Target Control of Boolean Networks with Permanent Edgetic Perturbations

Olivier Zeyen[1], Jun Pang[2]

Abstract— **Boolean network is a popular and well-established modelling framework for gene regulatory networks. The steady-state behaviour of Boolean networks can be described as attractors, which are hypothesised to characterise cellular phenotypes. In this work, we study the target control problem of Boolean networks, which has important applications for cellular reprogramming. More specifically, we want to reduce the total number of attractors of a Boolean network to a single target attractor. Different from existing approaches to solving control problems of Boolean networks with node perturbations, we aim to develop an approach utilising edgetic perturbations. Namely, our objective is to modify the update functions of a Boolean network such that there remains only one attractor. The design of our approach is inspired by Thomas' first rule, and we primarily focus on the removal of cycles in the interaction graph of a Boolean network. We further use results in the literature to only remove positive cycles which are responsible for the appearance of multiple attractors. We apply our solution to a number of real-life biological networks modelled as Boolean networks, and the experimental results demonstrate its efficacy and efficiency.**

I. Introduction

Complex diseases such as cancer and diabetes are challenging for modern medicine. To understand complex diseases and their progression, it is crucial to understand the mechanisms of cell transformation. Cellular reprogramming is a way to change one cell phenotype to another, allowing to reprogram any abundant cell into the desired cells [1], [2]. The identification of potential target genes and reprogramming paths remains a major challenge in *in vivo* cellular reprogramming [3]. Conventional experimental approaches are usually prohibited by the high complexity of biological systems and the high cost of experiments [4]. This leads to the need for developing efficient *in silico* approaches towards cellular reprogramming based on mathematical modelling.

In this work, we focus on the control of gene regulatory networks (GRNs) modelled as Boolean networks (BNs). GRNs are networks capturing the relationships between genes and their regulators in the cells, they represent biological systems characterised by the orchestrated interplay of such complex interactions resulting in highly nested feedback and feed-forward cycles. Among various mathematical modelling frameworks for GRNs, BNs have a distinct advantage [5], being simple and yet able to capture the important dynamical properties of the biological system under study, thus facilitating the modelling of large biological systems as a whole. In BNs, genes are modelled as binary variables, being either 'expressed' (value '1') or 'not expressed' (value '0') and activation/inhibition regulations between genes are described by Boolean update functions. The dynamics of a BN are determined by Boolean functions together with the update mode, either *synchronous* or *asynchronous*. The asynchronous updating scheme is considered more realistic than the synchronous one, since it randomly updates one node at each time step and therefore can capture different biological processes at different time scales [6]. The steady states of biological systems are described as *attractors* in BNs, to one of which the network eventually settles down. In biological context, attractors are hypothesised to characterise cellular phenotypes [5] and correspond to functional cellular states such as proliferation, apoptosis, differentiation, etc. [7].

In the context of BNs, cellular reprogramming, or the control of the GRNs, amounts to being able to drive the dynamics of the associated BN from an attractor to another 'desirable' target attractor by controlling or reprogramming the nodes of the BN. This has been recognised as the *source-target control* of BNs. The non-determinism of the asynchronous dynamics of BNs contributes to a better depiction of biological systems. As a consequence, it makes the control problem more challenging and renders the control methods designed for synchronous BNs inapplicable [8], [9]. Another major obstacle to BNs is the infamous state space explosion problem — the state space is exponential in the size of the network. It motivates the development of efficient solutions for the source-target control of BNs with instantaneous, temporary and permanent node perturbations [10], [11], [12], [13], [14]. The perturbation of a node means to change the expression of the node to either '1' or '0'. In view of the difficulties and expenses in conducting biological experiments, these node-perturbation based methods aim to compute the minimal control sets, which can be easily translated for wet-lab validation. In practice, cells in tissues and in culture normally exist as a population of cells, corresponding to different stable steady states [2]. This gives rise to the necessity of designing *target control* methods for BNs to compute a subset of nodes, the control of which can always drive the system from *any initial state* to a desired target attractor. The target control method developed in [15] adopts instantaneous node perturbations which are only applied instantaneously, but at a cost, a rather large number of control nodes are required when compared to temporary and permanent perturbations. In addition, it is difficult to guarantee that all the perturbations take effect at

[1]O. Zeyen is with the Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg.

[2]J. Pang is with the the Faculty of Science, Technology and Medicine and the Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg.

*Correspondence should be addressed to J. Pang (jun.pang@uni.lu).

the same time in biological experiments. Therefore, target control methods with temporary node perturbations were developed [16], [17], which compute a rather small set of node perturbations that solves the *target control* problem.

All the above control methods focus on identifying node perturbations in BNs, corresponding to 'knock-out' gene mutations in biological experiments. However, for complex diseases it is more common that several subtle changes cause the diseases. Mutations causing a molecular defect to a protein may lead to a distinct perturbation of a specific interaction [18], [19], [20], [21]. To account for this phenomena, novel predictive models of genotype-phenotype relationships are required. Towards this requirement, the conceptual model of 'edgetic perturbations' was proposed [18], [20], which corresponds to the *removal of an edge* in the original biological network. Control strategies using edgetic perturbations may pave the way for novel therapeutic strategies. However, there exist few works on identifying edgetic perturbations for cellular reprogramming in the context of BNs, where edgetic perturbations correspond to modifications of the Boolean update functions. An initial effort on identifying edgetic perturbations in BNs was made by Biane and Delaplace [22], but their method only works for Boolean control networks with the synchronous updating mode.

