An Integer Programming Framework for Identifying Stable Components in Asynchronous Boolean Networks

Shani Jacobson and Roded Sharan(✉)

Blavatnik School of Computer Science, Tel Aviv University, 69978 Tel Aviv, Israel
roded@tauex.tau.ac.il

Abstract. Executable models of biological circuits offer the ability to simulate their behavior under different settings with important biomedical applications. In particular, Boolean network models have been a prime research focus and dozens of manually curated Boolean models are available in public databases. A key challenge in studying the dynamics of these models is determining their asymptotic behavior, that is the state-sets or attractors they converge to. This is particularly challenging for large networks, as the state space size grows exponentially. Here we introduce a novel method for identifying stable components within attractors under an asynchronous update scheme. Our method leverages the observation that the majority of cellular functions in current models can be described as linear threshold functions, facilitating an efficient integer programming formulation for the problem. We conduct simulations on both synthetic and real biological networks, demonstrating that our proposed method is highly efficient and outperforms previous methods.

Keywords: Boolean network · asynchronous update · attractor finding · quasi attractor · integer linear programming

1 Introduction

The construction of executable models of biological systems is one of the holy grails of systems biology as they allow the simulation of the modeled systems under different environmental and genetic cues [23]. Several models have been proposed to describe a system's dynamics, including those based on differential equations [3,28,29] and discrete models [2,6,22]. The most common framework, studied already 50 years ago in the pioneering work of Kauffman [12], is a Boolean network model (BN) [25]. In this model, the activity value of each node is limited to a binary choice: either it is in an ON state or an OFF state. This activity value is determined by a Boolean function of the values of the node's predecessors in the network. Despite the model's simplicity, it still captures crucial aspects of the dynamic characteristics of the systems being modeled [10].

The model's state at a specific time point can be represented as a vector containing the activity values of all nodes. At each time point, the network's state

J. Ma (Ed.): RECOMB 2024, LNCS 14758, pp. 86–98, 2024.
https://doi.org/10.1007/978-1-0716-3989-4_6

may change according to the previous network state and the Boolean functions of the nodes. Several update schemes that determine which node can update its value at each time point can be considered. The simplest update scheme is the synchronous scheme [7] where all the nodes update their values simultaneously. Under this update scheme each state can transition to only one state and the resulting dynamic behavior is deterministic. In contrast, here we focus on the asynchronous update scheme [26], where only one random node can modify its value at each time point. This update scheme is stochastic in nature, as each state may transition to multiple possible states, depending on the randomly chosen node. While the analysis of asynchronous models is more complex, they resemble better real biological processes [21].

The dynamics of the system consists of all possible trajectories in the state space. Since there is a finite number of states, each trajectory will eventually enter a state-set, called an *attractor*, from which it cannot escape. The attractors characterize asymptotic behavior of the system. From a biological perspective, different attractors represent different modes of the cell, which may indicate potential asymptotic cellular functionalities [11]. An attractor can consist of a single network state, a.k.a. a fixed point, or a group of alternating states. In a fixed-point attractor all node values remain constant, while in a complex attractor some nodes change states (oscillate) while others may remain fixed. Figure 1a presents an example of a BN, and Fig. 1b depicts its state transition graph and its attractors under an asynchronous update scheme.

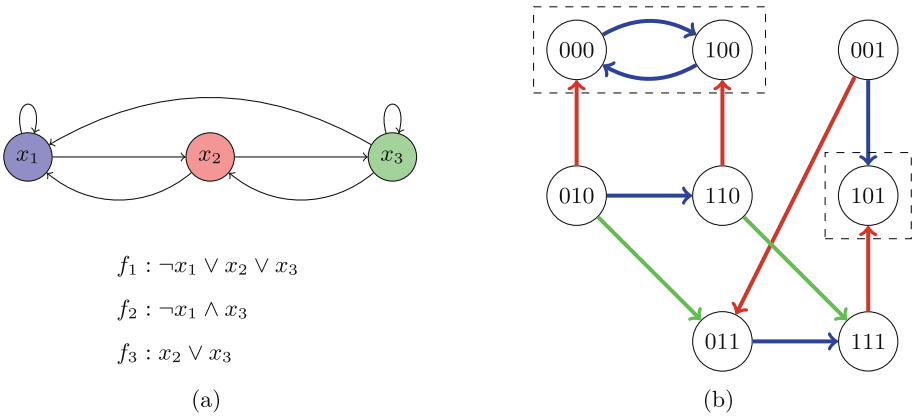


Fig. 1. A Boolean network (a) and its state transition graph (b). The colors of the edges in panel b indicate which node was updated. In this example, the network has two attractors, denoted by dotted boxes: a fixed point attractor and a complex attractor.

While the state space is finite, exhaustive search is not feasible in large networks because the number of states is exponential in the network's size. In the synchronous case, any trajectory that visits the same state twice is inevitably an attractor, facilitating SAT- or ILP-based approaches for their detection [4,5]. In

contrast, an asynchronous update scheme may induce more complex attractor structures, calling for different solutions. Klarner et al. [14,15] and Abdallah et al. [1] used answer set programming, a declarative programming method, to search for trap sets in the state transition graph. Mizera et al. [17] proposed a strongly connected component decomposition method. Several other works were based on creating an acyclic state transition graph, identifying fixed points within it and expanding them to attractors of the original network [24,27].

