Aggregation Algorithm towards Large-Scale Boolean Network Analysis

Yin Zhao, *Student Member, IEEE,* Jongrae Kim, *Member, IEEE,* and Maurizio Filippone, *Member, IEEE*

***Abstract*—The analysis of large-scale Boolean network dynamics is of great importance in understanding complex phenomena where systems are characterized by a large number of components. The computational cost to reveal the number of attractors and the period of each attractor increases exponentially as the number of nodes in the networks increases. This paper presents an efficient algorithm to find attractors for medium to large scale networks. This is achieved by analyzing subnetworks within the network in a way that allows to reveal the attractors of the full network with little computational cost. In particular, for each subnetwork modeled as a Boolean control network, the input-state cycles are found and they are composed to reveal the attractors of the full network. The proposed algorithm reduces the computational cost significantly, especially in finding attractors of short period, or any periods if the aggregation network is acyclic. Also, this paper shows that finding the best acyclic aggregation is equivalent to finding the strongly connected components of the network graph. Finally, the efficiency of the algorithm is demonstrated on two biological systems, namely a T-cell receptor network and an early flower development network.**

***Index Terms*—Boolean network, attractor, graph aggregation, acyclic aggregation**

I. Introduction

MANY mathematical models have been proposed in the literature to study biological networks, including genetic regulatory networks [1]. *Boolean networks* have attracted particular interest because of their simplicity and potential to model a large number of nodes in the network. Boolean networks were first proposed by Kauffman [2] to model genetic regulatory networks. In this framework, each gene is assumed to have two levels, either active (on, true or 1) or inactive (off, false or 0), and to be affected by several other genes and/or by itself. Besides the genetic regulatory networks, Boolean networks can be also used to model other biological interactions, such as biomolecular signaling pathways [3]. Although Boolean networks are not as detailed as continuous models given in the form of differential equations [4], they have been widely and successfully used in Systems Biology [5]–[7]. Unlike continuous models that usually involve several parameters, which are difficult or even impossible to be estimated or inferred, Boolean networks are parameter free models. One of the main problems in Boolean network modeling for biological or any physical dynamics is the identification of the update rules using observed data [8], [9]. Once these are obtained, similarly to the analysis of continuous systems, the dynamical properties of Boolean networks are analyzed by finding steady-states, attractors, size of the basin of attraction to each attractor, etc. Steady-states and attractors in particular are the most important characteristics when they are related to some specific physiological responses in biological networks [2], [10]–[13].

Y. Zhao is with the Key Laboratory of Systems & Control, Institute of Systems Science, Academy of Mathematics & Systems Science, Chinese Academy of Sciences, Beijing 100190, P. R. China. E-mail: zhaoyin@amss.ac.cn

J. Kim is with the Division of Biomedical Engineering, University of Glasgow, Glasgow G12 8QQ, UK. E-mail: Jongrae.Kim@glasgow.ac.uk

M. Filippone is with the School of Computing Science, University of Glasgow, Glasgow G12 8QQ, UK. E-mail: Maurizio.Filippone@glasgow.ac.uk

Manuscript received August 14, 2012; revised December 15, 2012 & January 25, 2013;

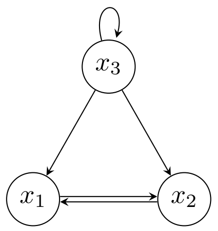


Fig. 1. An example of a network Graph comprising three nodes.

The interactions of genes or some biomolecular species are presented by logical functions as exemplified in Fig. 1 and its corresponding logical equations (1). The *network graph* in Fig. 1, shows an example of the interactions between the three nodes x_1, x_2, and x_3; a directed edge from node x_i to x_j means that the state (active or inactive) of x_j at time $t+1$ is affected by the state of x_i at time t.

As an example, consider the following updating rules among all possible functions corresponding to Fig. 1:

$$\begin{aligned} x_1(t+1) &= f_1[x_2(t), x_3(t)] = x_2(t) \wedge x_3(t), \\ x_2(t+1) &= f_2[x_1(t), x_3(t)] = x_1(t) \vee x_3(t), \\ x_3(t+1) &= f_3[x_3(t)] = \neg x_3(t), \end{aligned} \tag{1}$$

where $\wedge$, $\vee$, $\neg$ denote "AND", "OR", and "NOT" respectively.

The nodes in a network graph having no in-degrees, i.e., nodes that are not affected by others and/or themselves, can be interpreted as input nodes, while the nodes having no out-degrees can be interpreted as output nodes. Boolean networks with input nodes are called *Boolean control networks* and they form the building blocks that will be used in this paper to reveal attractors of larger networks. For example, considering only x_1 and x_2 and ignoring the update rule for x_3 in (1), x_1 and x_2 in Fig. 1 can be defined as a Boolean control network, where the dynamics of x_1 and x_2 are given by the first two equations in (1) and $u(t) = x_3(t)$ is the input.

Boolean networks with n nodes have 2^n number of possible states and it is proved that the computational complexity of

Boolean network related problems of interest is NP-hard [14]. Even for small Boolean networks, e.g. n around 50 or 100, it is almost impossible to find all attractors, in general. A lot of effort has been devoted to the solution of this issue, at least partially. One way to find attractors is to choose some initial states wisely and simulate the dynamics for each initial condition [15]; however, the global dynamics can be hardly revealed by this method. In [11], a probabilistic method is developed by choosing initial states randomly to find a certain percentage of steady states with a given confidence level. In our previous work [16], Boolean networks are divided into several groups and the input-output structure of each group approximates the global dynamics. However, only part of the nodes can be approximated and some information is lost.

The main motivation of the idea to use network aggregation in [16] is from [17], where the web is partitioned to reduce the computational cost in calculating page rank. Some aggregation methods are also discussed in [18], [19] and the references therein. In this work, we build upon the idea of aggregation of Boolean networks into several subnetworks, but instead of the approximation, the structure of the attractors of Boolean networks is accurately recovered by the composition of the input-state cycles of subnetworks.

The paper is organized as follows: section 2 describes the aggregation of Boolean networks; section 3 provides the method of revealing attractors of Boolean networks by composing the input-state cycles of subnetworks, and analyzes its computational complexity; section 4 presents a special aggregation structure called acyclic aggregation, for which the suggested algorithm is particularly efficient; the efficiency of the algorithm is demonstrated in section 5 using T-cell receptor network and an early flower development network; finally, section 6 presents the conclusion.

II. Aggregation of Boolean Networks

Consider the following Boolean network:

$$\begin{aligned} x_1(t+1) &= f_1[x_1(t), x_2(t), \ldots, x_n(t)], \\ x_2(t+1) &= f_2[x_1(t), x_2(t), \ldots, x_n(t)], \\ &\cdots, \\ x_n(t+1) &= f_n[x_1(t), x_2(t), \ldots, x_n(t)], \end{aligned} \tag{2}$$

where $x_i(t)$ for $i = 1, 2, \ldots, n$ denotes the state of node x_i at time t that can be either 0 for inactive or 1 for active. The nodes can be partitioned into s-number of blocks as follows:

$$\mathcal{X} = \{x_1, x_2, \ldots, x_n\} = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \ldots \cup \mathcal{X}_s,$$

where $\mathcal{X}_i$ is a proper subset of $\mathcal{X}$, $\mathcal{X}_i \cap \mathcal{X}_j$ is empty for $i \neq j$, $\mathcal{X}_i = \{x_{i1}, x_{i2}, \ldots, x_{in_i}\}$, n_i is the number of nodes in the i-th block, and x_{ij}, the j-th node in the i-th block, is equal to x_k for a $k \in \{1, 2, \ldots, n\}$. We call this partition an *aggregation* of the Boolean network.