In this paper, we tackle the target control problem of asynchronous BNs with edgetic perturbations. We aim to identify a set of edgetic perturbations (preferably small) for a BN, such that when applied, the network eventually reaches a given target attractor independently of its initial state. We focus on perturbing the edges (Boolean functions) of a Boolean network instead of the nodes. This is equivalent to perturbing the interactions between nodes instead of directly perturbing the nodes themselves. Our approach to the target control problem is based on the work of Richard [23] which was originally inspired by Thomas' first rule [24]. Thomas conjectured that positive cycles in the interaction graph of a dynamical system is a necessary condition for multiple steady states. Following the result in [23], if all the local interaction graphs of a BN have no positive cycles then the network has at most one attractor. This is a generalisation of Thomas' first rule [24]. One naive approach would be to remove every positive cycle from the BN and thus there would only remain one attractor. However, this approach often produces larger solutions which might be infeasible or very costly for wet-lab validation. A refined approach is to try and reduce the set of positive cycles before removing them, because a positive cycle is a *necessary but not sufficient* condition for multiple attractors. In other words, multiple attractors mean multiple positive cycles but multiple positive cycles do not necessarily give rise to multiple attractors. This leads to the main idea of our approach that before removing the cycles we would filter the cycles to only focus on the relevant cycles. We then compute a feedback edge set (FES) to remove these relevant cycles. In this way, we can identify a smaller set of edgetic perturbations than the naive approach, ensuring that the BN, after applying the perturbations, will have only one attractor, i.e., the given target attractor.

We have implemented our approach and evaluated it through a number of real-life biological networks modelled as BNs. The experimental results show that our approach is quite effective — it identifies relatively small sets of edgetic perturbations to solve the target control problem of these networks, when compared to the number of edges in the networks. We also present the time to compute the solutions which also demonstrates the efficiency of our approach.

II. Preliminaries

A. Concepts for Boolean networks

Boolean network is a modelling framework for GRNs, which was first introduced by Kauffman [5] and later refined and improved by Thomas [25].

Definition 1 (Boolean network [25]): A Boolean network (BN) $G(V, \bar{f})$ consists of a set of nodes $V = \{v_1, \ldots, v_n\}$ and a vector of Boolean update functions $\bar{f} = (f_1, \ldots, f_n)$, where each $f_i : \{x_1, \ldots, x_n\} \to \{0, 1\}$ is an update function associated with node v_i and $x_j \in \{0, 1\}$ is a value assigned to node v_j, with $i, j \in \{1, \ldots, n\}$.

For the rest of the exposition, we assume that an arbitrary but fixed network BN $G(V, \bar{f})$ of n variables and update functions f_i $(1 \leq i \leq n)$ is given to us.

Definition 2 (State of a Boolean network): A state of a BN is given by a vector $\bar{x} = (x_1, \ldots, x_n) \in \{0, 1\}^n$, where $x_i \in \{0, 1\}$ is a value assigned to the node v_i. The state of the network at a discrete time point t $(t = 0, 1, \ldots)$ is given by a vector $\bar{x}(t) = (x_1(t), x_2(t), \ldots, x_n(t))$, where $x_i(t)$ is a value assigned to the node v_i at time point t. The value of a node at the next time step $t + 1$ is given by the function f_i, i.e., $x_i(t + 1) = f_i(\bar{x}(t))$.

A BN evolves in discrete time steps. It starts initially in a state and its state changes in every time step according to the Boolean update functions. According to how the updating may happen, BNs are classified into two types, i.e., *synchronous* and *asynchronous*. In the synchronous updating scheme, all the nodes are updated simultaneously. On the other hand, in the asynchronous updating scheme, at each time step a node in the network is picked randomly and updated according to its update function. In this work, we shall be interested primarily in the asynchronous updating scheme. The transition relation of an asynchronous BN is given by the following definition [23]:

Definition 3 (State transition system): For a given BN we define $\Gamma(\bar{f})$ as its state transition system. The state set is $\{0, 1\}^n$ and for every state $\bar{x}$ and every component i such that $f_i(\bar{x}) \neq x_i$, there is an arc $\bar{x} \to \bar{x}^i$, with $\bar{x}^i$ the state obtained by flipping the ith component in the state $\bar{x}$.

We write $(\bar{x}, \bar{y}) \in \Gamma(\bar{f})$ to denote an arc going from the state $\bar{x}$ to the state $\bar{y}$ which belongs to the state transition system $\Gamma(\bar{f})$. The state space of a BN consists of all the states in $\{0, 1\}^n$. The concept of trap domain captures a set of states that the network cannot leave.

Definition 4 (Trap domain [26]): A trap domain A of $\Gamma(\bar{f})$ is a non-empty subset of states such that $\forall(\bar{x}, \bar{y}) \in \Gamma(\bar{f}) \wedge \bar{x} \in A \implies \bar{y} \in A$.

The long-run behaviour of a BN is described by its steady states, which are formally known as attractors. The attractors of a BN are of particular interest due to their biological interpretation as, for instance, attractors are hypothesised to characterise cellular phenotypes [5].

Definition 5 (Attractor [27]): An attractor A of a BN is a smallest trap domain, i.e., a trap domain which does not include any other trap domains except for itself.

From the definitions, we can see that an attractor is a bottom strongly connected component in the state transition system of a BN.

Next, we define hyperrectangular regions in the state space of BNs. This definition allows us to classify the steady states of BNs (i.e., attractors) into two types.

