Towards Optimal Decomposition of Boolean Networks

Cui Su, Jun Pang, and Soumya Paul

Abstract—In recent years, great efforts have been made to analyse biological systems to understand the long-run behaviours. As a well-established formalism for modelling real-life biological systems, Boolean networks (BNs) allow their representation and analysis using formal reasoning and tools. Most biological systems are robust - they can withstand the loss of links and cope with external environmental perturbations. Hence, the BNs used to model such systems are necessarily large and dense, and yet modular. However, existing analysis methods only work well on networks of moderate size. Thus, there is a great need for efficient methods that can handle large-scale BNs and for doing so it is inevitable to exploit both the structural and dynamic properties of the networks. In this paper, we propose a method towards the optimal decomposition of BNs to balance the relation between the structure and dynamics of a network. We show that our method can greatly improve the existing decomposition-based attractor detection by analysing a number of large real-life biological networks.

Index Terms—Boolean networks, strongly connected components, graph partition.

1 Introduction

Systems biology is a rapidly developing field that aims to provide a comprehensive, system-level interpretation of cellular behaviours from a holistic perspective. It includes the computational modelling and the analysis of complex biological systems.

Computational modelling of a biological system provides means to construct a computational/mathematical model from the expression data, which makes it possible to analyse biological systems using formal reasoning and tools. Among various computational modelling frameworks, Boolean networks (BNs) are widely used to quantitatively model large-scale biological systems, such as gene regulatory networks (GRNs) and their signalling pathways. In BNs, genes are modelled as binary nodes, being either expressed or not expressed. The interactions between the genes are described by Boolean functions assigned to each node, from which the structure of the network can be derived. The dynamics of the BNs are determined by a combination of the Boolean functions and the update modes, such as the *synchronous* mode or the *asynchronous* mode. The main advantage of BNs lies in being simple and yet being able to capture important biological and systemic properties of the biological systems. Specifically, the steady states of BNs called *attractors*, correspond to cellular phenotypes [1] or functional cellular states, such as proliferation, apoptosis or differentiation etc [2]. Thanks to the rapid expansion of biological databases, a number of large-scale BNs are constructed to model biological systems. These BNs are considered to be more stable and are thus likely to better withstand the loss of a link and cope with external environmental perturbations, which are important characteristics of such systems [3]. This leads to a growing need of efficient and scalable methods to analyse such large-scale BNs.

The efficient detection of attractors is a major challenge in the analysis of biological systems. It is non-trivial to identify all the attractors of a BN, since we need to explore the entire state space of the network, the size of which is exponential in the number of nodes. Great efforts have been made to develop efficient attractor detection algorithms and tools [4], [5], [6], [7], [8], [9]. These methods are mainly designed for BNs with the synchronous update mode, where all the nodes are updated simultaneously according to their Boolean functions. However, this update mode is considered not realistic in biology. The nodes are more likely to be updated asynchronously [10], which means that at each time step, a node is randomly chosen to be updated according to its function. Thus, every state may have multiple outgoing edges under asynchronous mode and an attractor is a bottom strongly connected component (SCC) in the transition system. Due to the differences of the transition systems under two update modes, the methods in [4], [5], [6], [7], [8] are not applicable to asynchronous BNs. Regarding to the satisfiability (SAT)-based method [9], its ability is greatly hindered as the potentially complex attractor structure leads to prohibitively large SAT formulas.

It has been known for a while that to detect the attractors of a BN efficiently, an algorithm should exploit both the structure and the dynamics of the BN [11]. In this spirit, Mizera *et al.* [12] developed a decomposition-based approach to accurately detect all the attractors in large asynchronous BNs. The main idea was to decompose a BN into SCCs based on its network structure and construct a weighted directed acyclic graph (DAG), the *SCC graph*. In an SCC graph, every vertex corresponds to an SCC and there is a directed edge between two vertices if and only if one of the associated SCCs depends on the vertices of the other (*control*

• *C. Su, J. Pang, and S. Paul are with the Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg. J. Pang, and S. Paul are also with the Faculty of Science, Technology and Communication, University of Luxembourg.*
E-mail: firstname.lastname@uni.lu
Correspondence should be addressed to J. Pang (jun.pang@uni.lu).

nodes). After that, 'blocks' are formed by merging SCCs with their control nodes and the blocks are sorted according to their topological ordering. In every block, the state transition system (STS) for that block was constructed focusing on the local network dynamics and the 'local' attractor detection is performed on this STS. These local attractors are then combined block-wise to derive the attractors for the entire BN. It is worth noting that the vital principle that guarantees the correctness of the decomposition-based approach [12] is that the decomposed components of the network should form a DAG. However, the efficiency of the SCC-decomposition-based approach is hindered when a BN has a large number of SCCs, which commonly arises in biological networks. This motivates the need for the optimal decomposition of BNs, which balances the structure and the dynamics of the networks and increases efficiency of the attractor detection. In this paper, we address this problem of the optimal decomposition of BNs by partitioning the SCC graphs to balance the size of different components, while maintaining the acyclic property between the components.

There are a few requirements to be satisfied when searching for an optimal partitioning of the SCC graphs (resulting from the SCC-decomposition of BNs). The first is that the weight of a partition should not exceed an upper bound. This upper bound is the core in balancing the relation between the structural and the dynamic properties of networks. If the upper bound is too large, the partitioned graph might have parts that are almost equal to the entire BN. It is too expensive to analyse the dynamics of such a partition in terms of time cost and memory. On the contrary, if the upper bound is too small, there will be a large number of partitions. In this case, even though the state space of every partition is small, it might still be time consuming to traverse all the partitions. The second is that the number of control nodes of each partition needs to be minimised. This is mainly due to the fact that we perform analysis in blocks, which are formed by partitions together with their control nodes. By minimising the number of control nodes, the sizes of blocks will be reduced, leading to smaller state transition systems. Last but not least, without overrunning the upper bound of the partition weight, the number of partitions should be minimised as well. Fewer parts lead to fewer blocks, and as a consequence we analyse fewer state transition systems.

The problem is similar to that of acyclic partitioning of DAGs which asks to partition a weighted DAG into k disjoint subsets such that the maximum weight of a subset is smaller than a given threshold while minimising the weight of the edges between these subsets. This problem is known to be NP-hard in the strong sense (e.g., see [13]) and hence efficient algorithms for the general case are highly unlikely. However, extensive work has been done in the literature to address this problem for special classes of graphs and to find efficient but non-optimal solutions [14], [15], [16], [17]. In our context, the work that is the most relevent is a multilevel graph partitioning method developed in [18]. This method employs the methods in [14], [15] and consists of three phases: coarsening, initial partitioning and refinement. They have demonstrated that their methods are much faster and can compute higher quality partitions than similar tools and algorithms in the literature [18]. However, this method cannot be applied directly to partition the SCC graphs of BNs and certain modifications should be made. First, in the SCC graph, the weight of an edge should be extended to a set of nodes. In this way, the weight of a cut set is the union of the weight of the edges in the set without duplicate elements. Secondly, unlike in [18], where the number of partitions is given as a predefined value, we compute a minimal number of partitions, within a threshold determined by the BN at hand.