Each block $\mathcal{X}_i$ has incoming edges from outside of the block and some outgoing edges to the outside. The source nodes of these edges can be interpreted as inputs and outputs for each block. Denote the set of inputs and outputs of the block $\mathcal{X}_i$ as $\mathcal{U}_i = \{u_{i1}, u_{i2}, \ldots, u_{im_i}\}$ and $\mathcal{Y}_i = \{y_{i1}, y_{i2}, \ldots, y_{ip_i}\}$, respectively, and the set of all source nodes, whose edges cut by the partition, as $\mathcal{C} = \{x_{c_1}, x_{c_2}, \ldots, x_{c_p}\}$. Note that $\mathcal{U}_i$ and/or $\mathcal{Y}_i$ could be empty set, i.e., there are no input and/or output to and from the i-th block.

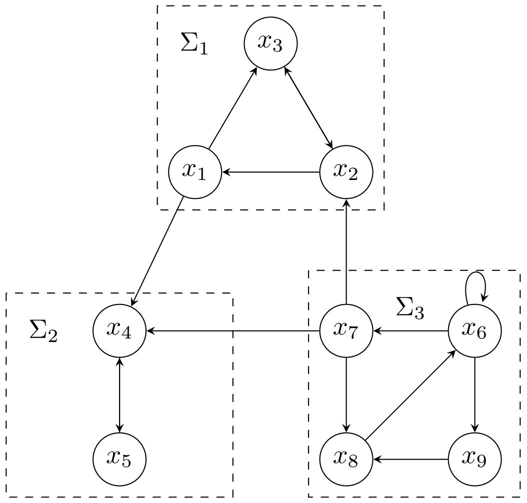


Fig. 2. An example of aggregation of a network comprising nine nodes into three Boolean control networks.

Remark 2.1:

1) y_{iq} in $\mathcal{Y}_i$ is a node in $\mathcal{X}_i$ and u_{il} in $\mathcal{U}_i$ is a node in another block, i.e.,
$$\mathcal{Y}_i \subset \mathcal{X}_i \subset \mathcal{X} \text{ and } \mathcal{U}_i \cap \mathcal{X}_i = \emptyset.$$
2) Each $x_{c_j} \in \mathcal{C}$ belongs to only one block (the output of a specific block), but could be the input of several other blocks. Hence,
$$\begin{aligned} \mathcal{C} &= \bigcup_{i=1}^{s} \mathcal{Y}_i = \bigcup_{i=1}^{s} \mathcal{U}_i, \\ \mathcal{Y}_i &\cap \mathcal{Y}_j = \emptyset, i \neq j, \\ p &= \sum_{i=1}^{s} p_i \leq \sum_{i=1}^{s} m_i. \end{aligned}$$

Then, the subnetwork Σ_i, with nodes in $\mathcal{X}_i$ and inputs in $\mathcal{U}_i$, is a Boolean control network given by

$$\begin{aligned} \Sigma_i: \quad x_{ij}(t+1) = f_{ij}[&x_{i1}(t), x_{i2}(t), \ldots, x_{in_i}(t), \\ &u_{i1}(t), u_{i2}(t), \ldots, u_{im_i}(t)], \end{aligned} \tag{3}$$

for $i = 1, 2, \ldots, s$ and $j = 1, 2, \ldots, n_i$.

Example 2.2: Consider a Boolean network example in Fig. 2. Assume its dynamics is described as

$$\begin{cases} x_1(t+1) = x_2(t) \\ x_2(t+1) = x_3(t) \wedge x_7(t) \\ x_3(t+1) = x_1(t) \leftrightarrow x_2(t) \\ x_4(t+1) = (x_1(t) \vee x_5(t)) \rightarrow x_7(t) \\ x_5(t+1) = \neg x_4(t) \\ x_6(t+1) = x_6(t) \bar{\vee} x_8(t) \\ x_7(t+1) = x_6(t) \\ x_8(t+1) = x_7(t) \vee x_9(t) \\ x_9(t+1) = \neg x_6(t), \end{cases} \tag{4}$$

where $\leftrightarrow$, $\rightarrow$, and $\bar{\vee}$ denote "EQUIVALENCE", "IMPLICATION", and "EXCLUSIVE-OR" operations respectively. Consider the aggregation into 3 blocks as shown in Fig 2,

$$\{x_1, x_2, x_3\} \in \mathcal{X}_1,\ \{x_4, x_5\} \in \mathcal{X}_2,\ \{x_6, x_7, x_8, x_9\} \in \mathcal{X}_3.$$

Now the inputs and outputs of each subsystem are

$$\begin{aligned}&\mathcal{U}_1 = \{u_{11} = x_7\},\ \mathcal{U}_2 = \{u_{21} = x_1, u_{22} = x_7\},\ \mathcal{U}_3 = \emptyset,\\ &\mathcal{Y}_1 = \{y_{11} = x_1\},\ \mathcal{Y}_2 = \emptyset,\ \mathcal{Y}_3 = \{y_{31} = x_7\},\\ &\mathcal{C} = \{x_{c_1} = x_1, x_{c_2} = x_7\}.\end{aligned}$$

Hence, there are three subnetworks

$$\Sigma_1 : \begin{cases} x_1(t+1) = x_2(t) \\ x_2(t+1) = x_3(t) \wedge u_{11}(t) \\ x_3(t+1) = x_1(t) \leftrightarrow x_2(t); \end{cases}$$

$$\Sigma_2 : \begin{cases} x_4(t+1) = (u_{21}(t) \vee x_5(t)) \rightarrow u_{22}(t) \\ x_5(t+1) = \neg x_4(t); \end{cases}$$

$$\Sigma_3 : \begin{cases} x_6(t+1) = x_6(t) \bar{\vee} x_8(t) \\ x_7(t+1) = x_6(t) \\ x_8(t+1) = x_7(t) \vee x_9(t) \\ x_9(t+1) = \neg x_6(t). \end{cases}$$

Note that the aggregation shown in Example 2.2 is not unique but there are many other different configurations. How to construct the best aggregation of the network to minimize the cost to find attractors will be discussed in section 4.

III. Revealing Attractors of the Whole Network

Finding attractors is one of the main problems in analyzing Boolean networks. Attractors are defined as follows:

Definition 3.1:

1) Consider the Boolean network given by (2). Its *State Transition Graph* is defined as a directed graph $\{\mathcal{D}^n, \mathcal{E}\}$, where $\mathcal{D} := \{0, 1\}$ and
$$\mathcal{E} = \{a \rightarrow b \,|a, b \in \mathcal{D}^n, b = f(a)\},$$
where $f = [f_1, f_2, \ldots, f_n]^T$.
2) A periodic path of the state transition graph is called an attractor of a Boolean network (2). An attractor with period 1 is also called a fixed point. Denote an attractor as
$$\{a_1 \rightarrow a_2 \rightarrow \ldots \rightarrow a_\ell \rightarrow a_1\},$$
where $a_i \in \mathcal{D}^n, i = 1, 2, \ldots, \ell$, and ℓ is the period or the length of the attractor. Attractors are also called cycles.