Definition 6 (Hyperrectangular region [26]): Let $\bar{x}$ and $\bar{y}$ be two states of a BN. We define $\pi(\bar{x}, \bar{y})$ to be the smallest hyperrectangular region containing both $\bar{x}$ and $\bar{y}$ as follows:

$$\pi(\bar{x}, \bar{y}) = \prod_{i=1}^{n} \{min(x_i, y_i), \ldots, max(x_i, y_i)\}$$

We distinguish two types of attractors, simple cyclic attractors and complex cyclic attractors. Simple cyclic attractors are a type of attractors that can be expressed using a single hyperrectangular region $\pi(\bar{x}, \bar{y})$. Notice that fixed points are a special case in which $\bar{x} = \bar{y}$. Complex cyclic attractors are a type of attractors that if expressed using hyperrectangular regions, require at least two hyperrectangular regions $\bigcup_{i=1}^{n} \pi(\bar{x}_i, \bar{y}_i)$ (with $n \geq 2$) to be fully expressed without containing any other states that are not part of the attractor. In this paper, we will only consider the case in which the target attractor Δ is either a fixed point or a simple cyclic attractor.

For two states $\bar{x}$ and $\bar{y}$ of a BN, we define $I(\bar{x}, \bar{y})$ as the set of nodes such that for each node v_i if $x_i \neq y_i$ then $v_i \in I(\bar{x}, \bar{y})$ [26]. If either $\bar{x}$ or $\bar{y}$ is not a state but a trap domain, we choose the states such that the set $I(\bar{x}, \bar{y})$ is minimal.

Definition 7 (Discrete partial derivative [23]): The discrete partial derivative of f_i with respect to the component j is the function $f_i^j : \{0,1\}^n \to \{-1, 0, 1\}$ defined by:

$$\begin{aligned} f_i^j(\bar{x}) \quad := \quad & f_i((x_1, \ldots, x_{j-1}, 1, x_{j+1}, \ldots, x_n)) - \\ & f_i((x_1, \ldots, x_{j-1}, 0, x_{j+1}, \ldots, x_n)) \end{aligned}$$

Definition 8 (Local interaction graph): The local interaction graph of a BN with update functions $\bar{f}$ at state $\bar{x}$, denoted $G\,\bar{f}(\bar{x})$, is the signed directed graph with the vertex set $V = \{1, \ldots, n\}$ and for all $i, j \in V$ there is an edge from j to i of the same sign as $f_i^j(\bar{x})$ if $f_i^j(\bar{x})$ is nonzero.

The global interaction graph $G\,(\bar{f})$ is the union of all the local interaction graphs. We write $e = i \to j$ to denote an edge going from node i to node j in the interaction graph. We have two functions: src and dst, which return the source and destination of e, respectively. Namely, we have $\mathsf{src}(e) = i$ and $\mathsf{dst}(e) = j$. We use the function sign to return the sign of the edge: $\mathsf{sign}(e) \in \{-, +\}$.

B. Data structures

In this section, we define several notations and data structures that we use to present our approach. We define $\mathcal{C}$ as the set of cycles in the global interaction graph of a BN. The sign of a cycle can be obtained by computing the product of the signs of all the edges in the cycle. With this, we define $\mathcal{C}_+$ as the set of positive cycles. We only consider elementary cycles, and define $\mathcal{A}$ as the set of attractors of the BN. We define $\mathcal{E}$ as the set of edges and thus we can define a cycle as being a set of edges. The symbol $\leftrightarrow$ represents a relation. For example, if we write $\mathcal{C} \leftrightarrow \mathcal{P}(\mathcal{A})$, this is a relation between cycles and sets of attractors. In this way, we can create new types from existing types. We define a function $\mathsf{vertices}$ which takes a cycle as parameter and returns a set of vertices. If we use $\mathcal{V}$ for the set of vertices the function $\mathsf{vertices}$ is defined as $\mathsf{vertices} : \mathcal{P}(\mathcal{E}) \to \mathcal{P}(\mathcal{V})$.

We need to define how a BN is represented in syntax. The Boolean functions can be represented as a disjunction of conjunctions of positive and negative literals, i.e., the so-called disjunctive normal form (DNF) or *prime implicants*. We define $\mathcal{BN}$ as being a shorthand for the type of Boolean networks which can be written as $\mathcal{V} \to \mathcal{P}(\mathcal{P}(\mathcal{L}))$, with $\mathcal{L}$ the set of literals. We then use the $\mathsf{sign} : \mathcal{L} \to \{+, -\}$ function to get the signs of the literals and $\mathsf{vertex} : \mathcal{L} \to \mathcal{V}$ to get the associated vertex.

C. Edgetic perturbations

We now define edgetic perturbations following [28]. In this work, we perturb edges in the local interaction graphs instead of the global interaction graph. Edgetic perturbations involve setting an edge to either true or false. This means that from the target genes perspective, the source of the edge will always be seen as either true or false. For example, suppose the Boolean update function for node v_5 is $f = (x_1 \wedge x_2 \wedge x_3) \vee (x_2 \wedge x_3 \wedge x_4)$. We now decide to perturb the edge $v_1 \to v_5$, and set the edge to true. The function would become $f = (\mathit{true} \wedge x_2 \wedge x_3) \vee (x_2 \wedge x_3 \wedge x_4)$. If, on the other hand, we set the edge to false, we get $f = (\mathit{false} \wedge x_2 \wedge x_3) \vee (x_2 \wedge x_3 \wedge x_4) = x_2 \wedge x_3 \wedge x_4$. Moreover, if we want to perturb an edge coming from v_2, we would have two possible edgetic perturbations because x_2 has two occurrences in f. In this case, both occurrences are positive which means that setting both to true or false will have the effect we are looking for. In general, however, a literal could appear both positive and negative in an update function and thus would be set to different values. However, this is not the case in the context of biological networks where we consider that a node can only have one type of effect on other nodes.

D. Target control of BNs

We formulate our research problem as follows.