With the above restrictions in mind, in this paper, we propose a method towards optimal decomposition of BNs. We first perform the SCC-decomposition and form an SCC graph. Then we partition the SCC graph in three steps: initial partitioning, topological refinement and iteration. The initial partitioning computes a preliminary solution. This solution is improved upon in the topological refinement, including the refinement of the boundary vertices and the partitions at the same topological level. After that, we iteratively perform the initial partitioning and the topological refinement to further improve the partitioning. We aim to decompose a BN in an optimal way, which balances the ratio between the structure and the dynamics of a BN. To demonstrate the effectiveness of the newly proposed decomposition of BNs, we apply it to the SCC-decomposition-based approach for attractor detection [12] and perform experiments on a number of biological networks. The results show that the performance of the method [12] can be significantly improved with our new decomposition method.

2 PRELIMINARIES

This section recalls the preliminary notions on directed graphs and BNs needed in the course of the paper.

2.1 Acyclic partitioning of DAGs

A *directed graph* $G(V, E)$ consists of a non-empty set of vertices V and a set of directed edges E. Each edge e in E is an ordered pair of vertices $u, v \in V$, denoted by (u, v). u is a *predecessor* of v and v is a *successor* of u. Let $G(V, E)$ be a directed graph and let $n = |V|$ and $m = |E|$ denote the number of vertices and edges resp. The *vertex weight* and *edge weight* of G are functions $\delta : V \to \mathbb{N}$ and $\eta : E \to \mathbb{N}$. A directed graph is said to be *strongly connected* if for every pair of nodes $u, v \in V$, there is a directed path from u to v and vice versa. A *strongly connected component (SCC)* of a directed graph is a maximal strongly connected subgraph.

A directed graph is called a *directed acyclic graph* (DAG) if it does not contain any directed cycles. Every DAG has a topological ordering, an ordering of the vertices V that for every edge $e(u, v), u, v \in V$, vertex u comes before vertex v in the ordering. We introduce the concept of topological credits as follows.

Definition 1 (Topological credit). Given a DAG $G(V, E)$, a vertex u with no parent has a credit of 0, denoted as $\varphi(u) = 0$. A vertex v with a set of parents $par(v)$ has a credit of $\varphi(v) = max_{v' \in par(v)}(\varphi(v')) + 1$.

A DAG is said to be *weighted* if it has associated vertex and edge weights. A $k-$*way partitioning* of a weighted DAG $G(V, E)$ divides the vertices V into k disjoint subsets $V_1, V_2, \ldots, V_k$. For any partition $V_i, i \in [1, k]$, its weight

equals to $\sum_{v \in V_i} \delta(v)$. A *cut set* is a set of edges whose endpoints are in different partitions, i.e. $\{e(u,v) \in E \mid u \in V_i, v \in V_j\}$, where $V_i, V_j \in V$ and $V_i \neq V_j$. The *weight of a cut set* Γ is the sum of the weights of the edges in the set, $\sum_{e \in \Gamma} \eta(e)$. The formal definition of cut set is given below.

Definition 2 (Cut set). A *cut set* of a $k-$ way partitioning of a weighted DAG $G(V,E)$ is a set of edges with the following properties.

- The removal of all edges in the set disconnects G.
- The removal of some (but not all) of the edges in the set does not disconnect G.

The partitioning of a weighted DAG, that maintains the acyclic property between different partitions, is called acyclic partitioning. In general, the maximal weight of a partition is limited by an upper bound L_{max}. Here, we give the formal definition of acyclic partitioning of a weighted DAG.

Definition 3 (Acyclic partitioning of a weighted DAG). Given a weighted DAG $G(V,E)$ and an upper bound L_{max}, find a partitioning $P = \{V_1, V_2, \ldots, V_k\}$, where $V_1 \cup \ldots \cup V_k = V$ and $V_i \cap V_j = \emptyset$ $(i \neq j)$, such that the maximal weight of one partition is not greater than L_{max} and the weight of the cut set is minimised. Moreover, the relation between $V_1, V_2, \ldots, V_k$ is acyclic.

2.2 Boolean networks

Boolean networks (BNs) were first proposed by Kauffman in 1969 as a network model of GRNs [1]. In BNs, genes are represented by nodes and the interactions between the genes are described as Boolean functions.

Definition 4 (Boolean network). A *Boolean network* $G(V, \boldsymbol{f})$ comprises of a set of nodes $V = \{v_1, v_2, \ldots, v_n\}$, and a vector of Boolean functions $\boldsymbol{f} = (f_1, f_2, \ldots, f_n)$, where $f_i : \{x_{i_1}, x_{i_2}, \ldots, x_{i_{k(i)}}\} \rightarrow \{0,1\}$ is a Boolean function associated with node v_i $(i = 1, 2, \ldots, n)$ and $x_{i_j} \in \{0,1\}$ for $j \in [1, k(i)]$ is a value assigned to node v_{i_j}. A state of the network is given by a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$, where $x_i \in \{0,1\}$ is a value assigned to node v_i.

A BN $G(V, \boldsymbol{f})$ can be viewed as a directed graph $G(V,E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ represents the set of nodes. For two nodes $v_i, v_j \in V$, there is a directed edge from node v_j to node v_i if and only if the Boolean function of v_i, i.e. f_i, depends on v_j. If the edge exists, node v_j is a *parent node* of v_i and node v_i is a *child node* of v_j. Every node $v_i \in V$ has a set of parent nodes $\{v_{i_1}, v_{i_2}, \ldots, v_{i_{k(i)}}\}$, where $k(i)$ denotes the number of parent nodes and $1 \leq i_1 < i_2 < \cdots < i_{k(i)} \leq n$. The state of a BN at a discrete time point t $(t = 0, 1, 2, \ldots)$ is given by a vector $\boldsymbol{x}(t) = (x_1(t), x_2(t), \ldots, x_n(t))$, where $x_i(t)$ is a binary variable, which represents the value of node v_i at time point t. The value of node v_i at time point $t+1$ is determined by its Boolean function f_i applied to the values of the set of parent nodes of v_i at time t, i.e., $x_i(t+1) = f_i(x_{i_1}(t), x_{i_2}(t), \ldots, x_{i_{k(i)}}(t))$.