Example 3.2: Fig. 3 shows the state transition graph of Boolean network (1), and there are two attractors as follows:

$$\{(0\ 0\ 1) \rightarrow (0\ 1\ 0) \rightarrow (0\ 0\ 1)\},$$
$$\{(1\ 1\ 0) \rightarrow (0\ 1\ 1) \rightarrow (1\ 1\ 0)\}.$$

For a Boolean control network,

$$x_i(t+1) = f_i(x_1(t), \ldots, x_n(t), u_1(t), \ldots, u_m(t)), \quad (5)$$

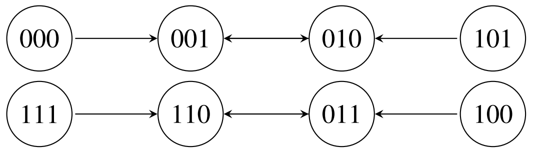


Fig. 3. State transition graph for (1), where states are ordered as $(x_1\, x_2\, x_3)$

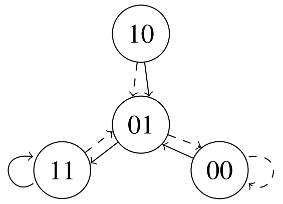


Fig. 4. State transition graph for Boolean control network (1), where states are ordered as $(x_1\, x_2)$ and the control $u = x_3$ is indicated by the solid arrow for $u = 1$, or by the dashed arrow for $u = 0$.

for $i = 1, 2, \ldots, n$, we have a similar definition of attractors as follows:

Definition 3.3:

1) Consider the Boolean control network given by (5). Its *State Transition Graph* is defined as a directed graph $\{\mathcal{D}^n, \mathcal{E}\}$, where
$$\mathcal{E} = \{a \rightarrow b \,|a, b \in \mathcal{D}^n, \exists u \in \mathcal{D}^m, b = f(a, u)\},$$
in which $f = [f_1, f_2, \ldots, f_n]^T$.
2) A periodic path of the state transition graph is called a cycle.
3) A periodic path with no repeated state in one period of the state transition graph is called an elementary cycle.

For a Boolean network, once the system enters a periodic path, it can not escape, thus we can call the periodic paths the attractors. But for Boolean control networks, periodic paths may be escapable by choosing different controls, thus we only call the periodic paths the cycles. A cycle of Boolean control networks may have repeated states in one period because of potentially multiple outgoing edges depending on inputs for each state, thus we need to make a distinction between cycles and elementary cycles. On the other hand, for Boolean networks, there is no repeated state in one period of the attractor as each state has only one outgoing edge as shown in Fig. 3, so all the attractors of Boolean networks are elementary.

Example 3.4: Fig. 4 shows the state transition graph, where x_3 is considered as the input in (1) (i.e., the edge from x_3 to itself is ignored). As there is one input, each state in the state transition graph has two outgoing edges depending on the different value of the input. For example, the cycle $\{(00) \rightarrow (00) \rightarrow (01) \rightarrow (00)\}$ in Fig. 4 is not an elementary cycle.

Input-state cycles of Boolean control networks need to be considered and they are defined by

Definition 3.5:

1) Consider the Boolean control network given by (5). Its *Input-State Transition Graph* is defined as a directed graph $\{\mathcal{D}^{n+m}, \mathcal{E}\}$, where
$$\mathcal{E} = \{(a, u) \rightarrow (b, u') \,|a, b \in \mathcal{D}^n, u, u' \in \mathcal{D}^m, b = f(a, u)\},$$
in which $f = [f_1, f_2, \ldots, f_n]^T$.

2) A periodic path of the input-state transition graph is called an input-state cycle.
3) A periodic path with no repeated state in one period of the input-state transition graph is called a elementary input-state cycle.

The definition implies that if there exits an u' in $\mathcal{D}^m$ such that an edge $(a,u) \to (b,u')$ exists, then edges $(a,u) \to (b,u'')$ for all u'' in $\mathcal{D}^m$ exist as well. For more details about input-state cycles of Boolean control networks, refer to [21], where the input-state cycles are simply called cycles.

It is easy to divide an input-state cycle with repeated states into several elementary input-state cycles, and conversely, it is also easy to combine elementary input-state cycles into cycles. Thus, we can use the algorithm in [20] to find all elementary input-state cycles, and then obtain input-state cycles by combination of these elementary ones; or alternatively, we can use the method in [21] to find input-state cycles directly. Note that, the number of elementary input-state cycles is much less than the number of input-state cycles. In fact, the number of input-state cycles may be infinite if the length is not limited. However, later we will see that the input-state cycles longer than 2^n are meaningless.

The main problem is now how to reveal an attractor of the whole Boolean network (2) from the input-state cycles of the subnetworks (3). To this end, denote $\mathcal{A}_i$ as the set of input-state cycles in Σ_i and define $\pi_i(\cdot,j) : \mathcal{D}^{n_i+m_i} \to \mathcal{D}$ as the projection of a state a in $\mathcal{D}^{n_i+m_i}$ onto the state of x_j, where $x_j \in \mathcal{X}_i \cup \mathcal{U}_i$. In addition, define the projection $\Pi_i(\cdot,j)$ of an input-state cycle $A \in \mathcal{A}_i$ onto the periodic trajectory of x_j, where $x_j \in \mathcal{X}_i \cup \mathcal{U}_i$, as follows: for a length-ℓ input-state cycle,

$$A = \{a_1 \to a_2 \to \cdots a_\ell \to a_1\}.$$

The projection, $\Pi_i(,j)$, is given by

$$\Pi_i(A,j) := \{\pi_i(a_1,j) \to \pi_i(a_2,j) \to \cdots \to pi_i(a_\ell,j) \to \pi_i(a_1,j)\}.$$

Note that the period of $\Pi_i(A,j)$ would be a divisor of ℓ. For notational simplicity, $\Pi(\cdot,j)$ is used without indicating the domain of each projection if there is no ambiguity.

Then we define the composition of two input-state cycles from different subnetworks. Note that, hereafter, if a subnetwork has no input nodes, we use its attractors, or say cycles, instead of input-state cycles. Assume Σ_1 and Σ_2 are two subnetworks, and

$$\mathcal{X}_1 \cup \mathcal{U}_1 = \{x_1,\ldots,x_{k_1},x_{k_1+1},\ldots,x_{k_1+k_2}\},$$
$$\mathcal{X}_2 \cup \mathcal{U}_2 = \{x_1,\ldots,x_{k_1},x_{k_1+k_2+1},\ldots,x_k\},$$

where x_1, x_2, $\ldots$, x_{k_1} are the common components in two subnetworks. If there are two input-state cycles for Σ_1 and Σ_2 with length, ℓ_1 and ℓ_2, respectively, i.e.,

$$A_1 = \{a_1 \to a_2 \to \cdots \to a_{\ell_1} \to a_1\} \in \mathcal{A}_1,$$
$$A_2 = \{b_1 \to b_2 \to \cdots \to b_{\ell_2} \to b_1\} \in \mathcal{A}_2,$$

and $\Pi(A_1,j) = \Pi(A_2,j)$ for $j = 1,2,\ldots,k_1$, then the composition of A_1 and A_2, i.e. $A_1 \times A_2$, is defined as follows:

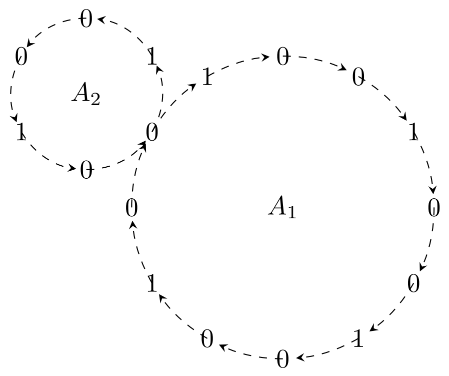


Fig. 5. An example of composition of two input-state cycles

1) Repeat A_1 and A_2 to length-ℓ periodic paths $\tilde{A}_1$ and $\tilde{A}_2$ where $\ell = \text{lcm}(\ell_1,\ell_2)$
$$\tilde{A}_1 = \{\tilde{a}_1 \to \tilde{a}_2 \to \cdots \to \tilde{a}_\ell \to \tilde{a}_1\},$$
$$\tilde{A}_2 = \{\tilde{b}_1 \to \tilde{b}_2 \to \cdots \to \tilde{b}_\ell \to \tilde{b}_1\},$$
where $\text{lcm}(\cdot,\cdot)$ is the least common multiple of the arguments.
2) Adjust the order of states in $\tilde{A}_1$ and $\tilde{A}_2$ by circular permutation in order to make $\Pi(\tilde{A}_1,j)$ and $\Pi(\tilde{A}_2,j)$ equal to each other for $j = 1,2,\ldots,k_1$.
3) $A_1 \times A_2$ is given by
$$A_1 \times A_2 = \Big\{ \Big[\tilde{a}_1, \pi(\tilde{b}_1,p), \pi(\tilde{b}_1,p+1),\ldots,\pi(\tilde{b}_1,k)\Big] \to \Big[\tilde{a}_2, \pi(\tilde{b}_2,p), \pi(\tilde{b}_2,p+1),\ldots,\pi(\tilde{b}_2,k)\Big] \to \cdots \to \Big[\tilde{a}_\ell, \pi(\tilde{b}_\ell,p), \pi(\tilde{b}_\ell,p+1),\ldots,\pi(\tilde{b}_\ell,k)\Big] \to \Big[\tilde{a}_1, \pi(\tilde{b}_1,p), \pi(\tilde{b}_1,p+1),\ldots,\pi(\tilde{b}_1,k)\Big]\Big\},$$
where $p = k_1 + k_2 + 1$. This is akin to the mechanism of two gears rotating together, as illustrated in Fig. 5.

The following example demonstrates the composition procedures:

Example 3.6: Consider two input-state cycles

$$A_1 = \{(0\ 0\ 1) \to (1\ 0\ 1) \to (0\ 1\ 1) \to (1\ 1\ 0) \to (0\ 0\ 1)\}$$
$$A_2 = \{(1\ 1) \to (0\ 0) \to (1\ 1)\},$$

where the states in A_1 are ordered as (x_1,x_2,x_3), and the states in A_2 are ordered as (x_1,x_4). Their projections onto x_1 are

$$\Pi(A_1,1) = \{0 \to 1 \to 0 \to 1 \to 0\} = \{0 \to 1 \to 0\}$$
$$\Pi(A_2,1) = \{1 \to 0 \to 1\},$$

thus, $\Pi(A_1,1) = \Pi(A_2,1)$. We can compose them.

1) Repeat A_1 and A_2 to length-4 periodic paths
$$\tilde{A}_1 = \{(0\ 0\ 1) \to (1\ 0\ 1) \to (0\ 1\ 1) \to (1\ 1\ 0) \to (0\ 0\ 1)\}$$
$$\tilde{A}_2 = \{(1\ 1) \to (0\ 0) \to (1\ 1) \to (0\ 0) \to (1\ 1)\}.$$
2) The projections of $\tilde{A}_1$ and $\tilde{A}_2$ onto x_1 are
$$\Pi(\tilde{A}_1,1) = \{0 \to 1 \to 0 \to 1 \to 0\} = \{0 \to 1 \to 0\}$$
$$\Pi(\tilde{A}_2,1) = \{1 \to 0 \to 1 \to 0 \to 1\} = \{1 \to 0 \to 1\}.$$

To make them equal to each other, reorder of the states in $\tilde{A}_2$ as

$$\tilde{A}_2 = \{(0\ 0) \to (1\ 1) \to (0\ 0) \to (1\ 1) \to (0\ 0)\}.$$

3) The composition is given by

$$\begin{aligned} A_1 \times A_2 =& \{(0\ 0\ 1\ \pi[(0,0),4]) \to (1\ 0\ 1\ \pi[(1,1),4]) \\ & \to (0\ 1\ 1\ \pi[(0,0),4]) \to (1\ 1\ 0\ \pi[(1,1),4]) \\ & \to (0\ 0\ 1\ \pi[(0,0),4])\} \\ =& \{(0\ 0\ 1\ 0) \to (1\ 0\ 1\ 1) \to (0\ 1\ 1\ 0) \\ & \to (1\ 1\ 0\ 1) \to (0\ 0\ 1\ 0)\}. \end{aligned}$$

Finally, we are ready to present an algorithm to recover the attractors of the whole network from its subnetworks. It is assumed that Boolean network graphs considered are at least weakly connected, i.e., there is always a path between any two nodes if the direction of the edges is ignored. This excludes networks with isolated multiple groups, where each isolated group can be analyzed one by one using the proposed algorithm in the following, if needed.

Algorithm 3.7: The attractors of the Boolean network (2) can be obtained by applying the following steps:

1) Partition the network graph to s-number of blocks $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_s\}$, and reorder them to $\{\mathcal{X}_{i_1}, \mathcal{X}_{i_2}, \ldots, \mathcal{X}_{i_s}\}$ such that the corresponding $\mathcal{X}_{i_\alpha}$ and $\mathcal{U}_{i_\alpha}$ for $2 \le \alpha \le s$ satisfy

$$\left(\bigcup_{\beta=1}^{\alpha-1} \mathcal{X}_{i_\beta} \cup \mathcal{U}_{i_\beta} \right) \cap (\mathcal{X}_{i_\alpha} \cup \mathcal{U}_{i_\alpha}) \neq \emptyset, \qquad (6)$$

i.e., each block is connected to the union of all blocks in front in the order by at least one edge.

2) For each subnetwork Σ_i, find all elementary input-state cycles, and then combine them to obtain all input-state cycles of Σ_i with the length of period less than or equal to 2^n. Denote the set of all input-state cycles as $\mathcal{A}_i$, for $i = 1, 2, \ldots, s$.

3) Find an attractor of Boolean network (2) by composing the input-state cycles of subnetworks as follows:

$$(((A_{i_1} \times A_{i_2}) \times A_{i_3}) \times \cdots) \times A_{i_s}, \qquad (7)$$

where A_{i_α} belongs to $\mathcal{A}_{i_\alpha}$, and $\{A_{i_\alpha} | \alpha = 1, 2, \ldots, s\}$ must satisfy the following for all x_{c_k} in $\mathcal{C}$ and α, β in $\{1, 2, \ldots, s\}$:

$$\Pi(A_\alpha, c_k) = \Pi(A_\beta, c_k) \qquad (8)$$

whenever x_{c_k} is an element of $(\mathcal{X}_\alpha \cup \mathcal{U}_\alpha) \cap (\mathcal{X}_\beta \cup \mathcal{U}_\beta)$.

Theorem 3.8: Consider the Boolean network (2), partitioned to subnetworks (3), where its network graph is assumed to be weakly connected, i.e. no isolated nodes in the network graph. The composition given by (7) is an attractor of Boolean network (2), and all attractors of the Boolean network (2) can be recovered using the above algorithm.

Proof. Firstly, as the network graph is weakly connected, for any aggregation, we can always find two subnetworks Σ_{i_1} and Σ_{i_2} such that $(\mathcal{X}_{i_1} \cup \mathcal{U}_{i_1}) \cap (\mathcal{X}_{i_2} \cup \mathcal{U}_{i_2})$ is non-empty, i.e., Σ_{i_1} and Σ_{i_2} satisfying the condition (6) for $\alpha = 2$. Then, we can find further Σ_{i_3} such that (6) holds for $\alpha = 3$ because of the weak connectivity of the network. These procedures, dividing and ordering the subnetworks to the index, $\{i_1, i_2, \ldots, i_\alpha\}$, such that (6) holds, are repeated till $\alpha = s$. Each edge of an input-state cycle A_{i_k} satisfies the dynamics of Σ_{i_k} in (3). (8) ensures that the states of overlapping nodes of the input-state cycles in (7) are equal to each other. Therefore, the edges of the composition (7), satisfy the overall dynamics (2), and (7) is an attractor of the Boolean network (2).