A state-of-the-art approach for attractor detection in the asynchronous setting is based on reducing the effective network size, thus shrinking the search space to allow exhaustive enumeration [18,30]. This reduction is achieved by identifying nodes that have no impact on the overall network dynamics, such as nodes without outgoing edges, or identifying sets of nodes that, under specific assignments, remain fixed regardless of the values of other nodes in the network. In particular, Albert's group introduced an algorithm to identify all such stable components within an attractor, referred to as a *quasi-attractors* [20,31]. The method searches for stable motifs, which are subsets of nodes together with assigned values that do not depend on the values of the rest of the nodes, and recursively constructs quasi-attractors from them. Notably, the search for stable motifs is based on identifying minimal cycles in the Boolean network and, thus, it is less efficient for networks containing many cycles.

Here we develop a novel integer linear programming (ILP) formulation for finding quasi-attractors in Boolean networks. The ILP simultaneously identifies all nodes that stabilize in an attractor, enabling a rapid and efficient detection of all quasi-attractors. It is based on the observation that the vast majority of biological Boolean functions (96% of the functions in the Cell Collective repository [9]) can be described as linear threshold functions. This insight facilitates the formulation of constraints on the node activity states as dictated by their Boolean functions in a concise manner. Moreover, it allows us to formulate linear constraints that express the stabilization criteria for a node based on the states of other nodes in the network. We evaluate our algorithm on both synthetic and real biological networks, demonstrating that our proposed method is highly efficient and outperforms previous methods.

2 Preliminaries

2.1 Boolean Networks

A Boolean network $\mathsf{BN} = (\mathsf{G}, \mathcal{F})$ consists of a directed graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ with N nodes, and a set $\mathcal{F} = \{f_1, f_2, ..., f_N\}$ of associated Boolean functions. For a node $v_i \in V$, define $\mathcal{P}_i = \{j \in [N] : (v_j, v_i) \in E\}$ as the set of indices of its predecessors in the network, and let $x_i \in \{0, 1\}$ denote its associated binary activity value.

Let $f : \{0,1\}^k \to \{0,1\}$ be a Boolean function. We say that $f(x_1, x_2, ..., x_k)$ is *positive unate* in x_j, if for every possible assignment except x_j, marked by $x_{-j} \in \{0,1\}^{k-1}$:

$$f(\{x_{-j},\ x_j = 0\}) \leq f(\{x_{-j},\ x_j = 1\})$$

Similarly, we say that $f(x_1, x_2, ..., x_k)$ is *negative unate* in x_j, if for every $x_{-j} \in \{0,1\}^{k-1}$:

$$f(\{x_{-j},\ x_j = 0\}) \geq f(\{x_{-j},\ x_j = 1\})$$

If for every $j \in [k]$, f is either positive or negative unate in x_j, we say that f is a *unate* function.

2.2 Linear Threshold Functions

A Boolean function f is a *linear threshold function* (LTF) [19] if and only if there exist $w_1, w_2, ...w_k \in \mathbb{Z}$ and a threshold $\tau \in \mathbb{Z}$ such that for every assignment $x \in \{0,1\}^k$:

$$f(x_1, x_2, ...x_k) = 1 \iff w_1x_1 + w_2x_2 + ... + w_kx_k \geq \tau$$

Note that for every $j \in [k]$, if $w_j > 0$ then f is positive unate in x_j, and if $w_j < 0$ then f is negative unate in x_j. Thus, a necessary (but insufficient) condition for a Boolean function to be a LTF is that the function is unate. Figure 2 illustrates an LTF.

Consider an LTF as described above. Let y_{min} be the minimum value that the LTF can attain, and let $\{x_j^{min}\}_{j=1}^k$ be the assignment that achieves this value:

$$y_{min} = \sum_{j=1}^{k} w_j x_j^{min}, \quad x_j^{min} = \begin{cases} 0 & w_j > 0 \\ 1 & w_j < 0 \end{cases}$$

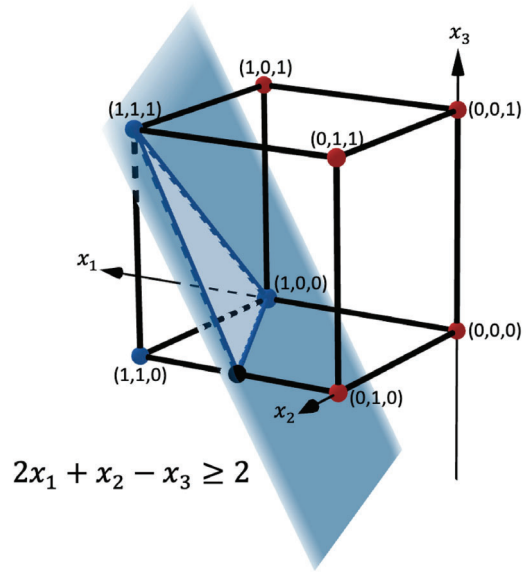
Note that if $y_{min} \geq \tau$ then for any possible assignment the value of f is 1. Similarly, let y_{max} be the maximum value that the LTF can attain, and let $\{x_j^{max}\}_{j=1}^k$ be the assignment that achieves this value:

$$y_{max} = \sum_{j=1}^{k} w_j x_j^{max}, \quad x_j^{max} = \begin{cases} 0 & w_j < 0 \\ 1 & w_j > 0 \end{cases}$$

Note that if $y_{max} \leq \tau$ then for any possible assignment the value of f is 0.

2.3 Network Dynamics

Given a BN with N nodes, a *state* of the network is an assignment of activity values to its nodes. A state $(x_1, \ldots, x_N)$ can transition to another state $(x'_1, \ldots, x'_n)$ under an asynchronous update scheme if they differ by exactly one bit at some position i and the Boolean function f of node i satisfies: $f(x_1, \ldots, x_N) = x'_i$. The corresponding *state transition graph* has 2^N nodes corresponding to all possible network states, and every two nodes s and s' are connected by a directed edge iff s can transition to s'. We say that s *reaches* s' iff $s = s'$ or there is a directed path from s to s' in the state transition graph.