Starting from an initial state, the dynamics of a BN evolves by transiting from one state to another based on different update modes [19]. Among several update modes, the synchronous and the asynchronous modes are the most popular ones. In the synchronous mode, all the nodes are updated simultaneously; while in the asynchronous mode, one node is non-deterministically chosen to be updated at a time. The synchronous mode is used mainly due to its simplicity; however, for modelling biological systems, the asynchronous mode is more suitable since the expression of a gene is usually not an instantaneous process [10]. Thus, in this paper, we focus on the asynchronous mode.

The state transitions of a BN can be illustrated by the state transition system (STS), which is defined below.

Definition 5 (State transition system). A state transition system $\mathcal{T}$ is a 3-tuple $\langle S, S_0, T \rangle$ where S is a finite set of states, $S_0 \subseteq S$ is the initial set of states and $T \subseteq S \times S$ is the transition relation, specifying the evolution of the system. When $S = S_0$, we write $\langle S, T \rangle$.

Based on Definition 5, we can model the dynamics of an asynchronous BN as a state transition system: the set S is the state space of the BN; the initial set S_0 equals to S as all the states are accessible; the transition relation T is given by the following equation.

$$T(\boldsymbol{x}(t), \boldsymbol{x}(t+1)) = \bigwedge_{j=1, j\neq i}^{n} \Big(x_j(t+1) \leftrightarrow x_j(t)\Big) \wedge \Big(x_i(t+1) \leftrightarrow f_i(x_{i_1}(t), x_{i_2}(t), \cdots, x_{i_{k(i)}}(t))\Big). \quad (1)$$

The attractors are the key characters of a BN as they depict the long-run behaviours of the BN [20] and are hypothesised to characterise cellular phenotypes in biological systems [1]. The attractors are formally defined below.

Definition 6 (Attractor of a BN). An *attractor* of a BN is a set of states satisfying that any state in this set can be reached from any other state in this set and no state in this set can reach any other state that is not in this set.

Example 1. **The PC12 cell differentiation network** was developed by Offermann *et al.* [21]. It is a comprehensive model used to clarify the cellular decisions towards proliferation or differentiation. This network consists of 33 nodes and has 7 single-state attractors. We refer to [21] for the structure of the network derived from the Boolean functions.

2.3 SCC-decomposition of BNs

A BN $G(V,E)$ can be decomposed into a set of maximal SCCs, denoted as $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$, where m is the number of SCCs. The set of SCCs is disjoint and the union of the SCCs equals to the whole set of nodes, namely $c_i \cap c_j = \emptyset$ for any $c_i \neq c_j$ in $\mathcal{C}$ and $c_1 \cup c_2 \cup \ldots \cup c_m = V$.

Considering SCCs $\mathcal{C}$ as vertices, we can form a *weighted directed graph* $\mathcal{G}(\mathcal{C}, \mathcal{E})$, which we call the *SCC graph* of a given BN $G(V,E)$. Every vertex $c \in \mathcal{C}$ of $\mathcal{G}$ corresponds to an SCC in the BN, so the number of vertices $|\mathcal{C}|$ equals to m. For any pair of SCCs $c_i, c_j \in \mathcal{C}, c_i \neq c_j$, there is a directed edge from c_i to c_j if and only if there exists a directed edge from a node in SCC c_i to a node in SCC c_j in the associated BN $G(V,E)$. In such case, we say that SCC c_i is a parent SCC of c_j. Let $par(c_j)$ denote the set of parent SCCs of c_j. The set of nodes, satisfying $\{u \mid e(u,v) \in E, u \in c_i \text{ for } c_i \in par(c_j), v \in c_j\}$, are control nodes of c_j, denoted as $ctr(c_j)$.

We assign vertex and edge weights, δ and η, to the SCC graph $\mathcal{G}(\mathcal{C}, \mathcal{E})$ as follows. δ is a function $\delta : \mathcal{C} \rightarrow \mathbb{N}$, such that

for every vertex $c_i \in \mathcal{C}$, $\delta(c_i) = |c_i|$, representing the number of nodes contained in this SCC. η is a function $\mathcal{E} \rightarrow 2^V$ where for every edge $e(c_i, c_j) \in \mathcal{E}$, $\eta(e)$ is given as $\eta(e) = (ctr(c_j) \cap c_i)$. Thus the weight of the edge e corresponds to the set of control nodes of SCC c_j from SCC c_i. It is easy to observe that the SCC graph $\mathcal{G}$ is also a DAG, since the directed edges in $\mathcal{E}$ are all from the parent SCCs to the child SCCs and there are no directed cycles.

An SCC together with its control nodes forms a *block*, denoted as B. We can form a *block graph* from the SCC graph by having the vertices as blocks and a directed edge from block B_i to B_j if and only if there is a directed edge from the SCC corresponding to B_i to that corresponding to B_j in the SCC graph. In this way, the block graph is also a DAG. If a control node in a block B_i belongs to an SCC which forms a block B_j, block B_j is called a parent of block B_i and B_i is a child of B_j. Let $par(B_i)$ denote the set of parent blocks of B_i and $ctr(B_i)$ denote the set of control nodes of B_i with respect to all the parent blocks. We refer to a block as an *elementary block* if it does not contain any control nodes, otherwise, it is a *non-elementary block*. We notice that as long as the block graph is guaranteed to be a DAG, other strategies to form blocks can also be used. Two blocks can be merged into a larger block.

Example 2. To continue with the PC12 cell network, we perform SCC-decomposition on this network. The network has 19 SCCs and the corresponding SCC graph is given in Figure 1. Each vertex represents an SCC. Among all the SCCs, only SCCs 3, 15 and 16 have 3, 2 and 12 nodes. Thus, the weights of vertices 3, 15 and 16 are 3, 2 and 12. The rest of the SCCs contain only one node and their vertex weights equal to 1. Regarding to an SCC c, the weights of edges from this SCC to its child SCCs are a subset of nodes in the SCC. Thus, for the SCCs with only one node, the weights of edges from this SCC to its child SCCs contain only that node.

The PC12 cell network with only 33 nodes has 19 SCCs, of which 16 are single-node SCCs. Each SCC with its control nodes forms a block. In order to analyse the entire network, we need to construct a local transition system for every block and analyse the constructed transition system. A large number of single-node SCCs hampers the efficiency of decomposition-based analysis methods. Moreover, the SCC-decomposition often leads to single-node SCCs in biological networks, which motivates us to develop the present decomposition method.

3 Decomposition of BNs

As mentioned in the introduction, the analysis of large-scale BNs often suffers from the infamous state explosion problem. Hence, we propose a new decomposition of BNs to divide an entire network into small components, whose transition systems are of acceptable sizes. This method is generic and can be applied to any analysis method.

3.1 Problem definition

As described in Section 2.3, the SCCs of a BN form a weighted DAG $\mathcal{G}(\mathcal{C}, \mathcal{E})$. Every vertex corresponds to an SCC and a vertex weight is an integer representing the number of nodes contained in the associated SCC. An edge weight represents the set of control nodes of the child SCC from its parent SCC.