Conversely, for any attractor A of the Boolean network (2), by projecting it onto the nodes in $\mathcal{X}_{i_k} \cup \mathcal{U}_{i_k}$, for $k = 1, 2, \ldots, s$, an input-state cycles A_{i_k}, is obtained and (8) holds, and the composition of A_{i_k}'s is equal to A. Hence, all the attractors of the Boolean network (2), can be revealed by (7). □

As the attractors of the Boolean network (2), cannot be longer than 2^n, their projections onto each subnetwork are shorter than or equal to 2^n as well. Hence, finding input-state cycles, whose length is longer than 2^n, is not necessary.

Complexity Analysis: The proposed algorithm has four main parts as follows:

P1. Aggregation of the Boolean network
P2. Finding all elementary input-state cycles for each subnetwork
P3. Combine the elementary input-state cycles to input-state cycles
P4. Compose the input-state cycles to attractors of the whole network.

Comparing to P2, the complexity of P1 is negligible as the computational cost increases polynomially with the size of the networks. For example, to calculate the eigenvectors of the Laplacian matrix of network graph, whose size is $n \times n$, the complexity is $O(n^3)$ if we use a spectral partitioning method such as min-cut aggregation [18] or max-modularity aggregation [19]. For P2, on the other hand, the fastest algorithm for finding all the elementary cycles of general graph is Johnson's algorithm [20], [22] and its complexity is $O((n+e)(c+1))$, where n, e, c are the numbers of nodes, edges, and elementary cycles respectively. Thus, for each subnetwork, the complexity to complete P2 is in the order of $O((2^{m_i+n_i} + 2^{2m_i+n_i})(\tilde{N}_i + 1))$, where $\tilde{N}_i$ is the number of elementary input-state cycles of Σ_i, and it is bounded above by $O(s2^{2m_\alpha+n_\alpha}\tilde{N}_\alpha)$, where $\alpha = \arg\max_i\{2^{2m_i+n_i}\tilde{N}_i\}$. The complexity of P3 together with P4 is no more than $O((N_\beta \bar{\ell})^s)$, where $\beta = \arg\max_i\{N_i\}$, $\bar{\ell}$ is the length of the longest input-state cycles, and N_i is the number of input-state cycles shorter than or equal to 2^n. Note that unlike the elementary input-state cycles, input-state cycles in Boolean control networks may have repeated states in general, thus N_i may be much greater than $\tilde{N}_i$.

The total number of states of Boolean networks with n-nodes is 2^n and the transition from each state to an updated state is unique. Thus, finding the cycles of whole Boolean networks directly requires computations in order 2^n. Comparing $O(s2^{2m_\alpha+n_\alpha}\tilde{N}_\alpha + (N_\beta\bar{\ell})^s)$ with $O(2^n)$, the proposed method will be very efficient if the size of each subnetwork is small enough and N_i's are not too big. If not, the algorithm requires more computations than the one of the brute force

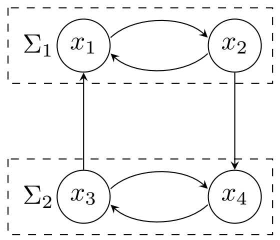


Fig. 6. Boolean network of example 3.9

TABLE I
NUMBER OF INPUT-STATE CYCLES (N_i), WHERE THE PERIOD IS LESS THAN OR EQUAL TO ℓ

ℓ	1	3	5	10	16
N_1	1	3	15	222	8819
N_2	3	8	18	241	8872

computation. This is an inherent difficulty of solving NP-hard problems. For the Boolean network whose network graph is sparse, on the other hand, if it is possible to set a reasonable size $M(\ll n)$, and partition the network such that subnetwork size, $2m_i + n_i$, is less than or equal to M, then the computational reduction will be significant. Note that the sparse network structure is quite common in biological networks [23], [24].

Another very important issue is the number of input-state cycles N_i for each subnetworks. The number of input-state cycles of Boolean control networks may be very large for long period attractors. However, we may be less interested in longer attractors as most of biologically and physically meaningful dynamics are related to the short period attractors including fixed points. If we are to find attractors with the length of period less than or equal to a fixed number T, N_i would not be very large. If $T = 1$, i.e. finding fixed points, the proposed algorithm yields the results extremely quickly. The mean maximum attractor length of Boolean networks is shown to be proportional to $\sqrt{n}$ in [1], thus it is reasonable to set $T \leq \sqrt{n}$, although in some cases there may be attractors longer than $\sqrt{n}$.

In the following example, the strength of the proposed algorithm and the computational issues are highlighted.

Example 3.9: Consider a Boolean network whose network graph is partitioned as shown in Fig. 6 and its dynamics is given by

$$\begin{cases} x_1(t+1) = x_2(t) \vee x_3(t) \\ x_2(t+1) = \neg x_1(t) \\ x_3(t+1) = x_4(t) \\ x_4(t+1) = x_2(t) \wedge x_3(t). \end{cases}$$

N_i, the number of input-state cycles for each subnetwork, whose length is less than or equal to ℓ, is shown in Table I. Even in this simple example, the number of input-state cycles, not the number of elementary input-state cycles, is huge. Thus, trying to find all attractors using the proposed algorithm requires more computation than the brute-force algorithm. The network has in fact only one attractor as follows:

$$\{(1100) \to (1000) \to (0000) \to (0100) \to (1100)\},$$

and it is composed by

$$A_1 = \{(110) \to (100) \to (000) \to (010) \to (110)\},$$

where the state is arranged in (x_1, x_2, x_3), and

$$A_2 = \{(100) \to (100) \to (000) \to (000) \to (100))\},$$

where the state is arranged in (x_2, x_3, x_4), with their projections onto x_2 and x_3 satisfying:

$$\Pi(A_1, 2) = \Pi(A_2, 2) = \{1 \to 1 \to 0 \to 0 \to 1\},$$
$$\Pi(A_1, 3) = \Pi(A_2, 3) = \{0 \to 0\}.$$

In order to find this attractor through the proposed algorithm the computational cost is much larger than the brute-force algorithm. However, if we only want to find fixed points, i.e. $T = 1$, fixed points for both subnetworks are found as follows:

$$A_{11} = \{(101) \to (101)\}$$
$$A_{21} = \{(111) \to (111)\}, \quad A_{22} = \{(100) \to (100)\},$$
$$A_{23} = \{(000) \to (000)\},$$

where A_{ij} is j-th input-state cycle of i-th subnetwork. It is immediately concluded that there are no fixed points as A_{ij} cannot be composed and these can be calculated very quickly.

IV. ACYCLIC AGGREGATION

There could be a large number of input-state cycles with long periods in each subnetwork if a Boolean network is divided into several subnetworks as demonstrated in Example 3.9. As already pointed out earlier, the proposed algorithm is only efficient to find short period attractors. However, the following example shows that we can also find efficiently all the attractors for Boolean networks with a special structure.