(x_1, x_2, x_3)	$f(x_1, x_2, x_3)$
$(0, 0, 0)$	0
$(0, 0, 1)$	0
$(0, 1, 0)$	0
$(0, 1, 1)$	0
$(1, 0, 0)$	1
$(1, 0, 1)$	0
$(1, 1, 0)$	1
$(1, 1, 1)$	1

Fig. 2. An illustration of the linear threshold function $f(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_1 \wedge \neg x_3)$. Left: 3D-Boolean cube and the plane $2x_1 + x_2 - x_3 = 2$ that defines the LTF. It can be seen that points corresponding to assignments with a value of 1 (blue) are on or to the left of the plane, while points corresponding to assignments with a value of 0 (red) are to the right of the plane. Right: the function's truth table. (Color figure online)

Let $\mathcal{A} \subseteq \{0,1\}^N$ be a set of states in a BN. We say that $\mathcal{A}$ is an *attractor* iff any state in $\mathcal{A}$ can reach any other state in $\mathcal{A}$ and no state outside $\mathcal{A}$. A *quasi-attractor* is a maximal set of nodes and their assignment, such that their values remain fixed under updates regardless of the values of all other nodes.

The definitions of an attractor and a quasi-attractor are closely related. Specifically, it can be observed that each attractor has a corresponding quasi-attractor (which could be empty): a set of nodes that stabilize within the attractor. Conversely, each quasi-attractor defines a subspace within the state space that necessarily contains at least one attractor. For example, in Fig. 1b the fixed attractor, '101', has a corresponding quasi-attractor with the same value, and the complex attractor, {'000', '100'}, has 'X00' as its corresponding quasi-attractor (v_1 oscillates).

3 ILP Framework for Finding Quasi-Attractors

In this section, we present the novel ILP framework we have designed to find quasi-attractors in a Boolean network whose functions are all LTFs. Consider a Boolean network $\mathsf{BN} = (\mathsf{G}, \mathcal{F})$ with N nodes. Define a *stable set* to be a subset

of nodes $S \subseteq V$ of size k and their corresponding values $\{x_i\}_{v_i \in S}$ that satisfy the following conditions:

- if $v_i \in S$, its value remains fixed and equals to x_i, given $\{x_i\}_{v_i \in S}$, and under all possible assignments of the nodes in $V \setminus S$.
- If $v_i \notin S$, given $\{x_i\}_{v_i \in S}$, there must exist two different assignments of nodes in $V \setminus S$ that yield different values for v_i.

3.1 LTF Representation

To represent a unate Boolean function as a linear threshold function, we design an ILP formulation that learns the coefficients and threshold of the representation. While a non-integer linear program could be used instead, the integer formulation is easily solved and the resulting integer coefficients enhance the performance of the main program in Sect. 3.2. Specifically, for a given unate Boolean function f with K predecessors $\mathcal{P} = \{p_1, p_2, \ldots, p_K\}$, we introduce K integer variables $w_1, w_2, \ldots, w_K$ and an integer variable τ. For each possible assignment of the predecessors $(x_1, x_2, ..., x_K) \in \{0,1\}^K$, if $f(x_1, x_2, ..., x_K) = 1$ we add a constraint:

$$\sum_{k=1}^{K} w_k \cdot x_k \geq \tau$$

if $f(x_1, x_2, ..., x_K) = 0$ we add a constraint:

$$\sum_{k=1}^{K} w_k \cdot x_k \leq \tau - 1$$

For a compact representation, our objective minimizes $\sum_{k=1}^{K} |w_k| = \sum_{k=1}^{K} s_k \cdot w_k$, where:

$$s_k = \begin{cases} 1 & \text{if } f \text{ is positive unate in } p_k \\ -1 & \text{if } f \text{ is negative unate } p_k \end{cases}$$

The representations are learned as an initial step of the algorithm and are kept fixed throughout.

Higher Order Threshold Functions. To support a broader range of Boolean functions, we generalized our algorithm to handle Boolean functions that cannot be represented with a single LTF but with a combination of D LTFs. That is, given a Boolean function f with K predecessors, we now allow D LTFs with parameters $\{w_1^{(d)}, w_2^{(d)}, ..., w_K^{(d)}\}_{d=1}^{D}$ and $\{\tau^{(d)}\}_{d=1}^{D}$ such that for every possible assignment $(x_1, x_2, ..., x_K) \in \{0,1\}^K$ it holds that:

$$f(x_1, x_2, ..., x_K) = 1 \iff \forall d \in D : \sum_{k=1}^{K} w_k^{(d)} x_k \geq \tau^{(d)}$$

We call such a function an LTF of order D. Interestingly, 94% of the unate functions in the Cell Collective repository [9] are LTFs of order 1, while the remaining functions are LTFs of order 2.

3.2 Quasi-attractor Detection

For convenience, we define two disjoint sets: $S_0 = \{v_i \in S : x_i = 0\}$ and $S_1 = \{v_i \in S : x_i = 1\}$. Note that $S_0 \cup S_1 = S$. For each $v_i \in V$ with an LTF representation: $\sum_{j \in \mathcal{P}_i} w_j^i x_j \geq \tau_i$, we introduce the following binary variables:

- x_i: A Boolean variable representing the value of node v_i.
- $\mathcal{I}_{v_i \in S}$: An indicator variable for the event $v_i \in S$.
- $\mathcal{I}_{v_i \in S_0}$: An indicator variable for the event $v_i \in S_0$.
- $\mathcal{I}_{v_i \in S_1}$: An indicator variable for the event $v_i \in S_1$.