The SCC graph partitioning is an instance of the acyclic DAG partitioning problem with some specific constraints. On the one hand, the weight of any partition should not exceed an upper bound L_{max}. This upper bound is to restrict the number of nodes contained in one partition. The value of L_{max} should not exceed the maximal value of the vertex weight, since it is common that a biological network contains a big SCC that cannot be split any further [22]. On the other hand, the size of the cut set weight, i.e. the number of control nodes, should be minimised. Since a block is formed by a partition and its control nodes, less control nodes lead to smaller blocks with smaller state transition systems. Moreover, within the upper bound of the partition weight, the number of partitions k should be minimised. This is motivated by the fact that after decomposing a BN, we need to construct local state transition systems for each block, which is time-consuming if there are too many blocks. These constraints naturally lead to the formal definition of acyclic partitioning of SCC graphs.

Definition 7 (Acyclic partitioning of an SCC graph). Given an SCC graph, which is a weighted DAG $\mathcal{G}(\mathcal{C}, \mathcal{E})$ with a vertex weight δ and an edge weight η, find a partitioning $P = \{V_1, V_2, \ldots, V_k\}$, where $V_1 \cup \ldots \cup V_k = \mathcal{C}$ and $V_i \cap V_j = \emptyset (i \neq j)$, satisfying the following constraints.

1) The maximal weight of any partition is equal or smaller than a threshold L_{max}.
2) The weight of the cut set $\eta(\Gamma)$ is minimised.
3) The number of partitions k is minimised.
4) The graph formed by $P = \{V_1, V_2, \ldots, V_k\}$ is still a DAG.

The acyclic partitioning of an SCC graph differs from the acyclic partitioning of a weighted DAG mainly in two aspects.

1) The weight of a cut set Γ is different in the two problems. In the DAG partitioning, the weight of an edge is an integer, so that the weight of a cut set Γ is the sum of the weights of the edges in Γ, i.e. $\eta(\Gamma) = \sum_{e \in \Gamma} \eta(e)$. In the SCC graph partitioning, the weight of an edge is a set of nodes. There may exist duplicate nodes in the weights of different edges. The weight of a cut set Γ is the union of the weights of edges in Γ.
2) The number of final partitions k is not predefined. Without overrunning an upper bound of the partition weight, we would like to get as few partitions as possible.

3.2 Acyclic partitioning of SCC graphs

We propose a method for SCC graph partitioning based on the methods in [18]. Our method contains three phases: the initial partitioning, the topological refinement and the iteration of the partitioning. The initial partitioning performs a preliminary partitioning of the vertices based on the topological ordering. In the topological refinement phase, we refine the preliminary solution by adjusting the boundary vertices as well as the partitions at the same topological level. Furthermore, we iterate the initial partitioning and the

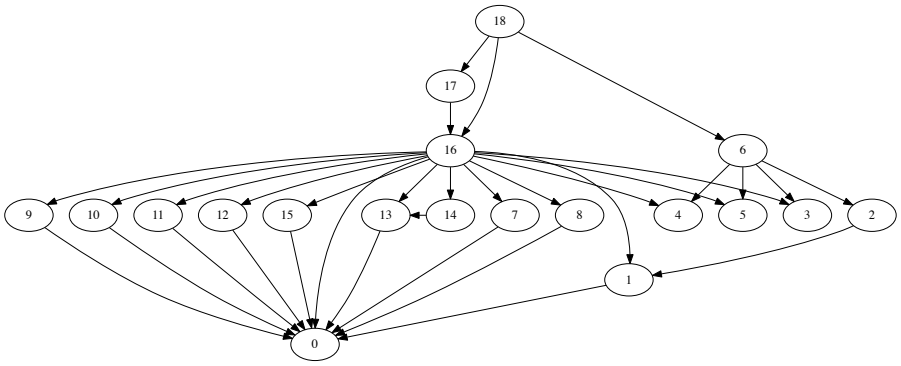


Fig. 1: The SCC graph of the PC12 cell network. Each vertex represents an SCC.

topological refinement until there is no more improvement in the partitioning results.

3.2.1 Initial partitioning

We adapt the greedy algorithm in [18] to perform the initial partitioning of SCCs. In this phase, we compute an initial solution based on the topological ordering of the weighted DAG.

The input to our algorithm is a weighted DAG $\mathcal{G}(\mathcal{C}, \mathcal{E})$ formed from a BN as described in Section 2.3. The vertices $\mathcal{C}$ are sorted according to their topological credits, denoted as $\varphi(u), u \in \mathcal{C}$. The upper bound of the partition weight L_{max} is a hyper-parameter of our decomposition method. Each vertex u in $\mathcal{C}$ is assigned with two properties: *part* and *free*, representing the partition which u belongs to and the mobility of u. The mobility of a vertex u determines if the vertex u is *movable* or not. We consider a vertex is movable if it has no predecessors or the mobility *free* of its direct predecessors are *false* as in [18].

The *gain* of moving a vertex u to a partition j is computed by checking the status of all its predecessors and successors after the move, as described in [18]. Applying the greedy strategy, among a number of movable vertices, the best vertex to be moved is the one with the lowest topological credit and the highest gain. It is worth noting that, to maintain the acyclic dependency of the graph, a vertex u should not be moved to a partition $j, j \in [1, k]$, if $\varphi(u) - max_{v \in V_j}(\varphi(v)) > 1$ or $min_{v \in V_j}(\varphi(v)) - \varphi(u) > 1$.

To perform the initial partitioning, we first initialise the number of partitions $k = 0$ and set all the vertices to movable. Then, we repeat the following steps as long as there exist vertices that are movable.

1) Save movable vertices to a set set.
2) For every vertex $u \in set$, we compute the gain of assigning u to partition k and insert a tuple $(u, \varphi(u), gain)$ to a queue $queue$.
3) After computing the gains for all the vertices in set, we sort the tuples in $queue$ with respect to the topological credit in an ascending order first and then again, in each credit level, according to the $gain$ in a descending order. Note that when we sort the queue $queue$ based on $gain$, the sorted topological ordering is preserved since the sorting is done separately for each credit level.
4) If the total weight of partition k is less than the upper bound L_{max}, we traverse the tuples in $queue$ based on the sorted order.
   a) Release the first item $(u, \varphi(u), gain)$ in $queue$.
   b) If partition k is empty, we assign vertex u to partition k and set the maximum and the minimum topological credits of partition k as $\varphi(u)$. Otherwise, vertex u can be assigned to partition k if and only if two conditions are satisfied. The first condition is that the weight of partition k will not overrun the upper bound L_{max} after the assignment. The second condition is that $\varphi(u) - max_{v \in V_k}(\varphi(v)) \leq 1$ and $min_{v \in V_k}(\varphi(v)) - \varphi(u) \leq 1$.
   c) If vertex u is assigned to partition k, we set $free(u)$ as $false$ and check if its direct successors are movable or not. If a direct successor of u, denoted as v is movable, we compute the gain of assigning v to partition k, insert $(v, \varphi(v), gain)$ to $queue$ and sort $queue$ as Step 3.
5) If $queue$ is not empty, we increase k by one and go to step 1.

