Journal of Theoretical Biology 408 (2016) 137–144

Contents lists available at ScienceDirect

Journal of Theoretical Biology

journal homepage: www.elsevier.com/locate/yjtbi

An efficient approach of attractor calculation for large-scale Boolean gene regulatory networks

Qinbin He*, Zhile Xia, Bin Lin

Department of Mathematics, Taizhou University, Linhai, Zhejiang 317000, China

HIGHLIGHTS

- We propose an effective approach of attractor calculation for Boolean networks.
- The approach calculates attractors by using constant nodes and simplified networks.
- Algorithm is effective to calculate all attractors for small average degree networks.

ARTICLE INFO

Article history:
Received 5 May 2016
Received in revised form
17 July 2016
Accepted 8 August 2016
Available online 11 August 2016

Keywords:
Boolean networks
Genetic regulatory networks
Attractor

ABSTRACT

Boolean network models provide an efficient way for studying gene regulatory networks. The main dynamics of a Boolean network is determined by its attractors. Attractor calculation plays a key role for analyzing Boolean gene regulatory networks. An approach of attractor calculation was proposed in this study, which improved the predecessor-based approach. Furthermore, the proposed approach combined with the identification of constant nodes and simplified Boolean networks to accelerate attractor calculation. The proposed algorithm is effective to calculate all attractors for large-scale Boolean gene regulatory networks. If the average degree of the network is not too large, the algorithm can get all attractors of a Boolean network with dozens or even hundreds of nodes.

1. Introduction

Boolean network models provide an efficient way for studying gene regulatory networks, which were first introduced by Kauffman (1969, 1993). Boolean network models can be particularly suited to large-scale gene regulatory networks (Albert and Othmer, 2003; Hu et al., 2015; Bérenguier et al., 2013; Rodríguez et al., 2012; Singh et al., 2012). In a Boolean network, the state of node mainly presents as ON (active, 1 value) and OFF (inactive, 0 value). The main dynamics of a Boolean network is determined by its attractors (steady states or limit cycles). Attractor computing plays a key role for analyzing Boolean gene regulatory networks. The update schemes of Boolean networks mainly are synchronous, asynchronous and stochastic. An n-node Boolean network, there are 2^n possible network states. To calculate all attractors, the full enumeration of the state space is extremely time-consuming for all 2^n possible network states need to be updated or evaluated, especially in large-scale networks. For example, for a 64-node network, with total 2^{64} states and with processing 100,000 states per second, it needs 600,000 years. However, networks based on real life systems can consist of dozens or even hundreds of nodes. Although the full enumeration of the state space is able to calculate all attractors, it is not suitable for computing attractors of large-scale networks. Many approaches have been proposed to calculate attractors in the past years, such as binary decision diagrams (Zheng et al., 2013; Garg et al., 2008), algebraic approach (Veliz-Cuba et al., 2015; Alan Veliz-Cuba et al., 2014; Hinkelmann et al., 2011), sampling approach (Oosawa and Savageau, 2002; Raeymaekers, 2002), reduction approach (Zańudo and Albert, 2013; Alan, 2011; Assieh Saadatpour et al., 2013), integer programming-based method (Qiu et al., 2014; Akutsu et al., 2012), SAT-based algorithm (Dubrova and Teslenko, 2011), sub-network method (Filippone et al., 2008; Leicht and Newman, 2008; Zhao et al., 2003), and predecessor-based approach (Wuensche, 1999; David, 2006). Until now, there is no generic method that can efficiently find every attractor for large-scale networks. The papers of Zheng et al. (2013) and Garg et al. (2008) used binary decision diagrams to calculate attractors for both synchronous and asynchronous networks. The papers of Veliz-Cuba et al. (2015), Alan Veliz-Cuba et al. (2014), and Hinkelmann et al. (2011) calculated all

* Corresponding author.
E-mail addresses: heqinbin@126.com (Q. He), zhilexia@163.com (Z. Xia), tzlinblin@163.com (B. Lin).

http://dx.doi.org/10.1016/j.jtbi.2016.08.006
0022-5193/

steady states of sparse Boolean networks with up to 1000 nodes. The approach based on sampling cannot guarantee finding all attractors. Reduction approach and sub-network method require specific networks and there is a risk of errors to get attractors. Integer programming-based method and SAT-based algorithm cannot effectively deal with large-scale Boolean networks with average in-degree greater than 2. The paper of David (2006) proposed an approach of finding every attractor in a Boolean network. The method analyzes certain partial states to see whether or not they are capable of occurring in an attractor state. Based on the paper of David (2006), this study improved related algorithm and proposed the algorithm considering constant nodes and simplifying networks to accelerate attractor calculation. The proposed algorithm is effective to calculate all attractors for large-scale Boolean networks. If the average degree of the network is not too large, our method can get all attractors of a Boolean network with dozens or even hundreds of nodes.

2. Results

2.1. Mathematical model and properties

In this study the synchronous updating mechanism is only considered for analyzing Boolean networks.

A directed network is composed of n nodes $V = \{v_1, v_2, \ldots, v_n\}$. At each discrete time step $t \geq 0$, each node v_i has a Boolean state $s_i \in \{0, 1\}$ and forms a network state $X = X(t) = (s_1(t), s_2(t), \ldots, s_n(t))$. The set of Boolean functions $f = (f_1, f_2, \ldots, f_n)$ can be expressed as follows:

$$s_i(t+1) = f_i(X(t)) \quad (i \in \{1, \ldots, n\}), \tag{1}$$

or

$$X(t+1) = f(X(t)) = (f_1(X(t)), \ldots, f_n(X(t))). \tag{2}$$

For a Boolean network, let the all input nodes of v_i be $v_{i_1}, v_{i_2}, \ldots, v_{i_k} (i \in \{1, \ldots, n\})$, shown in Fig. 1.

