analysis [49–55]. For example, Zanudo and Albert [54] recently proposed a method that uses network motifs (*i.e.,* subgraphs) to reduce the given networks. Then, the reduced networks are analyzed to find not only the steady states, but also cyclic attractors. By means of the network reduction method, they achieved good scalability up to 200-node BNs with indegree two. But, compared to our method, such network reduction based algorithms themselves are not scalable enough to handle large networks. However, it is worth noting that network reduction methods are beneficial when applied orthogonally to the existing steady-state identification methods to improve them. As an example, in 2014, Veliz-Cuba *et al.* [55] extended a process algebra based steady state finding algorithm by incorporating a network reduction. Once the reduced network is constructed, they can efficiently compute its steady states by applying the computational algebra technique. To take a lesson from them [55], we will consider to apply network reduction methods orthogonally to our algorithm in the future. We believe that the hybrid approach will give us a better chance to tackle challenges in identifying steady states, which is still unsolved in general. Another direction of our future work is to extend our method to determine all attractors including cyclic attractors.

Supporting Information

S1 Text. Correctness proof. The file presents the correctness proof of our steady-state detection algorithm. Open with your favorite pdf reader, *e.g.,* Adobe Reader.
(PDF)

S2 Text. Optimality proof. The file presents the optimality proof of our partitioning algorithm. Open with your favorite pdf reader, *e.g.,* Adobe Reader.
(PDF)

S3 Text. Analysis of the average time complexity of our composition algorithm. The file presents the analysis of the average time complexity of our composition algorithm. Open with your favorite pdf reader, *e.g.,* Adobe Reader.
(PDF)

Acknowledgments

This work was supported in part by IITP (B0101-15-0557) and NRF funded by the Korea Government, the Ministry of Science, ICT & Future Planning (2013M3A9A7046303, 2014R1A2A1A10052404 and 2015M3A9A7067220).

Author Contributions

Conceived and designed the experiments: CH IS. Performed the experiments: CH JH. Analyzed the data: CH IS. Contributed reagents/materials/analysis tools: CH. Wrote the paper: CH IS JH KHC.

References

1. de Jong H. Modeling and Simulation of Genetic Regulatory Systems: A Literature Review. Journal of Computational Biology. 2002; 9(1):67–103. Available from: http://dx.doi.org/10.1089/10665270252833208 doi: 10.1089/10665270252833208 PMID: 11911796
2. Glass L, Kauffman SA. The logical analysis of continuous, non-linear biochemical control networks. Journal of Theoretical Biology. 1973; 39(1):103–129. Available from: http://www.sciencedirect.com/science/article/pii/0022519373902087 doi: 10.1016/0022-5193(73)90208-7 PMID: 4741704
3. Kauffman SA. Metabolic stabiligy and epigenesis in randomly constructed genetic nets. Journal of Theoretical Biology. 1969 03; 22:437–467. doi: 10.1016/0022-5193(69)90015-0 PMID: 5803332
4. Kauffman SA. Homeostasis and Differentiation in Random Genetic Control Networks. Nature. 1969; 224:177–178. doi: 10.1038/224177a0 PMID: 5343519
5. Kauffman SA. The origins of order: self organization and selection in evolution. New York: Oxford university press; 1993. Available from: http://opac.inria.fr/record = b1077782
6. Somogyi R, Sniegoski C. Modeling the complexity of genetic networks: Understanding multigenic and pleiotropic regulation. Complexity. 1996; 1:45–63. Available from: http://citeseerx.ist.psu.edu/showciting;jsessionid=5B8E1057A4189B63F0AE6223D3575F4B?cid=44061 doi: 10.1002/cplx.6130010612
7. Helikar T, Konvalina J, Heidel J, Rogers JA. Emergent decision-making in biological signal transduction networks. Proceedings of the National Academy of Sciences. 2008; 105(6):1913–1918. Available from: http://www.pnas.org/content/105/6/1913.abstract doi: 10.1073/pnas.0705088105
8. Choi M, Shi J, Jung SH, Chen X, Cho KH. Attractor Landscape Analysis Reveals Feedback Loops in the p53 Network That Control the Cellular Response to DNA Damage. Science Signaling. 2012; 5 (251):ra83–ra83. doi: 10.1126/scisignal.2003363 PMID: 23169817
9. Saadatpour A, Wang RS, Liao A, Liu X, Loughran TP, Albert I, et al. Dynamical and Structural Analysis of a T Cell Survival Network Identifies Novel Candidate Therapeutic Targets for Large Granular Lymphocyte Leukemia. PLoS Comput Biol. 2011 11; 7(11):e1002267. Available from: http://dx.doi.org/10.1371%2Fjournal.pcbi.1002267 doi: 10.1371/journal.pcbi.1002267 PMID: 22102804
10. Fauré A, Naldi A, Chaouiya C, Thieffry D. Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. Bioinformatics. 2006; 22(14):e124–e131. Available from: http://bioinformatics.oxfordjournals.org/content/22/14/e124.abstract doi: 10.1093/bioinformatics/btl210 PMID: 16873462
11. Mendoza L, Xenarios I. A method for the generation of standardized qualitative dynamical systems of regulatory networks. Theoretical Biology and Medical Modelling. 2006; 3(1):13. Available from: http://www.tbiomed.com/content/3/1/13 doi: 10.1186/1742-4682-3-13 PMID: 16542429
12. Klamt S, Saez-Rodriguez J, Lindquist J, Simeoni L, Gilles E. A methodology for the structural and functional analysis of signaling and regulatory networks. BMC Bioinformatics. 2006; 7(1):56. Available from: http://www.biomedcentral.com/1471-2105/7/56 doi: 10.1186/1471-2105-7-56 PMID: 16464248
13. Li F, Long T, Lu Y, Ouyang Q, Tang C. The yeast cell-cycle network is robustly designed. Proceedings of the National Academy of Sciences of the United States of America. 2004; 101(14):4781–4786. Available from: http://www.pnas.org/content/101/14/4781.abstract doi: 10.1073/pnas.0305937101 PMID: 15037758
14. Davidich MI, Bornholdt S. Boolean Network Model Predicts Cell Cycle Sequence of Fission Yeast. PLoS ONE. 2008 02; 3(2):e1672. Available from: http://dx.plos.org/10.1371%2Fjournal.pone.0001672 doi: 10.1371/journal.pone.0001672 PMID: 18301750
15. Albert R, Othmer HG. The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in Drosophila melanogaster. Journal of Theoretical Biology. 2003; 223(1):1–18.