We now define the constraints that ensure the values of these indicators.

Stable Nodes with Value 0. For the indicator $\mathcal{I}_{v_i \in S_0}$, we introduce variables $\{x_{i,j}^{max}\}_{j \in \mathcal{P}i}$ and y_{max}^i, subject to the following constraints:

$$x_{i,j}^{max} = \begin{cases} x_j & \mathcal{I}_{v_j \in S} = 1 \\ 0 & \mathcal{I}_{v_j \in S} = 0 \text{ and } w_j^i < 0 \\ 1 & \mathcal{I}_{v_j \in S} = 0 \text{ and } w_j^i > 0 \end{cases} \qquad y_{max}^i = \sum_{j \in \mathcal{P}_i} w_j^i x_{i,j}^{max}$$

Here, y_{max}^i represents the maximum value achievable by the left side of the LTF under the assignment of the nodes in S. If $y_{max}^i < \tau_i$ (for any assignment to the nodes in $V \setminus S$), the value of v_i should be fixed at 0, implying the following constraints:

$$y_{max}^i \leq \tau_i - 1 + \Lambda \cdot (1 - \mathcal{I}_{v_i \in S_0})$$
$$y_{max}^i \geq \tau_i - \Lambda \cdot \mathcal{I}_{v_i \in S_0}$$

where $\Lambda > N \cdot \max_{i,j \in \mathcal{P}_i}\{|w_{i,j}|\} + \max_i\{|\tau_i|\}$ is some large constant. These constraints ensure that $\mathcal{I}_{v_i \in S_0} = 1 \iff y_{max}^i < \tau_i$. Finally, we add a constraint to ensure that if $\mathcal{I}_{v_i \in S_0} = 1$ then $x_i = 0$:

$$x_i \leq 1 - \mathcal{I}_{v_i \in S_0}$$

Stable Nodes with Value 1. For the indicator $\mathcal{I}_{v_i \in S_1}$, we introduce variables $\{x_{i,j}^{min}\}_{j \in \mathcal{P}i}$ and y_{min}^i, subject to the following constraints:

$$x_{i,j}^{min} = \begin{cases} x_j & \mathcal{I}_{v_j \in S} = 1 \\ 0 & \mathcal{I}_{v_j \in S} = 0 \text{ and } w_j^i > 0 \\ 1 & \mathcal{I}_{v_j \in S} = 0 \text{ and } w_j^i < 0 \end{cases} \qquad y_{min}^i = \sum_{j \in \mathcal{P}_i} w_j^i x_{i,j}^{min}$$

Here, y_{min}^i represents the minimum value achievable by the left side of the LTF under the assignment of the nodes in S. If $y_{min}^i \geq \tau_i$ (for any assignment to the nodes in $V \setminus S$), the value of v_i is fixed at 1. Therefore, we have the following constraints:

$$y_{min}^i \leq \tau_i - 1 + \Lambda \cdot \mathcal{I}_{v_i \in S_1}$$
$$y_{min}^i \geq \tau_i - \Lambda \cdot (1 - \mathcal{I}_{v_i \in S_1})$$

These constraints ensure that $\mathcal{I}_{v_i \in S_1} = 1 \iff y^i_{min} \geq \tau_i$. Finally, we add a constraint to ensure that if $\mathcal{I}_{v_i \in S_1} = 1$ then $x_i = 1$:

$$x_i \geq \mathcal{I}_{v_i \in S_1}$$

Stable Set Identification. It is immediate to compute the indicator $\mathcal{I}_{v_i \in S}$ from $\mathcal{I}_{v_i \in S_0}$ and $\mathcal{I}_{v_i \in S_1}$. To restrict the ILP to find an assignment with $|S| = k$, we simply require: $\sum_{i \in [N]} \mathcal{I}_{v_i \in S} = k$.

In order to identify all possible solutions, we construct an ILP model for each $k \in [N]$ and run them iteratively until no new solutions are found. To prevent the ILP from finding the same solution twice, we store the stable set of each solution found and add an auxiliary indicator as follows. Let us mark by S^n, S^n_0 and S^n_1 the stable set of the n^{th} solution. The indicator, $\mathcal{I}^n_\Delta$, specifies if the current stable set differs from the n^{th} solution with the following constraints:

$$\sum_{v_i \in S^n_0} (1 - \mathcal{I}_{v_i \in S_0}) + \sum_{v_i \in S^n_1} (1 - \mathcal{I}_{v_i \in S_1}) + \sum_{v_i \notin S^n} \mathcal{I}_{v_i \in S} \leq N \cdot \mathcal{I}^n_\Delta$$

$$\sum_{v_i \in S^n_0} (1 - \mathcal{I}_{v_i \in S_0}) + \sum_{v_i \in S^n_1} (1 - \mathcal{I}_{v_i \in S_1}) + \sum_{v_i \notin S^n} \mathcal{I}_{v_i \in S} \geq \mathcal{I}^n_\Delta$$

3.3 External Nodes

An external node refers to an element or factor that exists outside the network but interacts with nodes within it, thereby exerting influence on their behavior or function. All other nodes are internal nodes. In the analysis of the network dynamics, we allow these external factors to take any value, but their values remain constant throughout the trajectory. To this end, we simply add to the program above binary variables representing the external node activities (one per external factor). Formally, consider a network BN with M external nodes $V_{\text{ext}} = \{v_{N+1}, v_{N+2}, \ldots, v_{N+M}\}$, where $V = V_{\text{int}} \cup V_{\text{ext}}$. As described in Sect. 3.2, the variables $\{x_i\}_{v_i \in V_{\text{int}}}$ represent the values of the internal nodes. We introduce new binary variables $\{e_m\}_{v_m \in V_{\text{ext}}}$ to represent the values of the external nodes and include these variables in the LTF formulations of the internal nodes.