Any a state of $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$ is called a *input window* of v_i. Eq. (1) can be simply illustrated as its logical table, shown in Table 1.

We also use a binary value that uniquely represents a function by $f_i = \left(s_0^{(i)}, s_1^{(i)}, \ldots, s_{2^k-1}^{(i)}\right) (i \in \{1, \ldots, n\})$. Let g be a map, $g: Z \to Z$, and here Z is a finite set. Denote $g(g^{(i)}(z)) = g^{(i+1)}(z)$ $(z \in Z, i = 1, 2, 3, \ldots)$. The following conclusion is obvious.

Lemma 1. *Let g be a map, $g: Z \to Z$, and here Z is a finite set. Then, for any $z_0 \in Z$, $g^{(k)}(z_0) = g^{(p)}(g^{(k)}(z_0))$ for some $k, p > 0$ $(k, p \in \{1, 2, 3, \ldots\})$.*

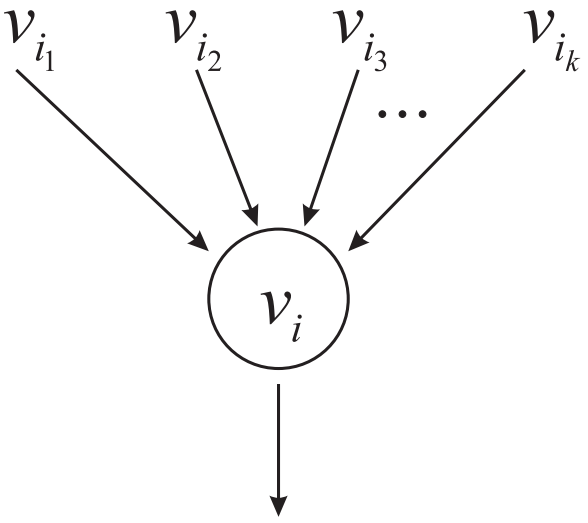

Fig. 1. Node v_i and its input node $v_{i_j}\left(j = 1, 2, \ldots, k\right)$.

Table 1
Logical table of Boolean function f_i.

Input window	f_i
$(s_{i_1}, s_{i_2}, \ldots, s_{i_k})^{(0)} = (0, 0, \ldots, 0)$	$s_0^{(i)}$
$(s_{i_1}, s_{i_2}, \ldots, s_{i_k})^{(1)} = (1, 0, \ldots, 0)$	$s_1^{(i)}$
$(s_{i_1}, s_{i_2}, \ldots, s_{i_k})^{(2)} = (0, 1, \ldots, 0)$	$s_2^{(i)}$
⋮	⋮
$(s_{i_1}, s_{i_2}, \ldots, s_{i_k})^{(2^k-1)} = (1, 1, \ldots, 1)$	$s_{2^k-1}^{(i)}$

Based on Lemma 1, for any an initial state $X = X(0)$ of Boolean network, the system updates itself until the system reaches its final states $A = \{X_0, X_1, \ldots, X_{p-1}\}$ $(p \geq 1)$, called an *attractor*. The attractor could be a limit cycle $(p > 1)$ or a steady state $(p = 1)$. The set of all states that converge to A forms an *attractor basin*. The all states of attractors constitute an *attractor state set*. All states except attractor states are called *transient states* or*non-attractor states.*

For $M \subseteq V = \{v_1, v_2, \ldots, v_n\}$, $y^M \in \{0, 1\}^{|M|}$ is a set of $|M|$ node states, called *a partial state.* A partial state y^M is *contained* in another partial state X^N if $M \subseteq N$ and each node $v_i \in M$ has the same Boolean state in both y^M and X^N, and we also say X^N *containsy*^{M}. A partial state y^M is *stable* if it is contained in some attractor state. A partial state y^M is *unstable* if it is not contained in any an attractor state. If a partial state X^N (regardless the states of the other nodes) will lead to a partial state y^M after time k steps, we call X^N is a *k-predecessor* of y^M, and denote $y^M \underbrace{\prec \cdots \prec}_{k} X^N$. It is obvious that we can get all predecessors of a partial state y^M and thus construct predecessor maps which have a tree-like structure.

For a gene regulatory network, an attractor often represents a special biological function, and therefore the total number of attractors is not so many, otherwise it will lose its biological significance. The directed networks have their *in-degrees* and *out-degrees* (the numbers of edges into and out of each node) independently. The edge into (out) a node is also called an *in-edge* (*out-edge*). In addition, for a directed network, the average in-degree is equal to the average out-degree. In this study, the average in-degree or the average out-degree also is called *average degree*. We study the average degree of gene regulatory networks is generally not large (such as less than 3). The total states of attract state set are not too large, because there are not so many of attractors. It is obvious that a lot of states (including the partial states) do not appear in attractor state set. Thus, it provides us the possibility to efficiently calculate attractors according to the features of attractor state set.

In theory, as long as we exclude all non-attractor states in total state space of a Boolean network, we will get all attractor states. Correspondingly, we will get all attractors of a Boolean network. But for a large-scale Boolean network, it is extremely time consuming to exclude all non-attractor states. In fact, this process just as the full enumeration of the state space is impossible to perform the calculation of attractors. Therefore, we should as many as possible to exclude non-attractor states, so that left as less as possible the remaining states. As long as we traverse these remaining states, we will get all attractors of a Boolean network. There are three key steps. Firstly, the process of exclusion non-attractor states cannot be any error. Otherwise we may not get all attractors. Secondly, we must as many as possible exclude non-attractor states. Thirdly, the process must be efficient to reduce computation.

Non-attractor state is closely related to the unstable partial states. Here, we first introduce some related theorems.