Example 4.1: Recall Example 2.2. As Σ_3 has no input, it is a Boolean network itself rather than a Boolean control network and it has a far less number of attractors compared to Boolean control networks in general. It can be easily seen that there is only one attractor of Σ_3 and it is as follows:

$$A_3 = \{(1011) \to (0110) \to (1011)\},$$

where the state is ordered as (x_6, x_7, x_8, x_9). The projection onto its output x_7 is given by

$$\Pi(A_3, 7) = \{0 \to 1 \to 0\}.$$

The attractors of Σ_1 to be composed with A_3 must have the same projection onto its inputs x_7. Thus, the input sequence to Σ_1 can be fixed to $\{0, 1, 0, 1, \ldots\}$. Σ_1 becomes a periodic time-varying Boolean network with the fixed periodic input from Σ_3. As the input is fixed, the number of possible input-state cycles of the Boolean control network Σ_1 is much less than the one with free input. It is now easy to obtain all input-state cycles of Σ_1, with the fixed input sequence. There is only one as follows:

$$A_1 = \{(1001) \to (0000) \to (0011) \to (0110) \to (1001)\},$$

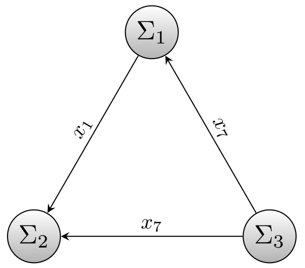


Fig. 7. Example of aggregation graph corresponding to example 4.1

where the state is ordered as (x_1, x_2, x_3, x_7). Finally, as x_1 and x_7 are inputs to Σ_2, we now fix the input sequence, (x_1, x_7), to Σ_2 as follows:

$$\{(11), (00), (01), (00), (11), (00), (01), (00), \cdots\}.$$

The corresponding input-state cycles of Σ_2 are obtained as follows:

$$A_{21} = \{(1101) \to (0100) \to (0101) \to (0100) \to (1101)\},$$
$$A_{22} = \{(1001) \to (0110) \to (0001) \to (0110) \to (1001)\},$$

where the state is ordered as (x_1, x_4, x_5, x_7).

Hence, all attractors of the whole network are found by

$$(A_3 \times A_1) \times A_{21}, \text{ and } (A_3 \times A_1) \times A_{22}.$$

Consider the network graph of a Boolean network shown in Fig. 7, where each block Σ_i represents a super node, and call it *aggregation graph*. If there are no periodic path in the aggregation graph, which is the case of Example 4.1, we call this aggregation an *acyclic aggregation*. In this case, there exist several root blocks, which have no input, i.e. they are Boolean networks. By projecting the outputs from the root blocks onto their child blocks, the child blocks are turned into periodic time-varying Boolean networks. Repeat these procedures until all blocks can be driven by the outputs from their parent blocks. In these procedures, there is no need to find all input-state cycles in each subnetwork and the computational demand decreases significantly. The only remaining question is how to find an acyclic aggregation if there exists any for a given network structure. To this end, we introduce some classical concepts in graph theory in the following:

Definition 4.2:

1) $G = \{\mathcal{X}, E\}$ is a directed network graph, where $\mathcal{X}$ is the set of nodes and E is the set of directed edges. $x_i \to x_j \in E$, if x_i and x_j are elements of $\mathcal{X}$, and there exists an edge starts at x_i and ends at x_j.
2) The directed graph G is called strongly connected, if for any $x_i, x_j \in \mathcal{X}$ there is a path $x_i \to x_{k_1} \to x_{k_2} \to \ldots \to x_{k_p} \to x_j$ from x_i to x_j.
3) $G' = \{\mathcal{X}', E'\}$ is called a subgraph of G, if $\mathcal{X}' \subset \mathcal{X}$, and for all $x_i, x_j \in \mathcal{X}'$, $x_i \to x_j \in E$ implies $x_i \to x_j \in E'$.
4) The subgraph $G' = \{\mathcal{X}', E'\}$ of G is called a strongly connected component, if it is a maximal strongly connected subgraph, i.e. adding any nodes that are not elements of $\mathcal{X}'$ and the corresponding edges to E' makes the obtaining subgraph being not strongly connected. Note that a single node can also be a strongly connected component.
5) For two aggregations of G, $P_1 = \{\mathcal{X}_{11}, \mathcal{X}_{12}, \ldots, \mathcal{X}_{1p}\}$, $P_2 = \{\mathcal{X}_{21}, \mathcal{X}_{22}, \ldots, \mathcal{X}_{2q}\}$, P_1 is said to be finer than P_2, if for any $\mathcal{X}_{1i} \in P_1$, there exists an $\mathcal{X}_{2j} \in P_2$ such that $\mathcal{X}_{1i} \subset \mathcal{X}_{2j}$.

It is easy to see that any two strongly connected components of a directed graph are disjoint. Thus, strongly connected components form an aggregation of the graph and we call it *graph of strongly connected components*. The following two lemmas are also classical results from graph theory. They show the relationship between strongly connected components and acyclic aggregation.

Lemma 4.3: A strongly connected graph does not have acyclic aggregations.

Proof. For any two network partitions, G_1 and G_2, of any aggregation of a strongly connected graph, G, and two arbitrary nodes, $a \in G_1$ and $b \in G_2$, there are paths from a to b, and b to a by the definition of strongly connected graph. Then, the paths form a cycle between G_1 and G_2. Hence, the strongly connected graph cannot be acyclic. □

Lemma 4.4: The graph of strongly connected components of a directed graph G, is acyclic.

Proof. If the graph of strongly connected components of G is not acyclic, there are k strongly connected components, $G_1, G_2, \ldots, G_k$, which form a cycle, where $k > 1$. For any $a \in G_i, b \in G_j$, $i \neq j, i, j = 1, 2, \ldots, k$, there exists a path from a to b and a path from b to a and $\cup_{i=1}^{k} G_i$ turns out to be strongly connected. This contradicts the fact that the strongly connected components are maximally strongly connected subgraphs. □

Combining these two lemmas, the following is trivial:

Corollary 4.5: The graph of strongly connected components of a directed graph is finer than any other acyclic aggregation.

There are many existing efficient algorithms for finding strongly connected components, for example, Tarjan's algorithm [25], which can be used to find the acyclic aggregation of the network graph of a Boolean network.

V. Applications to Biomolecular networks

First, to evaluate the efficiency of the proposed algorithm, the Boolean network model of T-cell receptor kinetics [26] is to be analyzed. The network graph of T-cell receptor kinetics is shown in Fig. 8, where the solid arrows with pointed heads represent activation, the dashed arrows with bar heads represent inhibition, the big bullets represent "AND", and the boxes with more than one arrow pointed to them represent "OR". For example, for $PAGCsk$, there is a dashed arrow from $TCRbind$ and a solid arrow from Fyn pointed to its box, thus its update rule is

$$PAGCsk(t+1) = \neg TCRbind(t) \vee Fyn(t).$$

There are three external inputs, $CD45$, $CD4$, and $TCRlig$ and the inputs are fixed to (1, 1, 1) as [11] so that the analysis of the responses of T-cell receptor kinetics focuses on a specific physiological input situation. The assumption of the inputs

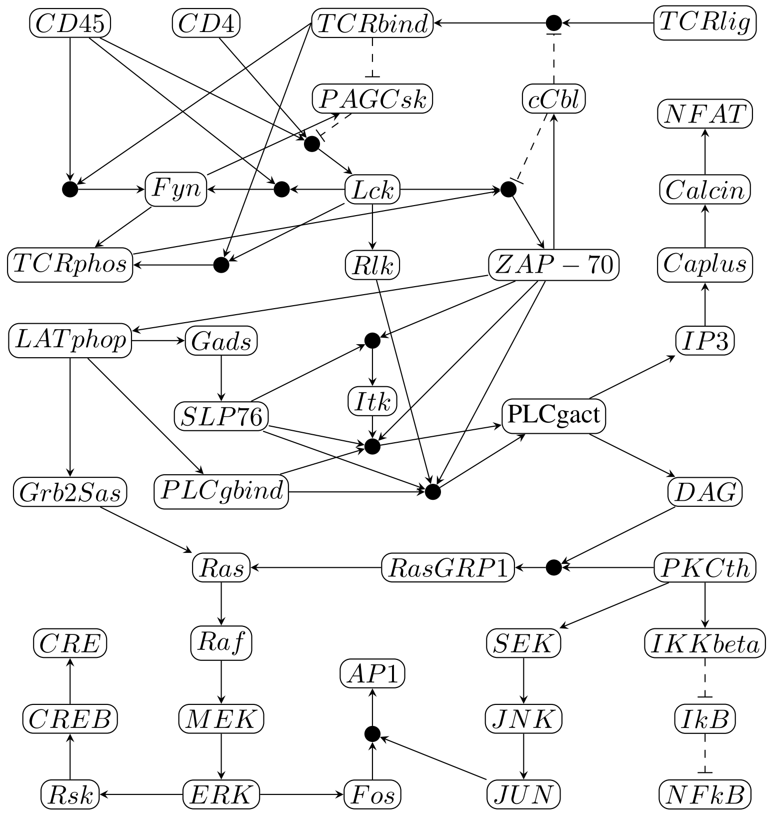


Fig. 8. Boolean network implementing a T-cell receptor model as presented in [26]