After the initial partitioning, the value of k is the number of partitions we computed so far. The results might be improved in the topological refinement and the iterations.

Remark. The differences of our initial partitioning with the one in [18] mainly lie in three aspects.

1) The computation of a gain of assigning a vertex u to a partition j. In the DAG formed by SCCs, we extended an edge weight from an integer to a set of nodes. We consider the initial size of the cut set weight as the number of all the control nodes in the BN. The gain indicates the decrease of the size of the cut set weight. Since the weight of two edges may have an overlap of nodes, to compute the cut set weight, a simple sum of the edge weights as [18] is not applicable here. The cut set weight is the union of the weight of the edges in the set without duplicate elements. Let M denote the set of predecessors of u that belongs to partition j, and Q denote the set of successors of u. The gain of assigning the vertex u to a partition j is $|\bigcup_{v \in M} \eta(e(v, u))| - |\bigcup_{p \in Q} \eta(e(u, p))|$. A larger gain indicates a larger decrease of the size of the cut set weight.
2) We take both the topological credit and the gain into

account to decide the assignment. The vertices are traversed according to an ascending order of their topological credits. When a vertex is assigned to some partitions, the mobility of its child vertices may become movable. More importantly, simply considering the gain may violate the acyclic property. Given two vertices $u, v \in \mathcal{C}$ and $\varphi(u) < \varphi(v)$, it may happen that v is assigned to a partition i and u is assigned to a partition j that has higher credit. Thus, loops may appear in the partitioned graph. Hence, we propose the heuristic that the topological credit has higher priority than the gain. All the movable vertices are processed first according to their topological credits. With the same credits, the assignment of the vertex with the highest gain will be handled first.

3) We only assign a vertex u to a partition j, if $\varphi(u) - max_{v \in V_j}(\varphi(v)) \leq 1$ and $min_{v \in V_j}(\varphi(v)) - \varphi(u) \leq 1$. This condition is adapted to guarantee the acyclic property of the partitioning result.

3.2.2 Topological refinement

The main objective of the topological refinement is to refine the preliminary solution based on the topological information. The refinement is achieved in two steps: the refinement of the *boundary vertices* and the refinement of the partitions with the same topological credits.

Refinement of the boundary vertices. Boundary vertices include *incoming boundary vertices* and *outgoing boundary vertices*. A vertex is an incoming boundary vertex if it has no predecessors or all its predecessors are in other partitions. Similarly, a vertex is an outgoing boundary vertex if it has no successors or all its successors are in other partitions. Note that after the initial partitioning, only the boundary vertices can be moved without violating the acyclic dependency. An incoming boundary u can be moved to a partition $j, j \in [1, k]$ which satisfies two constraints. The first is that partition j contains at least one predecessor of vertex u. The second is that partition j has the maximum topological credit among the partitions that meet the first constraint. In a similar way, an outgoing boundary v can be moved to a partition j that satisfies two constraints: partition j contains at least one successor of v and has the minimum topological credit among the partitions satisfying the previous constraint. It is worth noting that the violation of either constraint may lead to cycles. For the cases, where there are multiple partitions satisfying the constraints, the partition with the largest gain will be considered.

We adapt the FM-like, move-based direct "k-way" refinement algorithm in [18] to refine the initial partitioning. We propose two modifications.

1) The computation of a gain when moving a vertex u from the current partition $part(u)$ to another partition j. Motivated by the same reason explained in Section 3.2.1, we modify the computation of the gain in this phase. The gain denotes the increase/decrease of the size of the cut set weight $\eta(\Gamma)$ after the movement. Let M denote the set of predecessors/successors of u that are in partition j, Q denote the set of predecessors/successors of u that are not in partition j. If $part(u) > j$, the gain is $|\bigcup_{v \in M} \eta(v, u)| - |\bigcup_{p \in Q} \eta(u, p)|$. Otherwise, the gain is $|\bigcup_{v \in M} \eta(u, v)| - |\bigcup_{p \in Q} \eta(p, u)|$. We drop the movements that cause an overrun on the total weight of a partition j.
2) After all possible movements, the property *part* of vertices and the number of parts k are updated, since the refinement may cause empty partitions.

Refinement of the partitions with the same topological credits. Suppose the SCCs are divided into k partitions. All the partitions are topologically sorted and the maximal credit is φ_{max}. If $k > \varphi_{max}$, we perform this step to minimise the number of partitions. This step is straightforward yet effective. We simply group the partitions with the same topological credits, if the weight of the new partition is not greater than the upper bound L_{max}. In this way, every partition is considered as a whole and the partitioning is refined at the topological level.

3.2.3 Iteration

Despite the effectiveness of the initial partitioning and the topological refinement, there is still room for further improvement. In the initial partitioning and the refinement of the boundary vertices, we compute solutions at the level of vertices. For large graphs with complex structures, to compute a near optimal partitioning, it is insufficient to traverse the vertices only once. The refinement of the partitions with the same topological credits can only improve the partitioning at the same topological level.

In order to achieve a near optimal partitioning, we iteratively apply the initial partitioning and the topological refinement to a new graph constructed by treating each partition in the computed solution as a vertex. Suppose we have a solution $P = \{V_1, V_2, \ldots, V_k\}$ after performing the initial partitioning and refinement phases. Considering each partition $V_i \in P$ as a vertex, we form a DAG $G(P, E)$. The weight of a vertex $V_i \in P$ equals to the weight of the corresponding partition, denoted as $\delta(V_i)$. There is an edge from vertex V_i to vertex V_j, $V_i, V_j \in P$ and $V_i \neq V_j$, if and only if there is an edge from a node in V_i to a node in V_j in the original BN. The weight of an edge $e(V_i, V_j)$ is a set $\{u \mid e(u, v) \in E$ in the BN, $u \in V_i, v \in V_j\}$. We perform the initial partitioning and the topological refinement in the smaller DAG $G(P, E)$, convert the new solution to a DAG and repeat this process, until the number of partitions k stays the same.

During the iteration, every partition is considered as a vertex and we utilise the initial partitioning and the topological refinement to group these partitions. Thus, we can further improve the partitioning crossing different topological levels.