One noteworthy observation in networks that include external nodes is that various assignments of these external factors can often yield identical solutions for the non-external nodes. However, exhaustively finding each of these solutions would be too costly. To address this computational challenge, we introduce a compact ILP designed to identify all potential assignments of external factors in the context of a single solution of the ILP above.

Specifically, given $\{x_i\}_{v_i \in S}$, $S \subseteq V_{\text{int}}$, we seek solutions $\{e_m\}_{v_m \in V_{\text{ext}}}$ such that:

- if $v_i \in S$, its value remains fixed at x_i for all possible assignments of the nodes in $V \setminus S$.
- If $v_i \notin S$, there must exist two different assignments of nodes in $V \setminus S$ that yield different values for v_i.

To this end, we compute maximum and minimum values for every internal node v_i disregarding the external nodes as follows:

$$y^i_{min} = \sum_{\substack{v_j \in V_{\text{int}} \\ j \in \mathcal{P}_i \cap S_1}} w^i_j + \sum_{\substack{v_j \in V_{\text{int}} \\ j \in \mathcal{P}_i \cap (V \setminus S)}} \min\{0, w^i_j\}$$

$$y^i_{max} = \sum_{\substack{v_j \in V_{\text{int}} \\ j \in \mathcal{P}_i \cap S_1}} w^i_j + \sum_{\substack{v_j \in V_{\text{int}} \\ j \in \mathcal{P}_i \cap (V \setminus S)}} \max\{0, w^i_j\}$$

Next, if any of v_i's predecessors is an external node, we add one of the following sets of constraints depending on the status of v_i:

- If $v_i \in S_0$, we make sure its value does not surpass the threshold using the constraint:

$$\sum_{\substack{v_m \in V_{ext} \\ m \in \mathcal{P}_i}} w^i_m e_m \leq \tau_i - y^i_{max} - 1$$

- If $v_i \in S_1$, we similarly add:

$$\sum_{\substack{v_m \in V_{\text{ext}} \\ m \in \mathcal{P}_i}} w^i_m e_m \geq \tau_i - y^i_{min}$$

- If $v_i \in V \setminus S$, we add the following constraints:

$$\sum_{\substack{v_m \in V_{ext} \\ m \in \mathcal{P}_i}} w^i_m e_m \geq \tau_i - y^i_{max}$$

$$\sum_{\substack{v_m \in V_{ext} \\ m \in \mathcal{P}_i}} w^i_m e_m \leq \tau_i - y^i_{min} - 1$$

3.4 Implementation Details

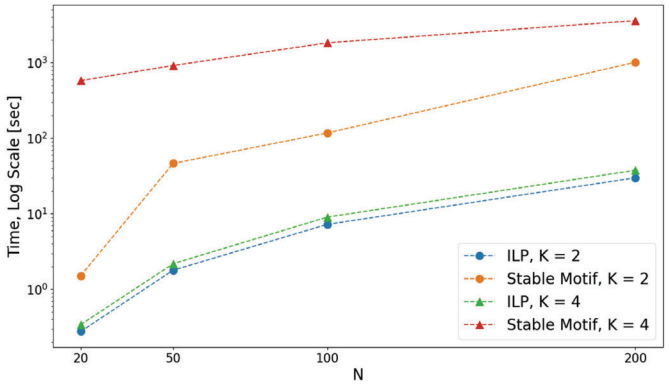
Our complete algorithm implementation is available on GitHub at https://github.com/shanijacobson/AttractorsILP. We utilized the Gurobi optimizer [8] as the solver for our ILP. For Stable Motif, we used the implementation in https://github.com/jcrozum/pystablemotifs. For the answer-set-programming approach, we used the implementation in https://github.com/hklarner/pyboolnet [16]. All experiments were executed on a personal MacBook Air (M1, 2020, 16 GB RAM, 512 GB SSD).

4 Results

We evaluated our algorithm on both synthetic and real biological networks and compared to previous work. The synthetic networks were generated randomly using the N-K model [13], in which each of the N nodes has K predecessors. The real networks were taken from the Cell Collective repository [9].

4.1 Synthetic Networks

We generated synthetic networks using the N-K model [13], with the constraint that their functions are unate. We varied N from 20 to 200 and set $K = 2$ or $K = 4$. These parameter choices follow the characteristics of the real networks analyzed below. For each (N, K) pair, we generated 100 synthetic networks. We applied our ILP algorithm and compared to the Stable Motif (SM) framework [20,31], limiting each algorithm to run up to one hour per network. Figure 3 shows the results for each (N, K) pair. The superiority of our method is evident, as it successfully solves all generated networks and its performance remains almost independent of K. In contrast, Stable Motif encounters growing difficulties as K and N increase along with the number of trajectories in the network.



(a)

(N, K)	ILP	SM
$(20, 2)$	100	100
$(50, 2)$	100	99
$(100, 2)$	100	98
$(200, 2)$	100	77
$(20, 4)$	100	84
$(50, 4)$	100	75
$(100, 4)$	100	65
$(200, 4)$	100	13

(b)

Fig. 3. Performance results on synthetic N-K networks. a) Average running time. b) Percent of network solved within a time limit of one hour.