Definition 9 (Target control of BN): Given a BN, the target control problem aims at perturbing some parameters (i.e., nodes or Boolean update functions) of the network so that from any state in the network, it will eventually reach a given target attractor Δ of the original BN.

In our case, the solution is going to be a set of permanent edgetic perturbations such that there remains only a single attractor in the BN, and the remaining attractor would be an attractor that we are given at the beginning. Moreover, we aim to use *as few perturbations as possible* because applying edgetic perturbations in wet-lab experiments is often complicated and costly, and we want to perturb the networks as little as possible.

A running example. We have an example BN_{ex} with four nodes $\{v_1, v_2, v_3, v_4\}$. The corresponding Boolean update functions are given below:

$$f_1 = x_1 \vee x_2; \quad f_2 = x_1 \wedge x_4;$$
$$f_3 = \neg x_1 \wedge x_4; \quad f_4 = x_3.$$

The update functions are already in the required DNF form.

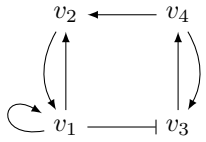

Fig. 1. The global interaction graph for BN_{ex}.

With these functions we can derive the state transition system. From the state transition system and following Definition 5, we find that BN_{ex} has three attractors, namely 0000, 0011 and 1000. Figure 1 shows the global interaction graph of BN_{ex}. The edges with an arrow head in a 'T' shape are negative edges (e.g., the edge from x_1 to x_3).

III. Our Approach

We first present results from the literature on which our solution is based. We continue by giving an overview of our approach and explain the details of how each component of our approach works.

By Thomas' first rule [24], we know that if $G(\bar{f})$ has no positive cycles, then $\Gamma(\bar{f})$ has at most one fixed point. Theorem 1 below gives us a local version of Thomas' first rule for local interaction graphs.

Theorem 1: (Thomas' first rule, local version [23]). If $G\,\bar{f}(\bar{x})$ has no positive cycle for every $\bar{x} \in \{0,1\}^n$ then $\Gamma(\bar{f})$ has a unique attractor.

With Theorem 1, we can see that if we simply remove all the edges from the feedback edge set computed on the subgraph of the global interaction graph containing only the positive cycles, then there would only remain one attractor. This would still remove too many edges because a positive cycle in a local interaction graph is a necessary but not a sufficient condition for the presence of two disjoint trap domains [26]. This is why the local interaction graphs may contain more cycles than necessary. Moreover, when computing the union of all the local interaction graphs, new cycles may appear. Therefore, if we focus on the global interaction graph instead, the algorithm will return more edgetic perturbations. The efficiency gains would be substantial and we thus use the global interaction graph.

Procedure target-control(BN, Δ)

Input: BN $\in \mathcal{BN}$, $\Delta \in \mathcal{A}$
Output: $\mathcal{BN}$
begin
  $attrs \leftarrow$ compute-attractors(BN)
  while $|attrs| \neq 1$ **do**
    $G(\bar{f}) \leftarrow$ interaction-graph(BN)
    $c_+ \leftarrow$ positive-cycles$(G(\bar{f}))$
    $C \leftarrow$ filter-cycles$(c_+, \Delta, attrs \setminus \Delta)$
    $FES \leftarrow$ filter-edges(C)
    $BN \leftarrow$ perturb-BN(BN, FES)
    $attrs \leftarrow$ compute-attractors(BN)
  end
  return *BN*
end

Richard and Commet [26] give us further details on the nodes involved in the positive cycle.

Theorem 2: (Thomas' conjecture, discrete version [26]). Suppose that A and B are two disjoint trap domains of $\Gamma(\bar{f})$. Let $(\bar{x}, \bar{y}) \in A \times B$ such that for all $(\bar{x'}, \bar{y'}) \in A \times B.\pi(\bar{x'}, \bar{y'}) \not\subset \pi(\bar{x}, \bar{y})$. Then there exists $z \in \pi(\bar{x}, \bar{y})$ such that $G\bar{f}(z)$ has a positive cycle.

Richard and Commet [26] show that the positive cycle responsible for the presence of two disjoint trap domains A and B only contains nodes in which $\bar{x}$ and $\bar{y}$ are different (i.e., $I(\bar{x}, \bar{y})$), with $\bar{x} \in A$ and $\bar{y} \in B$ (see Theorem 2).

To apply Theorem 2 on two attractors A and B we need to pick two states $\bar{x} \in A$ and $\bar{y} \in B$ such that the hyperrectangular region is as small as possible. We know that the positive cycle responsible for the presence of A and B uses a subset of the nodes in $I(\bar{x}, \bar{y})$ and we can filter the cycles accordingly. Theorem 2 thus allows us to focus on searching relevant positive cycles and to identify which edgetic perturbations are required. However, we cannot ensure that the interaction graph of the resulting BN is acyclic and thus ensure that the BN has only one attractor left. Therefore, we now need to repeat the detection and removal of cycles until there is indeed only one single attractor left.

Our final approach is now captured by the overall procedure target-control. The procedure simply starts by computing the set of attractors of a BN. The procedure then computes the global interaction graph $G(\bar{f})$ for the BN. Using the graph $G(\bar{f})$, it computes the set of positive cycles c_+ which is then reduced using Theorem 2 to identify only relevant cycles for the appearance of multiple attractors. This is achieved with the procedure filter-cycles. We then proceed to invoke filter-edges to obtain at least one FES. We then simply perform the necessary perturbations. If there exist multiple solutions to solve the target control of the BN with respect to Δ, the procedure perturb-BN just randomly picks one of the solutions to apply.[1] This is repeated until there is

[1]It is straightforward to update the procedure if it is desired to have all the solutions.

only a single attractor left. The re-computation of attractors is necessary because perturbing the edges permanently may create new attractors.