being constant during the analysis is based on the fact that for most biological networks external inputs cannot change fast and frequently enough during their dynamic responses. Note that, however, any periodic or constant external input scenarios can be analyzed without any significant increasing computational demand. Second, it is easy to see that the following nodes compose a strongly connected component having no input, i.e. $\mathcal{X}_1$ is a root block,

$$\mathcal{X}_1 = \{TCRbind, PAGCsk, Lck, Fyn, cCbl, \\ TCRphos, ZAP-70\}.$$

The attractors of Σ_1 corresponding to $\mathcal{X}_1$ can be easily found, for example, using the semi-tensor approach toolbox, as it is a Boolean network with only 7 nodes. Using a PC with Dual-Core 2.5GHz CPU, 8G RAM, it takes only 0.0983s to find all (two) attractors as follows:

$$A_{11} = \{(1101010) \rightarrow (1101010)\},$$
$$A_{12} = \{(1111010) \rightarrow (1101011) \rightarrow (1101110) \rightarrow \\ (0101010) \rightarrow (1100010) \rightarrow (1001000) \rightarrow \\ (1111010)\},$$

where the states are ordered as $(TCRbind, PAGCsk, Lck, Fyn, cCbI, TCRphos, ZAP-70)$. For the rest of nodes, each node is now considered as an individual block, i.e., a subgraph with one node. For each single-node block, which does not have any self-feedback, one periodic input will drive only one attractor of the block Thus, each attractor of Σ_1 can generate only one attractor in the whole network. It takes only 0.0278s to calculate two attractors of the whole network. Their projection onto the outputs of the T-cell receptor are as follows:

$$\{(0100) \rightarrow (0100)\}$$
$$\{(0100) \rightarrow (0100) \rightarrow (0100) \rightarrow \\ (0100) \rightarrow (0101) \rightarrow (0100)\},$$

where the states are ordered as $(NFAT, NFkB, AP1, CRE)$.

Hence, it takes a total 0.126s only to find all attractors in the T-cell receptor network. This is remarkable compared to other approaches to find all attractors in such a large Boolean networks. Given that the total number of states is 2^{37}, it is impossible or impractical at least to find all attractors in a reasonable amount of time using general methods such as the semi-tensor product approach [12] or random initial state evaluation [11]. In [11], the authors applied their algorithms to the asynchronous case of T-cell receptor to compare the running times: 20.7 minutes using Genysis's algorithm [27], 11.3s and 2.1s using their algorithm to find 90% percent steady states with 90% confidence and 80% percent steady states with 80% confidence respectively. However, only very few (far less than 1%) of the states in state space need to be considered using their algorithm applied to asynchronous Boolean networks. Their algorithm can also be applied to synchronous Boolean networks, but for synchronous Boolean networks no states can be excluded a priori, thus the total 2^{37} number of states in the state space of T-cell receptor network must be considered and it would take at least several hours to perform all the calculations. Moreover, the computation time increases drastically from hours to years by adding two or more nodes to the network.

The other 7 different external input combination cases are analyzed and the corresponding outputs are summarized in Table II, where inputs are ordered as $(CD45, CD4, TCRlig)$ and outputs are ordered as $(NFAT, NFkB, AP1, CRE)$. The results shown in the table imply that the T-cell receptor response is rather robust, as most input combinations cannot change its steady states if the inputs remain constant for a sufficient amount of time.

Note that the T-Cell receptor model contains only one strongly connected component with more than one node; all the rest are single-node blocks. This structure of network graph is ideal for our algorithm in terms of computational efficiency. In order to demonstrate the performance of the algorithm on a less trivial example, an early flower development network is used [28]. Its network graph comprises 24 nodes (excluding the input nodes) and turns out to have also acyclic aggregation. It has two strongly connected components with more than one node: one has 8 nodes and another has 4 nodes. Using the proposed algorithm, it takes only 0.176s to find all attractors (in fact there is only one), while it is impossible to use a standard PC to analyze the attractors of this network using the semi-tensor product approach directly.

VI. Conclusion

In order to reduce the computational complexity in finding attractors of Boolean networks an aggregation algorithm is developed. The proposed algorithm is based on the idea of dividing the whole network into several subgraphs and of composing the attractors of the whole networks from the input-state cycles found in each subnetwork. The algorithm is shown to be more efficient than finding attractors directly from the whole network in the following scenarios: i) the network graph can be divided into a few subnetworks, whose sizes are small enough to be analyzed using some analytic methods, e.g., the semi-tensor approach, ii) short-period attractors are to be found, and/or iii) the aggregation graph is acyclic.

If a Boolean network is put into an acyclic aggregation by finding strongly connected components in the graph where all components are small enough (say, less than 20 nodes or so), the proposed algorithm finds all attractors of the Boolean network very efficiently. On the other hand, if the network graph cannot be partitioned acyclically (i.e., the network graph is itself strongly connected), or some strongly connected components are too large, then large network components can be divided into smaller blocks using the min-cut criterion; short-period attractors can still be found with little computation using the proposed algorithm.

In many applications, some variables have more than t-wo states, i.e. multi-valued logical networks or their update rules are functions of continuous variables, i.e., mixed-valued logical networks [13], [29]. For example, the state of genes and the concentration of proteins are quantified to more than two levels and simple PID controllers for engineering systems are implemented with logical decision algorithms. Generalization of the suggested algorithm to those cases will produce powerful tools to analyze various dynamical systems.

Acknowledgment

This research was initiated during J. Kim's visiting to the Institute of Systems and Science, Chinese Academy of Science, Beijing, China, in the summer 2011, funded by the Royal Academy of Engineering. The authors thank to Professor Daizhan Cheng at the Institute for the initial motivations and many valuable discussions and comments. The research is mainly conducted during Y. Zhao's visiting in the summer 2012 to University of Glasgow, funded by the summer internship program, University of Glasgow, Glasgow, UK. It is also partly supported by NNSF 61074114, China and EPSRC Grant EP/G036195/1, UK. The authors would also like to thank the editors and reviewers for their valuable comments that make the paper improved a lot.

References

[1] H. D. Jong, "Modeling and simulation of genetic regulatory systems: a literature review," in *INRIA Research Report*, 2000, pp. 1–43.

[2] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets." *J. Theoretical Biology*, vol. 22, no. 3, p. 437, 1969.

[3] R. Albert and R. S. Wang, "Discrete dynamic modeling of cellular signaling networks," in *Methods in Enzymolody*, vol. 467, 2009, pp. 291–306.

[4] T. Cheng, H. L. He, and G. M. Church, "Modeling gene expression with differential equations," in *Pacific Symposium on Biocumputing*, vol. 4, Singapore, 1999, pp. 29–40.

[5] S. Pandey, R. S. Wang, L. Wilson, S. Li, Z. Zhao, T. E. Gookin, S. M. Assmann, and R. Albert, "Boolean modeling of transcriptome data reveals novel modes of heterotrimeric g-protein action," *Molecular Systems Biology*, vol. 6, no. 1, pp. 2375–2387, 2010.