Available from: http://www.sciencedirect.com/science/article/pii/S0022519303000353 doi: 10.1016/S0022-5193(03)00035-3 PMID: 12782112

16. Akutsu T, Kuhara S, Maruyama O, Miyano S. A System for Identifying Genetic Networks from Gene Expression Patterns Produced by Gene Disruptions and Overexpressions; 1998.

17. Devloo V, Hansen P, Labbé M. Identification of all steady states in large networks by logical analysis. Bulletin of Mathematical Biology. 2003; 65(6):1025–1051. Available from: http://dx.doi.org/10.1016/S0092-8240%2803%2900061-2 doi: 10.1016/S0092-8240(03)00061-2 PMID: 14607287

18. Dubrova E, Teslenko M, Martinelli A. Kauffman networks: Analysis and applications. In: in Proceedings of the IEEE/ACM International Conference on Computer-Aided Design; 2005.

19. Garg A, Xenarios I, Mendoza L, DeMicheli G. An Efficient Method for Dynamic Analysis of Gene Regulatory Networks and inÂ silico Gene Perturbation Experiments. In: Speed T, Huang H, editors. Research in Computational Molecular Biology. vol. 4453 of Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin / Heidelberg; 2007. p. 62–76. Available from: http://dx.doi.org/10.1007/978-3-540-71681-5_5

20. Naldi A, Thieffry D, Chaouiya C. Decision Diagrams for the Representation and Analysis of Logical Models of Genetic Networks. In: Calder M, Gilmore S, editors. Computational Methods in Systems Biology. vol. 4695 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2007. p. 233–247. Available from: http://dx.doi.org/10.1007/978-3-540-75140-3_16

21. Zheng D, Yang G, Li X, Wang Z, Liu F, He L. An Efficient Algorithm for Computing Attractors of Synchronous And Asynchronous Boolean Networks. PLoS ONE. 2013 04; 8(4):e60593. Available from: http://dx.doi.org/10.1371%2Fjournal.pone.0060593 doi: 10.1371/journal.pone.0060593 PMID: 23585840

22. Tamura T, Akutsu T. An O(1.787 n)-Time Algorithm for Detecting a Singleton Attractor in a Boolean Network Consisting of AND/OR Nodes. In: Csuhaj-Varjú E, Ésik Z, editors. Fundamentals of Computation Theory. vol. 4639 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2007. p. 494–505. Available from: http://dx.doi.org/10.1007/978-3-540-74240-1_43

23. Akutsu T, Melkman AA, Tamura T, Yamamoto M. Determining a Singleton Attractor of a Boolean Network with Nested Canalyzing Functions. Journal of Computational Biology. 2011; 18(10):1275–1290. Available from: http://dx.doi.org/10.1089/cmb.2010.0281 doi: 10.1089/cmb.2010.0281 PMID: 21554129

24. Dubrova E, Teslenko M. A SAT-Based Algorithm for Finding Attractors in Synchronous Boolean Networks. IEEE/ACM Transactions on Computational Biology and Bioinformatics. 2011; 8(5):1393–1399. doi: 10.1109/TCBB.2010.20 PMID: 21778527