To compute the set of elementary cycles of a graph we refer the readers to the work of Johnson [29]. We then simply compute the signs of the cycles by computing the product of the signs of the edge, and remove all the negative cycles from the set (i.e., positive-cycles).

Continuing with our example BN_{ex} in Section II, from its global interaction graph depicted in Figure 1, we can find the set of positive cycles: $\mathcal{C}_+ = \{(v_1, v_1)\}, \{(v_1, v_2), (v_2, v_1)\}, \{(v_3, v_4), (v_4, v_3)\}\}$.

The next step is to apply the procedure filter-cycles to filter out all the cycles that are considered irrelevant following Theorem 2. This procedure takes as input a set of positive cycles C, a target attractor Δ and a set of attractors Ψ to remove from the network. Thanks to Theorem 2, we know that all nodes involved in a positive cycle c need to be in a set $I(\bar{x}, \bar{y})$ with $\bar{x} \in \Delta$ and $\bar{y} \in a \in \Psi$. We can thus filter the set of cycles. To remove the edges afterwards, we need some context to know exactly what the necessary edgetic perturbations are, i.e., how to modify the update functions so that the only remaining attractor is Δ. Thus the procedure filter-cycles returns a map from the cycles to sets of attractors. We obtain the relation $\mathcal{C}_+ \leftrightarrow \mathcal{P}(\mathcal{A})$ such that for every pair (c, ϕ) in the relation we have $\forall a \in \phi, \mathsf{vertices}(c) \subseteq I(a, \Delta)$, with $c \in \mathcal{C}_+$ and $\phi \subseteq \mathcal{A}$.

For our running example BN_{ex}, its attractors are 0000, 0011 and 1000. Let us choose the target attractor Δ as 1000, so we have $\Psi = \{0000, 0011\}$. Now we compute $I(\Delta, 0000) = \{v_1\}$ and $I(\Delta, 0011) = \{v_1, v_3, v_4\}$. We thus obtain: $result = \{\{(v_1, v_1)\} \mapsto \{0000, 0011\}, \{(v_3, v_4), (v_4, v_3)\} \mapsto \{0011\}\}$

We can see that the attractor 0011 is mapped to two cycles instead of just one. The reason for this is because we keep all the cycles that are in the subgraph defined by the set of nodes $I(\Delta, 0011)$. This subgraph contains the two cycles. The comparison $I(\Delta, 0000)$ returns a single node, thus the only possibility is the self loop on node v_1.

Following the relation returned by the procedure filter-cycles, we would need to compute a FES on the cycles in the relation. We also want to keep the relation between the identified edges and the attractors. To reduce the number of possibilities, the procedure filter-edges simply looks for overlap between the cycles. If there is overlap between two cycles in the relation, we merge the relations by computing the intersection of the cycles and the union of the sets of attractors. This allows us to reduce the total number of perturbations. The result contains multiple solutions.

In our example BN_{ex}, there is no overlap between the cycles that we aim to remove, so the procedure filter-edges would simply return its input. For our example, the sets $\{(v_1, v_1), (v_3, v_4)\}$ and $\{(v_1, v_1), (v_4, v_3)\}$ are valid FES both returned by our procedure.

We are now ready to compute the required edgetic perturbations in the BN as described in the procedure perturb-BN.

The procedure perturb-BN (randomly) picks one FES,

Procedure perturb-BN(FES, Δ, BN)

Input: FES $\in \mathcal{P}(\mathcal{E}) \leftrightarrow \mathcal{P}(\mathcal{A})$, $\Delta \in \mathcal{A}$, BN $\in \mathcal{BN}$
Output: $\mathcal{BN}$
begin
  foreach *s in dom(FES)* **do**
    $e \leftarrow$ pick one edge from s
    if Δ *has the gene dst(e) set to true* **then**
      clauses $\leftarrow$ pick the clauses for dst(e) containing src(e)
      remove src(e) from the *clauses*
    else
      foreach *a in FES(s)* **do**
        clauses $\leftarrow$ pick the clauses for dst(e) satisfied by a containing src(e)
        remove *clauses* from the network for gene dst(e)
      end
    end
  end
  return *BN*
end

then for each edge e in the FES, the procedure first checks if the target of e (dst(e)) is true in Δ. Because of the way we filter the cycles, we know that if a node is true in Δ, it is false in all the attractors that we aim to remove (and similarly if a node is false in Δ). We have established that dst(e) is true in Δ. Now we remove all the occurrences of src(e) from node dst(e)'s update function except the ones that correspond to negative self loops. If we have established that dst(e) is false in Δ, then we have to go through all the attractors associated with the edge e, namely the ones in the set $FES(e)$. For each attractor $a \in FES(e)$, we select the conjunctive clauses of the function associated with node dst(e) which are satisfied in a and contain the node src(e). Since we want to make the node dst(e) false, we want to set the node src(e) to true when it is negative and false when it is positive. Notice that this is the absorbing element of the conjunction, thus the complete conjunction becomes false. In both cases we need to take care not to directly remove any negative self loops as removing them may also modify cyclic attractors while not contributing to our goal. Negative self loops may be perturbed as a side effect from other perturbations.

Let us continue with our example BN_{ex}. The first edge we need to remove is the edge (v_1, v_1). Our target attractor being $\Delta = 1000$, we have the destination of our edge set to true. The function we will modify is the function $f_1 = x_1 \vee x_2$. The clause x_2 is not satisfied by our target Δ and it does not contain the source of our edge, namely v_1. Let us now consider the clause x_1. We see that x_1 is satisfied by Δ and it contains v_1. Following our procedure perturb-BN, we will remove x_1 making the clause x_1 empty. We get $f_1 = true \vee x_2 = true$. There is still one edge to be removed following our procedure, but for our example, if we recompute the attractors after performing the first identified perturbation,

we notice that there is only a single attractor left, namely Δ. We can thus observe that our approach does not always return the smallest number of perturbations. This indicates that our approach *cannot* return a set of perturbations of minimal size.