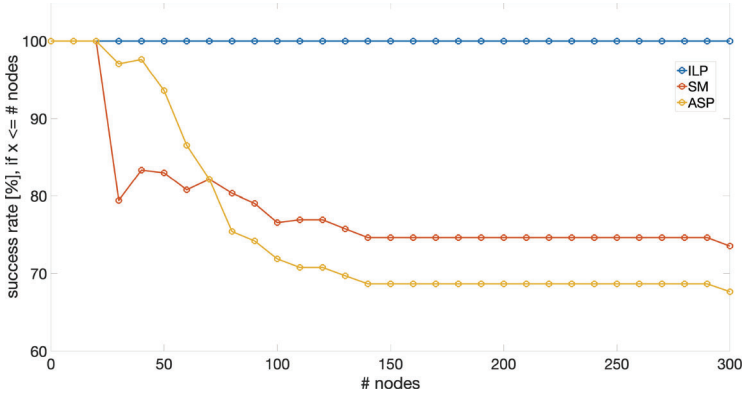


Fig. 4. Success rate of solving real networks under a time limit of 4 h as a function of the total number of nodes in the network.

Table 1. Performance evaluation on real networks with a time limit of 4 h per network.

	Network Parameters					Quasi-Attractors			Performance sec		
	# States	# Externals	# Edges	# 1st order LTFs	# 2nd order LTFs	Fixed	Complex	Avg Stable Nodes	ILP	ASP	SM
Cortical_Area_Development	5	0	14	5	0	2	0	5	**0.05**	0.15	0.2
CD4_T_cell_dynamics_-_Workshop_v2	6	10	40	6	0	10	0	6	**3.72**	191.7	418.57
Metabolic_Interactions_in_the_Gut_Microbiome	8	4	30	8	0	8	0	8	**0.16**	3.2	7.09
Mammalian_Cell_Cycle_2006	9	1	35	8	1	1	1	5.5	**0.38**	0.49	0.65
Regulation_of_the_Larabinose_operon_of_Escherichia_coli	9	4	22	9	0	9	0	9	**0.11**	1.36	1.32
Toll_Pathway_of_Drosophila_Signaling_Pathway	9	2	13	9	0	3	0	9	**0.05**	0.24	0.25
Cell_Cycle_Transcription_by_Coupled_CDK	9	0	19	9	0	1	0	9	**0.11**	0.15	**0.11**
Arabidopsis_thaliana_Cell_Cycle	14	0	66	10	4	0	1	0	1.68	**0.29**	3.19
VEGF_Pathway_of_Drosophila_Signaling_Pathway	10	8	26	10	0	5	0	10	**4.64**	15.55	15.23
Lac_Operon	10	3	24	10	0	4	0	10	**0.15**	0.92	0.8
HH_Pathway_of_Drosophila_Signaling_Pathways	11	13	45	11	0	12	0	11	**3.09**	938.35	997.58
SKBR3_Breast_Cell_Line_Shortterm_ErbB_Network	11	5	46	11	0	31	0	11	**0.35**	278.3	Failure
HCC1954_Breast_Cell_Line_Shortterm_ErbB_Network	11	5	51	11	0	17	0	11	**0.4**	39.25	Failure
BT474_Breast_Cell_Line_Shortterm_ErbB_Network	11	5	51	11	0	28	0	11	**0.42**	87.37	Time out
Wg_Pathway_of_Drosophila_Signalling_Pathways	12	14	43	12	0	10	0	12	**0.33**	1998.9	2552.81
Cardiac_development	13	2	39	12	1	6	0	13	**0.52**	0.86	1.16
TOL_Regulatory_Network	14	10	58	12	2	62	0	14	**6.37**	195.57	128.75
CD4+_T_Cell_Differentiation_and_Plasticity	12	6	84	12	0	24	1	11.88	**8.38**	29.61	Time out
Predicting_Variabilities_in_Cardiac_Gene	13	2	39	12	1	6	0	13	**0.52**	0.89	0.69
Body_Segmentation_in_Drosophila_2013	14	3	31	13	1	5	0	14	**0.22**	1.37	3.48
FGF_pathway_of_Drosophila_Signalling_Pathways	14	9	31	14	0	13	0	14	**4.83**	51.81	86.51
Neurotransmitter_Signaling_Pathway	14	2	22	14	0	1	1	8.5	**0.17**	0.98	3.23
Human_Gonadal_Sex_Determination	19	0	78	15	4	3	0	19	2.17	**0.53**	2.86
Budding_Yeast_Cell_Cycle_2009	18	0	57	15	3	0	1	0	**0.8**	Failure	55.33
B_cell_differentiation	17	5	44	17	0	58	0	17	**0.42**	6.24	19.5
Processing_of_Spz_Network	18	6	34	17	1	42	0	18	**1.01**	6.02	3.34
Aurora_Kinase_A_in_Neuroblastoma	19	4	47	17	2	2	2	15.25	**0.64**	12.3	6.24
TLGL_Survival_Network_2011_Reduced_Network	18	0	43	18	0	1	2	16.67	**0.41**	0.67	2.54
Oxidative_Stress_Pathway	18	1	32	18	0	1	0	18	**0.31**	0.43	54.31
Iron_acquisition_and_oxidative	20	2	40	18	2	0	1	0	**0.55**	2.37	2.33
HCC1954_Breast_Cell_Line_Longterm_ErbB_Network	19	6	74	19	0	212	0	19	**3.15**	2565.34	Failure
BT474_Breast_Cell_Line_Longterm_ErbB_Network	19	6	74	19	0	139	0	19	**1.53**	1721.38	Failure
Mammalian_Cell_Cycle	19	1	51	19	0	3	0	19	**0.35**	0.48	1.12
T_cell_differentiation	19	4	38	19	0	29	0	19	**0.31**	7.37	7.59
SKBR3_Breast_Cell_Line_Longterm_ErbB_Network	21	4	85	21	0	412	0	21	**6.39**	951.62	Failure
PTM_in_Acute_Lymphoblastic_Leukemia	24	2	80	22	2	2	1	23.33	**1.28**	2.95	1.78
Trichostrongylus_retortaeformis	25	1	59	24	1	6	1	23.86	**0.74**	9.61	2.4
FA_BRCA_pathway	28	0	122	25	3	0	1	27	2.94	11.28	**2.45**
Death_Receptor_Signaling	25	3	48	25	0	16	0	25	**0.54**	32.13	7.14
CD4+_T_cell_Differentiation	29	9	100	27	2	237	14	28.63	**10.26**	Time out	Time out
Treatment_of_CastrationResistant_Prostate_Cancer	28	14	64	28	0	954	0	28	**0.85**	Time out	4100.9
Tumour_Cell_Invasion_and_Migration	30	2	158	30	0	7	0	30	**15.15**	97.22	141.2
Cholesterol_Regulatory_Pathway	32	2	43	32	0	3	0	32	**0.74**	1.87	1.17
Bordetella_bronchiseptica	33	0	79	32	1	3	0	33	**1.16**	7.48	4.35
TCell_Signaling_2006	37	3	56	37	0	3	0	37	**1.01**	21.9	2.36
Guard_Cell_Abscisic_Acid_Signaling	40	4	80	39	1	6	4	35.6	**1.06**	82.16	19.41
Apoptosis_Network	39	2	75	39	0	0	8	30.63	**0.96**	132.32	11.57
Differentiation_of_T_lymphocytes	41	9	106	39	2	1510	0	41	**3.05**	Time out	Failure
Virtual_chondrocyte_-GRN_layer	43	10	146	41	2	568	0	43	**14.68**	Time out	1164.62
Virtual_chondrocyte_-PPI_layer	44	12	109	43	1	832	64	43.41	**2.38**	Time out	906.28
Senescence_Associated_Secretory_Phenotype	49	2	98	48	1	14	2	47.56	**2.08**	6356.13	7.86
MAPK_Cancer_Cell_Fate_Network	49	4	108	49	0	6	3	46	**2.02**	Failure	37.67
B_bronchiseptica_and_T_retortaeformis_coinfection	52	1	136	50	2	30	0	52	**3.82**	Time out	Failure
Signaling_Pathway_for_Butanol_Production	53	13	150	53	0	12	76	36.53	**3.53**	Failure	Time out
TLGL_Survival_Network_2011	54	6	201	54	0	5	6	52.18	**4.36**	Time out	305.1
Glucose_Repression_Signaling_2009	55	18	114	54	1	10332	0	55	**27.16**	Failure	Time Out
TLGL_Survival_Network_2008	54	7	200	54	0	11	22	51.12	**5.14**	Time out	722.43
_Bortezomib_Responses_in_U266_Human_Myeloma_Cells	62	5	131	61	1	38	0	62	**3.36**	Time out	94.26
PC12_Cell_Differentiation	61	1	109	61	0	3	0	61	2.58	114.08	**2**
HGF_Signaling_in_Keratinocytes	62	6	109	62	0	13	0	62	**3.04**	Time Out	25.46
_IGVH_mutations_in_chronic_lymphocytic_leukemia	66	25	137	65	1	576	1392	44.95	**11.05**	Time out	Time out
Lymphopoiesis_Regulatory_Network	67	14	174	65	2	12280	161	66.93	**47.83**	Time out	Time out
Colitisassociated_colon_cancer	69	1	154	69	0	2	6	58.375	**4.44**	Time out	2435.06
IL6_Signalling	71	15	163	71	0	16	100	29.21	**5.27**	Failure	Time out
T_Cell_Receptor_Signaling	94	7	165	93	1	56	24	86.73	**4.86**	Time Out	43.71
IL1_Signaling	104	14	232	104	0	36	0	104	**6.77**	Time out	Time out
Influenza_A_Virus_Replication_Cycle	120	11	305	120	0	329	19	119.14	**156.7**	Time out	Time out
Signaling_in_Macrophage_Activation	268	26	552	267	1	112640	36864	329.03	**286.64**	Time Out	Failure