25. Guo W, Yang G, Wu W, He L, Sun M. A Parallel Attractor Finding Algorithm Based on Boolean Satisfiability for Genetic Regulatory Networks. PLoS ONE. 2014 04; 9(4):e94258. Available from: http://dx.doi.org/10.1371%2Fjournal.pone.0094258 doi: 10.1371/journal.pone.0094258 PMID: 24718686

26. Zhang SQ, Hayashida M, Akutsu T, Ching WK, Ng MK. Algorithms for finding small attractors in boolean networks. EURASIP J Bioinformatics Syst Biol. 2007 Jan; 2007:4. Available from: http://dx.doi.org/10.1155/2007/20180

27. Di Cara A, Garg A, De Micheli G, Xenarios I, Mendoza L. Dynamic simulation of regulatory networks using SQUAD. BMC Bioinformatics. 2007; 8(1):462. Available from: http://www.biomedcentral.com/1471-2105/8/462 doi: 10.1186/1471-2105-8-462 PMID: 18039375

28. de Jong H, Page M. Search for Steady States of Piecewise-Linear Differential Equation Models of Genetic Regulatory Networks. IEEE/ACM Trans Comput Biology Bioinform. 2008; 5(2):208–222. Available from: http://doi.acm.org/10.1145/1371585.1371590 doi: 10.1109/TCBB.2007.70254

29. Akutsu T, Hayashida M, Tamura T. Integer programming-based methods for attractor detection and control of boolean networks. In: Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on; 2009. p. 5610–5617.

30. Akutsu T, Melkman AA, Tamura T. Singleton and 2-periodic attractors of sign-definite Boolean networks. Information Processing Letters. 2012; 112(1-2):35–38. Available from: http://www.sciencedirect.com/science/article/pii/S0020019011002675 doi: 10.1016/j.ipl.2011.10.002

31. Eén N, Sörensson N. An Extensible SAT-solver. In: Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers; 2003. p. 502–518. Available from: http://dx.doi.org/10.1007/978-3-540-24605-3_37

32. Schaefer TJ. The Complexity of Satisfiability Problems. In: Proceedings of the Tenth Annual ACM Symposium on Theory of Computing. STOC'78. New York NY, USA: ACM; 1978. p. 216–226. Available from: http://doi.acm.org/10.1145/800133.804350

33. Zhao Y, Kim J, Filippone M. Aggregation Algorithm Towards Large-Scale Boolean Network Analysis. Automatic Control, IEEE Transactions on. 2013 Aug; 58(8):1976–1985. doi: 10.1109/TAC.2013.2251819

34. Albert R. Network Inference, Analysis, and Modeling in Systems Biology. The Plant Cell. 2007; 19 (11):3327–3338. Available from: http://www.plantcell.org/content/19/11/3327.short doi: 10.1105/tpc.107.054700 PMID: 18055607

35. Giot L, Bader JS, Brouwer C, Chaudhuri A, Kuang B, Li Y, et al. A Protein Interaction Map of Drosophila melanogaster. Science. 2003; 302(5651):1727–1736. Available from: http://www.sciencemag.org/content/302/5651/1727.abstract doi: 10.1126/science.1090289 PMID: 14605208

36. Yook SH, Oltvai ZN, Barabási AL. Functional and topological characterization of protein interaction networks. PROTEOMICS. 2004; 4(4):928–942. Available from: http://dx.doi.org/10.1002/pmic.200300636 doi: 10.1002/pmic.200300636 PMID: 15048975

37. Geisler-Lee J, O'Toole N, Ammar R, Provart NJ, Millar AH, Geisler M. A Predicted Interactome for Arabidopsis. Plant Physiology. 2007; 145(2):317–329. Available from: http://www.plantphysiol.org/content/145/2/317.abstract doi: 10.1104/pp.107.103465 PMID: 17675552

38. Ma HW, Zeng AP. The connectivity structure, giant strong component and centrality of metabolic networks. Bioinformatics. 2003; 19(11):1423–1430. Available from: http://bioinformatics.oxfordjournals.org/content/19/11/1423.abstract doi: 10.1093/bioinformatics/btg177 PMID: 12874056

39. Ma'ayan A, Jenkins SL, Neves S, Hasseldine A, Grace E, Dubin-Thaler B, et al. Formation of Regulatory Patterns During Signal Propagation in a Mammalian Cellular Network. Science. 2005; 309 (5737):1078–1083. Available from: http://www.sciencemag.org/content/309/5737/1078.abstract doi: 10.1126/science.1108876 PMID: 16099987