Since the attractors of the BN need to be recomputed at each run, we have the following theorem stating the complexity of our procedure target-control.

Theorem 3: In the worst case target-control terminates in exponential time.

When performing edgetic perturbations we are only making it easier for the target attractor Δ to be present. Moreover, for the case in which Δ is a simple cyclic attractor, notice that we only modify the nodes in which Δ is different from the other attractors, and thus we only modify the nodes which are fixed in Δ. We cannot make such guarantees in case of complex cyclic attractors.

Theorem 4: Under the assumption that the target attractor Δ is a simple cyclic attractor or a fixed point, the procedure target-control is a sound solution to the target control problem of a BN.

However, as explained previously, the solutions returned by target-control are not guaranteed to be of minimal size.

IV. Evaluation

In this section, we present an evaluation of the proposed approach to solve the target control of BNs. Table I (left part) gives general information about the tested networks. The first column is the name of the network, the second column gives the number of nodes in the network and the third column gives the number of edges in the unsigned, directed global interaction graph. The columns '#CA' and '#CX' give the number of simple cyclic attractors (which includes fixed points) and complex cyclic attractors in the networks, respectively.

For most of the networks, their detailed description can be found in [30]. The other networks have been taken from the repository for the project PyBoolNet [31]. The PyBoolNet [31] and the software tool CABEAN [32] have been used to implement and test our approach. The PyBoolNet tool being written in Python and CABEAN in C, another library rendering CABEAN accessible through Python [33] has been used as well. For our experiments, the networks are first loaded using the PyBoolNet library, then we compute the attractors using the CABEAN tool. For each attractor of a network, we set it as the target and compute the corresponding FES. We then test every FES returned by the procedure filter-edges. Once an FES is applied, the attractors are computed using the CABEAN tool to check how many attractors are left and if the target attractor still is an attractor of the network. All the computations were done on a desktop computer with 16 GB of RAM and a quad core Intel i5-4460 processor running at 3.2 GHz.

Table I (right part) presents the experimental results on the networks. The first and second columns of the right part indicate the minimum and maximum number of edgetic perturbations identified across all the possible target attractors and across all the FESs returned by filter-edges. Every modification of a clause (i.e. setting a literal to either true or false) is counted as a perturbation. The columns min and max time indicate the minimum and maximum time in seconds required to compute the results. The computation time includes the time required to compute the attractors of the network once, the time required to test all the FESs and the time required to compute the attractors of all the networks after the FESs have been removed. The computation time however does not include the time required to load the Boolean network using the PyBoolNet [31] library. The last column indicates the maximum number of FESs returned by the procedure filter-edges.

From Table I, we see that the number of edgetic perturbations (at least the minimum ones) required to solve the target control of real-life BNs is relatively small when compared with the number of edges in the networks. The computation time also looks promising. If we take the largest network T-diff as an example, we see that the number of perturbations are between 7 and 12 across all attractors taken as the targets. The time required to test one attractor is between 63 seconds and 635.7 seconds even though for some attractors the procedure had to test 216 FESs. The network that required the longest computation time, is the T-LGL network, which requires at most 918.8 seconds for one attractor to be tested as the target. These results demonstrate that our approach scales reasonably well.

It is worth comparing the results of our approach with the method to solve target control of BNs with node perturbations. Table II (left part) shows the minimum and maximum number of perturbations required to achieve target control using permanent node perturbations implemented in CABEAN [32]. Table II (right part) shows the ratios obtained by dividing the maximum number of perturbations by the number of nodes or edges depending if the perturbations are node perturbations or edgetic perturbations. The dashes simply indicate that we did not manage to compute the node perturbations within one hour with CABEAN. From Table II, we see that, in most cases except for the hema network, the ratio for edgetic perturbations is smaller than the ratio for node perturbations.

In our experiments, we observe that after a single iteration of the loop in the procedure target-control, most networks have a single attractor left except for the networks dinwoodie life and hema which sometimes have two attractors left after the first iteration. We can thus see that in most cases the loop in the procedure target-control is not necessary. The target attractor has never been removed by our procedure except for the network dahlhaus neuroplastoma where the target was a complex cyclic attractors.[2]

V. Related Work

A lot of research has been devoted to developing control methods for complex networks including biolgoical networks

[2]This is an example explaining why we focus on taking a simple cycle or a fixed point as the target attractor.

networks	#nodes	#edges	#CA	#CX	min #P	max #P	min time	max time	max #FES
davidich yeast	10	28	13	0	3	13	0.1	0.2	1
faure cellcycle	10	35	1	1	3	3	0.0	0.0	1
myeloid	11	30	6	0	6	9	0.1	1.5	12
dinwoodie life	15	35	7	0	4	6	1.7	3.1	32
cardiac	15	38	3	0	3	5	0.1	0.1	2
dahlhaus neuroplastoma	23	47	16	16	7	8	3.1	10.3	4
tumour	32	158	9	0	5	33	3.6	639.5	8
hema	33	86	5	0	6	14	8.2	59.5	64
mapk	53	103	12	0	4	7	10.7	11.3	4
T-LGL	60	142	3	0	4	8	58.2	918.8	40
T-diff	68	151	12	0	7	12	63.0	635.7	216

TABLE I

GENERAL INFORMATION ABOUT THE NETWORKS WITH OUR MAIN EXPERIMENTAL RESULTS.