4.2 Real Biological Networks

Next, we applied our algorithm to real BNs taken from the Cell Collective repository [9], a database of Boolean representations of real biological networks. We focused on 68 networks where all functions could be expressed using LTFs. We compared our results against Stable Motif (SM) and against an answer-set programming (ASP) algorithm [14,15], which finds full attractors. We imposed a time limit of 4 hours on each method (per network).

The results are presented in Table 1. Clearly, our ILP-based method outperforms the other two algorithms in almost all tested networks, solving each network on average in 10 s and spending a maximum of 5 min. In comparison, ASP solved only 46 (68%) of the networks and Stable Motif solved only 50 (74%) of the networks under the time limit (Fig. 4). It is worth mentioning that the only one non-LTF network that could be solved by our competitors had 15 nodes and could also be solved by a simple exhaustive search approach. Additionally, in cases where ASP succeeded, we verified that for each quasi-attractor there is exactly one corresponding full attractor; moreover, for each such matching pair, all the nodes that do not stabilize in the quasi-attractor, oscillate in the full attractor. This implies that in all these cases the quasi-attractor finding could be complemented by exhaustive search to identify full attractors efficiently.

5 Conclusions

In this work, we proposed a novel ILP-based method for identifying quasi-attractors in asynchronous Boolean networks. The method leverages the representation of most Boolean functions in biological networks as linear threshold functions to construct an efficient integer program for quasi-attractor detection. Future research may use the properties of linear threshold functions to efficiently find the oscillatory parts of the corresponding attractors.