40. Aldana M. Boolean dynamics of networks with scale-free topology. Physica D: Nonlinear Phenomena. 2003; 185(1):45–66. Available from: http://www.sciencedirect.com/science/article/pii/S016727890300174X doi: 10.1016/S0167-2789(03)00174-X

41. Müssel C, Hopfensitz M, Kestler HA. BoolNet—an R package for generation, reconstruction and analysis of Boolean networks. Bioinformatics. 2010; 26(10):1378–1380. doi: 10.1093/bioinformatics/btq124 PMID: 20378558

42. R Development Core Team. R: A Language and Environment for Statistical Computing. Vienna, Austria; 2008. ISBN 3-900051-07-0. Available from: http://www.R-project.org

43. Barabási AL, Albert R. Emergence of Scaling in Random Networks. Science. 1999; 286(5439):509–512. Available from: http://www.sciencemag.org/content/286/5439/509.abstract doi: 10.1126/science.286.5439.509 PMID: 10521342

44. Barabási AL, Oltvai ZN. Network biology: understanding the cell's functional organization. Nat Rev Genet. 2004 02; 5:101–113. Available from: http://dx.doi.org/10.1038/nrg1272 doi: 10.1038/nrg1272 PMID: 14735121

45. Patil A, Nakamura H. Disordered domains and high surface charge confer hubs with the ability to interact with multiple proteins in interaction networks. FEBS Letters. 2006; 580(8):2041–2045. Available from: http://www.sciencedirect.com/science/article/pii/S0014579306002961 doi: 10.1016/j.febslet.2006.03.003 PMID: 16542654

46. Han JDJ, Bertin N, Hao T, Goldberg DS, Berriz GF, Zhang LV, et al. Evidence for dynamically organized modularity in the yeast protein–protein interaction network. Nature. 2004; 430(6995):88–93. doi: 10.1038/nature02555 PMID: 15190252

47. Berntenis N, Ebeling M. Detection of attractors of large Boolean networks via exhaustive enumeration of appropriate subspaces of the state space. BMC Bioinformatics. 2013; 14(1):361. Available from: http://www.biomedcentral.com/1471-2105/14/361 doi: 10.1186/1471-2105-14-361 PMID: 24330355

48. Chatain Th, Haar S, Jezequel L, Paulevé L, Schwoon S. Characterization of Reachable Attractors Using Petri Net Unfoldings. In: Mendes P, editor. Proceedings of the 12th Conference on Computational Methods in System Biology (CMSB'14). vol. 8859 of Lecture Notes in Bioinformatics. Manchester, UK: Springer-Verlag; 2014. p. 129–142. Available from: http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/CHJPS-cmsb14.pdf

49. Bilke S, Sjunnesson F. Stability of the Kauffman model. Physical Review E. 2002 Jan; 65(1):016129. doi: 10.1103/PhysRevE.65.016129

50. Richardson KA. Simplifying Boolean Networks. Advances in Complex Systems. 2005; 08(04):365–381. Available from: http://www.worldscientific.com/doi/abs/10.1142/S0219525905000518 doi: 10.1142/S0219525905000518

51. Naldi A, Monteiro PT, Chaouiya C. Efficient Handling of Large Signalling-regulatory Networks by Focusing on Their Core Control. In: Proceedings of the 10th International Conference on Computational Methods in Systems Biology. CMSB'12. Berlin, Heidelberg: Springer-Verlag; 2012. p. 288–306. Available from: http://dx.doi.org/10.1007/978-3-642-33636-2_17

52. Veliz-Cuba A. Reduction of Boolean network models. Journal of Theoretical Biology. 2011; 289:167–172. Available from: http://www.sciencedirect.com/science/article/pii/S0022519311004450 doi: 10.1016/j.jtbi.2011.08.042 PMID: 21907211

53. Naldi A, Remy E, Thieffry D, Chaouiya C. Dynamically consistent reduction of logical regulatory graphs. Theoretical Computer Science. 2011; 412(21):2207–2218. Selected Papers from the 7th International Conference on Computational Methods in Systems Biology7th International Conference on Computational Methods in Systems Biology. Available from: http://www.sciencedirect.com/science/article/pii/S0304397510005839 doi: 10.1016/j.tcs.2010.10.021

54. Zañudo JGT, Albert R. An effective network reduction approach to find the dynamical repertoire of discrete dynamic networks. Chaos. 2013; 23(2):–. Available from: http://scitation.aip.org/content/aip/journal/chaos/23/2/10.1063/1.4809777 PMID: 23822509

55. Veliz-Cuba A, Aguilar B, Hinkelmann F, Laubenbacher R. Steady state analysis of Boolean molecular network models via model reduction and computational algebra. BMC Bioinformatics. 2014; 15(1):221. Available from: http://www.biomedcentral.com/1471-2105/15/221 doi: 10.1186/1471-2105-15-221 PMID: 24965213