	min #P (node)	max #P (node)	#max-node-p/#nodes	#max-edge-p/#edges
davidich yeast	1	5	0.5	0.464
faure cellcycle	1	1	0.1	0.086
myeloid	2	4	0.364	0.3
dinwoodie life	2	4	0.267	0.171
cardiac	1	3	0.2	0.132
dahlhaus neuroplastoma	-	-	-	0.17
tumour	-	-	-	0.209
hema	2	4	0.121	0.163
mapk	3	4	0.075	0.068
T-LGL	-	-	-	0.056
T-diff	-	-	-	0.079

TABLE II

RESULTS OF THE NODE PERTURBATIONS COMPUTED WITH THE TOOL CABEAN [32], TOGETHER WITH THE RATIOS THAT COMPARE THE MAXIMUM NUMBER OF PERTURBATIONS TO THE NUMBER OF EDGES OR NODES DEPENDING IF THE PERTURBATIONS WERE NODE OR EDGETIC PERTURBATIONS.

(GRNs in particular). In this section, we review some of the related work, i.e., methods developed for the control of BNs, which are used as models of GRNs.

BNs are a widely accepted modelling framework for GRNs. Many research projects have been conducted for developing control methods for BNs with the synchronous updating scheme [34], [35], [36], [37], [38], [39]. However, all of these methods are not applicable to asynchronous BNs, which are considered to be more realistic to capture the dynamics of GRNs. Developing efficient and scalable control methods is considered more difficult for asynchronous BNs, mainly because of the non-deterministic behaviours in their state transition systems introduced by the asynchronous updating schemes. To cope with this challenge, new control methods employing the 'divide and conquer' strategy to explore both the structural and dynamical properties of a BN are proposed to identify key network nodes that can drive the the BN into a desired attractor. The methods proposed in [10], [11], [12], [13], [14] aim to solve the source-target control problem for BNs with instantaneous, temporary and permanent node perturbations in which both the source and target attractors are given, while the methods proposed in [15], [17] solve the target control problem for BNs in which only the target attractor is specified. The 'stable motifs' based method [16] also solves the target control problems for BNs by taking both network structure and functional information, which proves to be very efficient when dealing with large BNs.

All the above discussed methods for asynchronous BNs identify node perturbations for the corresponding control problems. Efficient methods for identifying edgetic perturbations for controlling BNs are still lacking in the literature. There exist a few works studying the impact of function perturbations [40], [41], [42] on BNs, and the one mostly related to ours is achieved by Campbell and Albert [28]. Their method computes a set of edgetic perturbations for the purpose of eliminating a specific fixed-point attractor, while our approach solves the target control problem of BNs by finding edgetic perturbations in order to make the perturbed BN only have one specified target attractor.

VI. DISCUSSION AND CONCLUSION

In this work, we have shown how to compute edgetic perturbations to solve the target control problem of asynchronous BNs. Our approach is not optimal (in terms of the number of computed perturbations) and may still be improved, even though the experimental results on a number of BNs seem encouraging – it is effective in terms of the relatively small number of identified edgetic perturbations; meanwhile, the small amount of time used to compute the solutions demonstrates its efficiency.

Through the experimental results, we noticed that the identified edgetic perturbations are very close to the node perturbations to solve a same target control problem. There may be an efficient way to further reduce the number of edgetic perturbations by integrating the control methods on node perturbations and our current approach.

REFERENCES

[1] T. Graf and T. Enver, "Forcing cells to change lineages," *Nature*, vol. 462, no. 7273, pp. 587–594, 2009.

[2] A. d. Sol and N. J. Buckley, "Concise review: A population shift view of cellular reprogramming," *Stem Cells*, vol. 32, no. 6, pp. 1367–1372, 2014.

[3] D. Srivastava and N. DeWitt, "In vivo cellular reprogramming: the next generation," *Cell*, vol. 166, no. 6, pp. 1386–1396, 2016.

[4] L.-Z. Wang, R.-Q. Su, Z.-G. Huang, X. Wang, W.-X. Wang, C. Grebogi, and Y.-C. Lai, "A geometrical approach to control and controllability of nonlinear dynamical networks," *Nature Communications*, vol. 7, p. 11323, 2016.

[5] S. A. Kauffman, "Homeostasis and differentiation in random genetic control networks," *Nature*, vol. 224, pp. 177–178, 1969.

[6] J. A. Papin, T. Hunter, B. O. Palsson, and S. Subramaniam, "Reconstruction of cellular signalling networks and analysis of their properties," *Nature Reviews Molecular Cell Biology*, vol. 6, no. 2, p. 99, 2005.

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

[8] J. Kim, S.-M. Park, and K.-H. Cho, "Discovery of a kernel for controlling biomolecular regulatory networks," *Scientific Reports*, vol. 3, no. 2223, 2013.

[9] 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.

[10] C. Su, S. Paul, and J. Pang, "Controlling large boolean networks with temporary and permanent perturbations," in *Proceedings of the 23rd International Symposium on Formal Methods*, ser. Lecture Notes in Computer Science, vol. 11800. Springer-Verlag, 2019, pp. 707–724.

[11] C. Su and J. Pang, "Sequential temporary and permanent control of Boolean networks," in *Proceedings of the 18th International Conference on Computational Methods in Systems Biology*, ser. Lecture Notes in Computer Science, vol. 12314. Springer-Verlag, 2020, pp. 234–251.

[12] H. Mandon, C. Su, J. Pang, S. Paul, S. Haar, and L. Paulevé, "Algorithms for the sequential reprogramming of boolean networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 16, pp. 1610–1619, 2019.