Example 3. To illustrate the partitioning method, we give the partitioning results of the PC12 cell network in Figure 2 and Table 1. The upper bound of a partition weight is 12, which is the maximal size of the SCCs. In Table 1, the indices of SCCs are consistent with Figure 1. Figure 2a shows the results of the initial partitioning phase. 19 SCCs are partitioned into 8 partitions. Figure 2b is the results of the topological refinement. In this phase, by adjusting the boundary vertices, the number of partitions is reduced to 5. Partition T_2 and T_3 are at the same topological level. However, the total weight of these

two partitions exceeds the upper bound, they cannot be merged into one group. The two phases are iterated three times to improve the partitioning. In the second iteration, the two partitions T_1 and T_2 in Figure 2b are merged. The third iteration shows that the results can not be improved any further and we have 4 partitions in the end as shown in Figure 2c.

3.3 Integration with the SCC-decomposition method

In [12], we proposed an SCC-decomposition method for attractor detection of asynchronous BNs, where we simply decompose BNs into SCCs in order to divide the networks into small components while preserving the acyclic dependency between the components. However, the SCC-decomposition often leads to a large number of SCCs (e.g., illustrated by the PC12 network), which makes it only suit well-structured BNs. Here, we integrate our new network decomposition method with the SCC-decomposition-based approach for attractor detection. This is realised by reformulating the function "FORM_BLOCK($\mathcal{G}$)" in [12] as below.

1) Decompose a BN into SCCs and construct the associated SCC graph $\mathcal{G}$, where a vertex describes an SCC and there is an edge from vertex c_i to vertex c_j, if the corresponding SCC c_j depends on SCC c_i. Note that the size of an SCC is assigned to the associated vertex as weight. The set of control nodes of a child SCC c_j from a parent SCC c_i is assigned to the edge $e(c_i, c_j)$ as weight.
2) Apply the SCC graph partitioning described in Section 3.2 to $\mathcal{G}$.
3) Form blocks by combing merged SCCs with their control nodes.
4) Sort the blocks in an ascending order according to their topological credits.

Based on this, we can perform the rest of procedures for attractor detection as described in [12].

4 Evaluation

To demonstrate the efficiency of our new decomposition method, we compare the performance of the attractor detection with the SCC-decomposition [12] and the new decomposition method on real-life biological models. The SCC graph partitioning is implemented in Python based on an efficient Python module graph-tool [23]. The decomposition-based attractor detection is implemented in our software tool ASSA-PBN [24], [25], based on the model checker [26] to encode BNs into the efficient data structure binary decision diagrams. All the experiments are performed on a computer (MacBook Pro), which contains a CPU of Intel Core i7 @3.1 GHz and 8 GB of DDR3 RAM. We first describe the three large biological networks below, and give an overview of these networks in Table 2.

The Apoptosis network models the central intrinsic and extrinsic apoptosis pathways and the pathways connected with them [27]. It gives insights into the feedback loops and the crosstalk effects. This network consists of 97 nodes. Among these nodes, the housekeeping node is fixed to ON to model constitutive activation of certain nodes in the network.

The T-cell signalling network is a large-scale logical model for T cell activation, which not only reveals important structural features, like feedback loops and network-wide dependencies, but also captures the global behaviours of this complex network for both natural and perturbed conditions [28]. This network consists of 99 nodes.

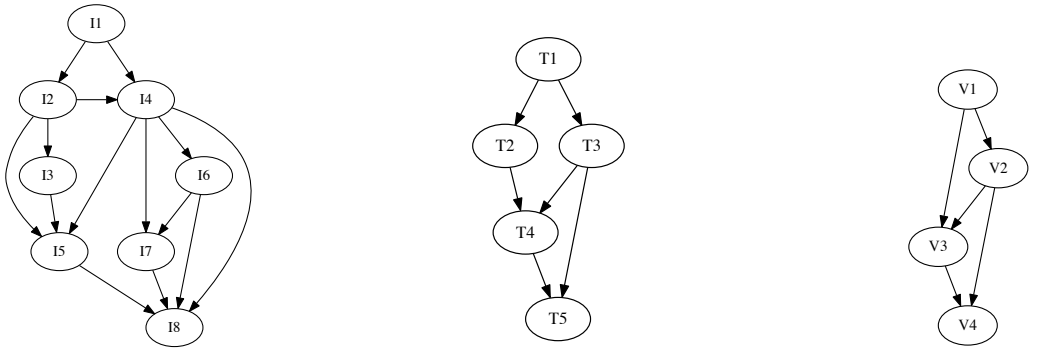
The CD4^+ T-Cell network is a computational model of a CD4C immune effector T-cell to imitate cellular dynamics and molecular signalling under both healthy and immunocompromised conditions (i.e., leukemic conditions) [29]. This model can predict and examine the heterogeneous effects and mechanisms of CAV1 *in silico*. This network consists of 188 nodes.

We start with the SCC-decomposition of the networks. As shown in Table 2, these three networks can be decomposed into 60, 68 and 103 SCCs, respectively. The maximal size of SCCs of the three network are 33, 32 and 86. It is worth noticing that in the CD4^+ T-Cell network, there is a big SCC containing 86 nodes, the rest 102 nodes forms 102 SCCs. This means that every node not in the big SCC forms a single-node SCC.

When performing the new decomposition method, the upper bound of a partition weight is set as $L_{max} = scale * max_{SCC}$, where max_{SCC} is the maximal size of SCCs in the considered network. In order to analyse the influence of the upper bound on the performance of attractor detection, we perform the SCC graph partitioning with a list of upper bounds by setting $scale$ from 0.1 to 1 with an increment of 0.1. The number of partitions and the corresponding weight of the cut set $|\eta(\Gamma)|$ are listed in Table 3. The results on the T-cell signalling network and the CD4^+ T-cell network show that, bigger upper bounds somehow do not guarantee less partitions. This is mainly due to that the SCCs with lower topological credits have a higher priority to be assigned to a partition, which influences the results. In general, the weight of the cut set $\eta(\Gamma)$ shrinks as the scale increases. We do not give the detailed results regarding to the number of iterations, while the results of the experiments indicate that the iteration improves the partitioning results in terms of the number of partitions and the weight of the cut set. For the three networks, to compute the final results, the initial partitioning and the topological refinement are iterated for one to three times.

We perform attractor detection with the SCC-based decomposition method and the decomposition method developed here, on the three networks. The detected attractors with the two decomposition methods are identical, since both methods decompose the networks into DAGs and their correctness is shown in [12]. The apoptosis network has 4 attractors when 10 input nodes are fixed. For the T-cell signalling network, 2 attractors are detected under the condition that all the input nodes are fixed to OFF. For the CD4^+ T-cell network, 6 attractors are detected when the input nodes are also fixed to OFF.

The time costs of the attractor detection are given in Table 3. T_{SCC} and T_{new} represent the time cost of the attractor detection with the SCC-decomposition and the new decomposition, respectively. The results indicate that with the new decomposition method, the efficiency of the attractor detection is greatly improved. With the SCC-decomposition, the



(a) Initial partitioning (b) Topological refinement (c) Iteration

Fig. 2: Partitioning the SCC graph of the PC12 cell network.