References

1. Abdallah, E.B., Folschette, M., Roux1, O., Magnin, M.: ASP-based method for the enumeration of attractors in non-deterministic synchronous and asynchronous multi-valued networks. Algorithms Mol. Biol. **12**, 20–23 (2017)
2. Albert, R., Othmer, H.G.: The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in drosophila melanogaster. J. Theor. Biol. **223**(1), 1–18 (2003)
3. Aldridge, B.B., Burke, J.M., Lauffenburger, D.A., Sorger, P.K.: Physicochemical modelling of cell signalling pathways. Nat. Cell Biol. **8**, 1195–1203 (2006)
4. Bruner, A., Sharan, R.: A robustness analysis of dynamic Boolean models of cellular circuits. J. Comput. Biol. **27**, 133–143 (2019)
5. Dubrova, E., Teslenko, M.: A sat-based algorithm for finding attractors in synchronous Boolean networks. IEEE/ACM Trans. Comput. Biol. Bioinform. **8**, 1393–1399 (2011)
6. Espinosa-Soto, C., Padilla-Longoria, P., Alvarez-Buylla, E.R.: A gene regulatory network model for cell-fate determination during Arabidopsis thaliana flower development that is robust and recovers experimental gene expression profiles. Plant Cell **16**(11), 2923–2939 (2004)
7. Fauré, A., Naldi, A., Chaouiya, C., Thieffry, D.: Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. Bioinformatics **22**, e124–e131 (2006)
8. Gurobi Optimization, LLC: Gurobi optimizer reference manual (2023). https://www.gurobi.com

9. Helikar, T., et al.: The cell collective: toward an open and collaborative approach to systems biology. BMC Syst. Biol. **6**, 96 (2012)
10. Huang, S., Ingber, D.E.: Shape-dependent control of cell growth, differentiation, and apoptosis: switching between attractors in cell regulatory networks. Exp. Cell Res. **26**(1), 91–103 (2000)
11. de Jong, H., Page, M.: Search for steady states of piecewise-linear differential equation models of genetic regulatory networks. IEEE/ACM Trans. Comput. Biol. Bioinform. **5**(2), 208–222 (2008)
12. Kauffman, S.: Metabolic stability and epigenesis in randomly constructed genetic nets. J. Theor. Biol. **22**, 437–467 (1969)
13. Kauffman, S.: The Origins of Order: Self-Organization and Selection in Evolution. Spin Glasses and Biology, pp. 61–100. World Scientific (1992)
14. Klarner, H., Bockmayr, A., Siebert, H.: Computing maximal and minimal trap spaces of Boolean networks. Nat. Comput. **14**, 535–544 (2015)
15. Klarner, H., Siebert, H.: Approximating attractors of Boolean networks by iterative CTL model checking. Front. Bioeng. Biotechnol. **3**, #130 (2015)
16. Klarner, H., Streck, A., Siebert, H.: PyBoolNet: a Python package for the generation, analysis and visualization of Boolean networks. Bioinformatics **33**, 770–772 (2017)
17. Mizera, A., Pang, J., Qu, H., Yuan, Q.: Taming asynchrony for attractor detection in large Boolean networks. IEEE/ACM Trans. Comput. Biol. Bioinf. **16**, 31–42 (2018)
18. Naldi, A., Remy, E., Thieffry, D., Chaouiya, C.: Dynamically consistent reduction of logical regulatory graphs. Theor. Comput. Sci. **412**, 2207–2218 (2011)
19. O'Donnell, R.: Analysis of Boolean Functions, pp. 113–141. Cambridge University Press (2014)
20. Rozum, J., Zañudo, J., Gan, X., Deritei, D., Albert, R.: Parity and time reversal elucidate both decision-making in empirical models and attractor scaling in critical Boolean networks. Sci. Adv. **7**, eabf8124 (2021)
21. Saadatpour, A., Albert, I., Albert, R.: Attractor analysis of asynchronous Boolean models of signal transduction networks. J. Theor. Biol. **266**, 641–656 (2010)
22. Saez-Rodriguez, J., et al.: A logical model provides insights into t cell receptor signaling. PLoS Comput. Biol. **3**, e163 (2007)
23. Sharan, R.: Toward a role model. EMBO Rep. **14**(11), 948 (2013)
24. Skodawessely, T., Klemm, K.: Finding attractors in asynchronous Boolean dynamic. Adv. Complex Syst. **14**, 439–449 (2011)
25. Thomas, R.: Boolean formalization of genetic control circuits. J. Theor. Biol. **42**, 563–585 (1973)
26. Thomas, R.: Regulatory networks seen as asynchronous automata: a logical description. J. Theor. Biol. **153**(1), 1–23 (1991)
27. Trinh, V.G., Hiraishi, K., Benhamou, B.: Computing attractors of large-scale asynchronous Boolean networks using minimal trap spaces. In: Proceedings Bioinformatics, Computational Biology and Health (BCB), pp. 1–10 (2022)
28. Tyson, J.J., Chen, K., Novak, B.: Network dynamics and cell physiology. Nat. Rev. Mol. Cell Biol. **2**, 908–916 (2001)
29. Tyson, J.J., Chen, K., Novak, B.: Network dynamics and cell physiology. Curr. Op. Cell Biol. **15**, 221–231 (2003)
30. Veliz-Cuba, A.: Reduction of Boolean network models. Theor. Comput. Sci. **289**, 167–172 (2011)
31. Zanudo, J.G.T., Albert, R.: An effective network reduction approach to find the dynamical repertoire of discrete dynamic networks. Chaos **23**, 025111 (2013)