[6] T. Helikar, J. Konvalina, J. Heidel, and J. A. Rogers, "Emergent decision-making in biological signal transduction networks," *Proceedings of the National Academy of Sciences*, vol. 105, no. 6, pp. 1913–1918, 2008.

[7] J.-R. Kim, J. Kim, Y.-K. Kwon, H.-Y. Lee, P. Heslop-Harrison, and K.-H. Cho, "Reduction of complex signaling networks to a representative kernel," *Sci. Signal.*, vol. 4, no. 175, pp. ra35+, 2011.

[8] S. Liang, S. Fuhrman, R. Somogyi, and etc., "Reveal, a general reverse engineering algorithm for inference of genetic network architectures," in *Pacific symposium on biocomputing*, vol. 3, Hawaii, US, 1998, pp. 18–29.

[9] D. Cheng, H. Qi, and Z. Li, "Model construction of Boolean network via observed data," *IEEE Trans. Neural Networks*, vol. 22, no. 4, pp. 525–536, 2011.

[10] J. Heidel, J. Maloney, C. Farrow, and J. Rogers, "Finding cycles in synchronous Boolean networks with applications to biochemical systems," *International Journal of Bifurcation and Chaos in Applied Sciences and Engineering*, vol. 13, no. 3, pp. 535–552, 2003.

[11] F. Ay, F. Xu, and T. Kahveci, "Scalable steady state analysis of Boolean biological regulatory networks," *PLoS ONE*, vol. 4, no. 12, p. e7992, 2009.

[12] D. Cheng and H. Qi, " A linear representation of dynamics of Boolean networks," *IEEE Trans. Aut. Contr.*, vol. 55, no. 10, pp. 2251–2258, 2010.

[13] D. Cheng, H. Qi, and Z. Li, *Analysis and Control of Boolean Networks – A Semi-tensor Product Approach*. London: Springer, 2011.

[14] Q. Zhao, "A remark on "Scalar Equations for synchronous Boolean Networks with biologicapplications" by C. Farrow, J. Heidel, J. Maloney, and J. Rogers," *IEEE Trans. Neural Networks*, vol. 16, no. 6, pp. 1715–1716, 2005.

[15] R. Albert and H. G. Othmer, "The topology and signature of the regulatory interactions predicts the expression pattern of the segment polarity genes in drosophila melanogaster," *J. Theoretical Biology*, vol. 223, no. 1, pp. 1–18, 2003.

[16] D. Cheng, Y. Zhao, J.-R. Kim, and Y.-B. Zhao, "Approximation of Boolean networks," in *Proc. of 10th World Congress on Intelligent Control and Automation (WCICA 2012)*, Beijing, 2012, pp. 2280–2285.

TABLE II
OUTPUTS ATTRACTORS OF T-CELL RECEPTOR NETWORK WITH DIFFERENT INPUTS

Inputs	Output Attractors
(111)	$\{(0100) \to (0100)\}$
	$\{(0100) \to (0100) \to (0100) \to (0100) \to (0101) \to (0100) \to (0100)\}$
(110)	$\{(0100) \to (0100)\}$
(101)	$\{(0100) \to (0100)\}$
(100)	$\{(0100) \to (0100)\}$
(011)	$\{(0100) \to (0100)\}$
(010)	$\{(0100) \to (0100)\}$
(001)	$\{(0100) \to (0100)\}$
(000)	$\{(0100) \to (0100)\}$

[17] H. Ishii, R. Tempo, and E.-W. Bai, "A web aggregation approach for distributed randomized pagerank algorithms," *IEEE Trans. Aut. Contr.*, vol. 57, no. 11, pp. 2703–2717, 2012.

[18] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta, "A survey of kernel and spectral methods for clustering," *Pattern Recognition*, vol. 41, pp. 176–190, 2008.

[19] E. A. Leicht and M. E. J. Newman, "Community structure in directed networks," *Physical Review Letters*, vol. 100, p. 118703, 2008.

[20] D. B. Johnson, "Finding all the elementary circuits of a directed graph," *SIAM J. Comput.*, vol. 4, no. 1, pp. 77–84, 1975.

[21] Y. Zhao, H. Qi, and D. Cheng, "Input-state incidence matrix of Boolean control networks and its applications," *Sys. Contr. Lett.*, vol. 46, no. 12, pp. 767–774, 2012.

[22] P. Mateti and N. Deo, "On algorithms for enumerating all circuits of a graph," *SIAM J. Comput.*, vol. 5, no. 1, pp. 90–99, 1976.

[23] R. D. Leclerc, "Survival of the sparsest: robust gene networks are parsimonious," *Molecular Systems Biology*, vol. 4, no. 1, 2008.

[24] S. Mukherjee, S. Pelech, R. M. Neve, W.-L. Kuo, S. Ziyad, P. T. Spellman, J. W. Gray, and T. P. Speed, "Sparse combinatorial inference with an application in cancer biology," *Bioinformatics*, vol. 25, no. 2, pp. 265–271, 2009.

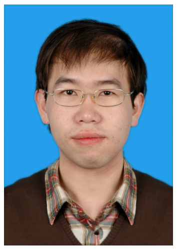
[25] R. E. Tarjan, "Depth-first search and linear graph algorithms," *SIAM J. Comput.*, vol. 1, no. 2, pp. 146–160, 1972.

[26] S. Klamt, J. Saez-Rodriguez, J. A. Lindquist, and etc., "A methodology for the structural and functional analysis of signaling and regulatory networks," *Bioinformatics*, vol. 7, p. 56, 2006.

[27] A. Garg, I. Xenarios, L. Mendoza, and G. DeMicheli, "An efficient method for dynamic analysis of gene regulatory networks and in silico gene perturbation experiments," in *Research in Computational Molecular Biology*. Springer, 2007, pp. 62–76.

[28] F. Wellmer, M. Alves-Ferreira, A. Dubois, J. L. Riechmann, and E. M. Meyerowitz, "Genome-wide analysis of gene expression during early arabidopsis flower development," *PLoS Genetics*, vol. 2, no. 7, p. e117, 2006.

[29] H. Bolouri and E. H. Davidson, "Modeling transcriptional regulatory networks," *BioEssays*, vol. 24, no. 12, pp. 1118–1129, 2002.



Yin Zhao received the B.S. degree in Mathematics and Applied Mathematics from Tsinghua University, Beijing, China in 2008. He is currently a Ph.D. candidate in Academy of Mathematics and Systems Science, Chinese Academy of Sciences. His research interests include complex systems, systems biology, game theory, etc.

Jongrae Kim graduated, in the year 2002, from Texas A & M University at College Station, Texas, USA with a Ph.D. in Aerospace Engineering. He was a post-doctoral researcher with University of California at Santa Barbara, CA, USA in 2002 and 2003 and a research associate with University of Leiceter, UK, in 2004 and 2007. He has been a Lecturer in Biomedical Engineering/Aerospace Sciences, University of Glasgow, UK, since 2007. His main research interests are in the area of robustness analysis, optimal control and estimation, optimisation, and dynamics.

Maurizio Filippone received a Master's degree in Physics and a Ph.D. in Computer Science from the University of Genova, Italy, in 2004 and 2008, respectively.

In 2007, he was a Research Scholar with George Mason University, Fairfax, VA. From 2008 to 2011, he was a Research Associate with the University of Sheffield, U.K. (2008-2009), with the University of Glasgow, U.K. (2010), and with University College London, U.K (2011). He is currently a Lecturer with the University of Glasgow, U.K. His current research interests include statistical methods for pattern recognition.

Dr Filippone serves as an Associate Editor for Pattern Recognition and the IEEE Transactions on Neural Networks and Learning Systems.