Initial partitioning	SCCs	Topological refinement	SCCs	Iteration	SCCs
$I1$	$[18]$	$T1$	$[17, 18]$	$V1$	$[17, 18, 2, 6]$
$I2$	$[6, 17]$	$T2$	$[2, 6]$	$V2$	$[16]$
$I3$	$[2]$	$T3$	$[16]$	$V3$	$[1, 3{-}5, 7{-}12]$
$I4$	$[16]$	$T4$	$[1, 3{-}5, 7{-}12]$	$V4$	$[0, 13, 14, 15]$
$I5$	$[1, 3{-}5, 7{-}12]$	$T5$	$[0, 13, 14, 15]$		
$I6$	$[14, 15]$				
$I7$	$[13]$				
$I8$	$[0]$				

TABLE 1: Partitioning the SCC graph of the PC12 cell network.

Networks	# nodes	# edges	# attractors	# SCCs	max_{SCC}
apoptosis	97	192	4	60	33
T-cell signalling	99	163	2	68	32
$CD4^+$ T-cell	188	380	6	103	86

TABLE 2: An overview of the three real-life biological networks.

time costs for the three networks are 5.44, 2.98 and 446.44 seconds respectively. With the new decomposition method, the range of the time costs are $(0.13 - 3.96)$, $(0.10 - 2.57)$ and $(70.55 - 133.10)$ seconds. The speedups gained by the new decomposition on the attractor detection are computed by T_{SCC}/T_{new}. It is obvious that, regardless of the scale, our new decomposition method can always improve the efficiency of the decomposition-based attractor detection. Especially, in most cases, we can get speedups above 28 for the apoptosis network and above 18 for the T-cell signalling network. Furthermore, the scale or the upper bound has an influence on the speedup. We can see that the trend of the speedups fluctuates with the increase of the scale. When the scale is around 0.6, the new decomposition method usually gains higher speedups.

Besides the upper bound, the weight of the cut set and the network density may also have an influence on the performance of the decomposition-based attractor detection methods. Further investigation is still required to discover the correlations between these factors and the efficiency of the decomposition-based attractor detection method.

5 Conclusion

As discussed in Section 1, to analyse large-scale networks, it is inevitable to extend and improve the capability of the SCC-decomposition-based attractor detection by developing a method taking the structure and the dynamics of the networks into account. To address this challenge, we propose a new decomposition of BNs by partitioning the SCC graphs. Derived from the network structure and dynamics, our optimisation on the decomposition of BNs can be considered as acyclic DAG partitioning while minimising (1) the size of the weight of the cut set and (2) the number of partitions without exceeding (3) the upper bound of the partition weight.

We developed a multilevel method to partition the SCC graphs, based on the methods [18]. We have demonstrated the effectiveness of our new decomposition method by applying it to improve the efficiency of the SCC-decomposition-based approach for attractor detection on a number of real-life biological networks. With the new decomposition of BNs, the efficiency of the decomposition-based attractor detection is greatly improved.

We found that the efficiency of the decomposition-based attractor detection can be influenced by many factors, including the network density, the upper bound of a partition weight, the weight of the cut set. We plan to perform intensive experiments on biological networks to eventually find the optimal decomposition of BNs that suits well for the decomposition-based approach for attractor detection.

Besides the attractor detection, we believe the new de-

Networks	scale	# partitions	$\lvert\eta(\Gamma)\rvert$	T_{SCC} (s)	T_{new} (s)	speedups
apoptosis	0.1	23	76	5.44	0.17	31.47
	0.2	16	73		0.16	34.68
	0.3	9	62		2.66	2.05
	0.4	7	57		3.96	1.38
	0.5	6	53		0.13	41.56
	0.6	5	50		0.16	34.03
	0.7	5	46		0.19	28.96
	0.8	5	44		0.18	29.59
	0.9	5	43		3.00	1.82
	1.0	4	41		2.36	2.31
T-cell signalling	0.1	25	65	2.98	1.83	1.63
	0.2	14	57		0.12	24.42
	0.3	10	48		0.16	18.97
	0.4	7	41		0.11	26.36
	0.5	9	50		2.52	1.18
	0.6	7	43		1.78	1.67
	0.7	9	41		0.10	30.40
	0.8	7	42		2.57	1.16
	0.9	8	48		0.10	28.64
	1.0	7	42		0.11	26.84
$CD4^+$ T-cell	0.1	19	104	446.44	70.63	6.32
	0.2	8	86		84.55	5.28
	0.3	9	91		99.83	4.47
	0.4	6	92		70.55	6.33
	0.5	5	82		80.74	5.53
	0.6	4	79		73.97	6.04
	0.7	4	82		133.10	3.35
	0.8	4	82		124.96	3.57
	0.9	5	76		115.62	3.86
	1.0	4	62		82.96	5.38

TABLE 3: Results of the new decomposition method and the SCC-decomposition method for attractor detection on the three real-life biological networks.

composition of BNs can also contribute to improve our decomposition-based target control of BNs [30]. Similar to the attractor detection, the factors should also be adjusted to obtain the best performance for the target control of BNs.

Moreover, an important advantage of the new decomposition of BNs is that the numbers of control nodes among different partitions are minimised, this contributes to the preservation of information when we construct the local state transition systems for the non-elementary blocks according to the behaviours of their control nodes. Based on this, we can develop an approximate decomposition-based target control approach by constructing the local transition systems for non-elementary blocks only according to projecting the behaviours of the control nodes in the basins of the parent blocks. In this way, we sacrifice accuracy for efficiency, while with the optimal decomposition of BNs, we believe such an approach can compute a good approximation with a reasonable execution time for the analysis of large biological networks.

ACKNOWLEDGMENTS

This work is partially supported by the research project SEC-PBN (funded by the University of Luxembourg) and the ANR-FNR project AlgoReCell. We thank Yann Le Corre for his implementation and comments on the method in [18].

REFERENCES

[1] S. A. Kauffman, "Homeostasis and differentiation in random genetic control networks," *Nature*, vol. 224, no. 5215, pp. 177–178, 1969.

[2] S. Huang, "Genomics, complexity and drug discovery: insights from Boolean network models of cellular regulation," *Pharmacogenomics*, vol. 2, no. 3, pp. 203–222, 2001.

[3] L. Stone, "The feasibility and stability of large complex biological networks: a random matrix approach," *Scientific Reports*, vol. 8, no. 1, p. 8246, 2018.

[4] R. Somogyi and L. D. Greller, "The dynamics of molecular networks: applications to therapeutic discovery," *Drug Discovery Today*, vol. 6, no. 24, pp. 1267–1277, 2001.

[5] D. J. Irons, "Improving the efficiency of attractor cycle identification in Boolean networks," *Physica D: Nonlinear Phenomena*, vol. 217, no. 1, pp. 7–21, 2006.

[6] Y. Zhao, J. Kim, and M. Filippone, "Aggregation algorithm towards large-scale Boolean network analysis," *IEEE Transactions on Automatic Control*, vol. 58, no. 8, pp. 1976–1985, 2013.

[7] W. Guo, G. Yang, W. Wu, L. He, and M. Sun, "A parallel attractor finding algorithm based on Boolean satisfiability for genetic regulatory networks," *PLOS ONE*, vol. 9, no. 4, p. e94258, 2014.

[8] Q. Yuan, H. Qu, , J. Pang, and A. Mizera, "Improving BDD-based attractor detection for synchronous Boolean networks," *Science China Information Sciences*, vol. 59, no. 8, p. 080101, 2016.

[9] E. Dubrova and M. Teslenko, "A SAT-based algorithm for finding attractors in synchronous Boolean networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 8, no. 5, pp. 1393–1399, 2011.

[10] R. Thomas, "Regulatory networks seen as asynchronous automata: a logical description," *Journal of Theoretical Biology*, vol. 153, no. 1, pp. 1–23, 1991.

[11] A. J. Gates and L. M. Rocha, "Control of complex networks requires both structure and dynamics," *Scientific Reports*, vol. 6, p. 24456, 2016.

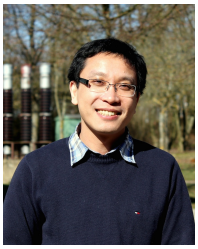
[12] A. Mizera, J. Pang, H. Qu, and Q. Yuan, "Taming asynchrony for attractor detection in large Boolean networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 16, no. 1, pp. 31–42, 2019.

[13] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[14] B. W. Kernighan, "Optimal sequential partitions of graphs," *Journal of the ACM*, vol. 18, no. 1, pp. 34–40, 1971.

[15] J. Cong, Z. Li, and R. Bagrodia, "Acyclic multi-way partitioning of Boolean networks," in *Proc. 31st Annual Design Automation Conference*. ACM, 1994, pp. 670–675.
[16] N. Fauzia, V. Elango, M. Ravishankar, J. Ramanujam, F. Rastello, A. Rountev, L.-N. Pouchet, and P. Sadayappan, "Beyond reuse distance analysis: Dynamic analysis for characterization of data locality potential," *ACM Transactions on Architecture and Code Optimization*, vol. 10, no. 4, p. 53, 2013.
[17] O. Moreira, M. Popp, and C. Schulz, "Evolutionary acyclic graph partitioning," *arXiv preprint arXiv:1709.08563*, 2017.
[18] J. Herrmann, J. Kho, B. Uçar, K. Kaya, and Ü. V. Çatalyürek, "Acyclic partitioning of large directed acyclic graphs," in *Proc. 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. ACM, 2017, pp. 371–380.
[19] A. Mizera, J. Pang, C. Su, and Q. Yuan, "ASSA-PBN: A toolbox for probabilistic boolean networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, no. 4, pp. 1203–1216, 2018.
[20] I. Shmulevich and E. R. Dougherty, *Probabilistic Boolean Networks: The Modeling and Control of Gene Regulatory Networks*. SIAM Press, 2010.
[21] B. Offermann, S. Knauer, A. Singh, M. L. Fernández-Cachón, M. Klose, S. Kowar, H. Busch, and M. Boerries, "Boolean modeling reveals the necessity of transcriptional regulation for bistability in PC12 cell differentiation," *Frontiers in Genetics*, vol. 7, p. 44, 2016.
[22] A. L. Barabasi and R. Albert, "Statistical mechanics of complex networks," *Reviews of Modern Physics*, vol. 74, pp. 47–94, 2002.
[23] T. P. Peixoto, "The graph-tool python library," *figshare*, 2014. [Online]. Available: http://figshare.com/articles/graph_tool/1164194
[24] A. Mizera, J. Pang, and Q. Yuan, "ASSA-PBN 2.0: A software tool for probabilistic Boolean networks," in *Proc. 14th International Conference on Computational Methods in Systems Biology*, ser. LNCS, vol. 9859. Springer, 2016, pp. 309–315.
[25] A. Mizera, J. Pang, H. Qu, and Q. Yuan, "ASSA-PBN 3.0: Analysing context-sensitive probabilistic Boolean networks," in *Proc. 16th International Conference on Computational Methods in Systems Biology*, ser. LNCS, vol. 11095. Springer-Verlag, 2018.
[26] A. Lomuscio, H. Qu, and F. Raimondi, "MCMAS: an open-source model checker for the verification of multi-agent systems," *International Journal on Software Tools for Technology Transfer*, vol. 19, no. 1, pp. 9–30, 2017.
[27] R. Schlatter, K. Schmich, I. A. Vizcarra, P. Scheurich, T. Sauter, C. Borner, M. Ederer, I. Merfort, and O. Sawodny, "ON/OFF and beyond - a Boolean model of apoptosis," *PLOS Computational Biology*, vol. 5, no. 12, p. e1000595, 2009.
[28] J. Saez-Rodriguez, L. Simeoni, J. A. Lindquist, R. Hemenway, U. Bommhardt, B. Arndt, U.-U. Haus, R. Weismantel, E. D. Gilles, S. Klamt *et al.*, "A logical model provides insights into T cell receptor signaling," *PLoS Computational Biology*, vol. 3, no. 8, p. e163, 2007.
[29] B. D. Conroy, T. A. Herek, T. D. Shew, M. Latner, J. J. Larson, L. Allen, P. H. Davis, T. Helikar, and C. E. Cutucache, "Design, assessment, and in vivo evaluation of a computational model illustrating the role of CAV1 in CD4$^+$ T-lymphocytes," *Frontiers in Immunology*, vol. 5, p. 599, 2014.
[30] S. Paul, C. Su, J. Pang, and A. Mizera, "A decomposition-based approach towards the control of Boolean networks," in *Proc. 9th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*. ACM, 2018, pp. 11–20.

Cui Su received her MSc degrees in Systems Engineering from Yanshan University in 2016. She is currently a PhD student at the Interdisciplinary Centre for Security, Reliability and Trust at the University of Luxembourg, working on the research project SEC-PBN. Her research interests lie in network control and computational systems biology.



Dr. Jun Pang received his PhD in Computer Science from Vrije Universiteit Amsterdam, The Netherlands in 2004. Currently, he is a senior researcher in the Computer Science and Communications research unit at the University of Luxembourg. His research interests include formal methods, security and privacy, social media mining, and computational systems biology.

Dr. Soumya Paul received his PhD in Theoretical Computer Science from the Institute of Mathematical Sciences, Chennai, India in 2012. Currently, he is a Research Associate at the Faculty of Science, Technology and Communications at the University of Luxembourg. His research interests include logic and strategic reasoning and computational systems biology.