[13] H. Mandon, C. Su, S. Haar, J. Pang, and L. Paulevé, "Sequential reprogramming of Boolean networks made practical," in *Proceedings of the 17th International Conference on Computational Methods in Systems Biology*, ser. Lecture Notes in Computer Science, vol. 11773. Springer, 2019, pp. 3–19.

[14] S. Paul, C. Su, J. Pang, and A. Mizera, "An efficient approach towards the source-target control of Boolean networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 17, pp. 1932–1945, 2020.

[15] A. Baudin, S. Paul, C. Su, and J. Pang, "Controlling large boolean networks with single-step perturbations," *Bioinformatics*, vol. 35, pp. i558 – i567, 2019.

[16] J. G. T. Zañudo and R. Albert, "Cell fate reprogramming by control of intracellular network dynamics," *PLoS Computational Biology*, vol. 11, 2015.

[17] C. Su and J. Pang, "A dynamics-based approach for the target control of Boolean networks," in *Proceedings of the 11th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*. ACM Press, 2020, pp. 50:1–50:8.

[18] Q. Zhong, N. Simonis, Q.-R. Li, B. Charloteaux, F. Heuze, N. Klitgord, S. Tam, H. Yu, K. Venkatesan, D. Mou, V. Swearingen, M. A. Yildirim, H. Yan, A. Dricot, D. Szeto, C. Lin, T. Hao, C. Fan, S. Milstein, D. Dupuy, R. Brasseur, D. E. Hill, M. E. Cusick, and M. Vidal, "Edgetic perturbation models of human inherited disorders," *Molecular Systems Biology*, vol. 5, p. 321, 2009.

[19] M. Dreze, B. Charloteaux, S. Milstein, P.-O. Vidalain, M. A. Yildirim, Q. Zhong, N. Svrzikapa, V. Romero, G. Laloux, R. Brasseur, J. Vandenhaute, M. Boxem, M. E. Cusick, D. E. Hill, and M. Vidal, "Edgetic perturbation of a c. elegans bcl2 ortholog," *Nature Methods*, 2009.

[20] B. Charloteaux, Q. Zhong, M. Dreze, M. Cusick, D. Hill, and M. Vidal, "Protein-protein interactions and networks: forward and reverse edgetics," *Methods in Molecular Biology*, vol. 759, pp. 197–213, 2011.

[21] J.-P. Lambert, G. Ivosev, A. L. Couzens, B. G. Larsen, M. Taipale, Z. yuan Lin, Q. Zhong, S. Lindquist, M. Vidal, R. Aebersold, T. Pawson, R. Bonner, S. Tate, and A. Gingras, "Mapping differential interactomes by affinity purification coupled with data independent mass spectrometry acquisition," *Nature Methods*, vol. 10, pp. 1239 – 1245, 2013.

[22] C. Biane and F. Delaplace, "Abduction based drug target discovery using Boolean control network," in *Proceedings of the 15th International Conference on Computational Methods in Systems Biology*, ser. Lecture Notes in Computer Science, vol. 10545. Springer, 2017, pp. 57–73.

[23] A. Richard, "Positive and negative cycles in Boolean networks," *Journal of Theoretical Biology*, vol. 463, pp. 67–76, 2019.

[24] R. Thomas, "On the relation between the logical structure of systems and their ability to generate multiple steady states or sustained oscillations," in *Numerical Methods in the Study of Critical Phenomena*. Springer-Verlage, 1981, pp. 180–193.

[25] ——, "Boolean formalization of genetic control circuits," *Journal of Theoretical Biology*, vol. 42, no. 3, pp. 563–585, 1973.

[26] A. Richard and J.-P. Comet, "Necessary conditions for multistationarity in discrete dynamical systems," *Discrete Applied Mathematics*, vol. 155, pp. 2403–2413, 2007.

[27] 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.

[28] C. Campbell and R. Albert, "Edgetic perturbations to eliminate fixed-point attractors in Boolean regulatory networks," *Chaos*, vol. 29, p. 023130, 2019.

[29] D. B. Johnson, "Finding all the elementary circuits of a directed graph," *SIAM Journal on Computing*, vol. 4, pp. 77–84, 1975.

[30] C. Su, S. Paul, and J. Pang, "Scalable control of asynchronous Boolean networks," in *Proceedings of the 17th International Conference on Computational Methods in Systems Biology*, ser. Lecture Notes in Computer Science, vol. 11773. Springer-Verlag, 2019, pp. 364–367.

[31] H. Klarner, "Pyboolnet," https://github.com/hklarner/PyBoolNet, accessed on 2021-03-16.

[32] C. Su and J. Pang, "CABEAN: A software for the control of asynchronous Boolean networks," *Bioinformatics*, vol. 37, no. 6, pp. 879–881, 2021.

[33] L. Paulevé, "CABEAN-python," https://github.com/algorecell/cabean-python#cabean-python, accessed on 2021-03-16.

[34] J. Liang, H. Chen, and J. Lam, "An improved criterion for controllability of Boolean control networks," *IEEE Transactions on Automatic Control*, vol. 62, pp. 6012–6018, 2017.

[35] H. Zhou, J. Su, X. Hu, C. Zhou, H. Li, Z. Chen, Q. Xiao, B. Wang, W. Wu, Y. Sun, Y. Zhou, C. Tang, F. Liu, L. Wang, C. Feng, M. Liu, S. Li, Y. Zhang, H. Xu, H. Yao, L. Shi, and H. Yang, "Glia-to-Neuron Conversion by CRISPR-CasRx Alleviates Symptoms of Neurological Disease in Mice," *Cell*, vol. 181, no. 3, pp. 590–603, 2020.

[36] J. Lu, J. Zhong, D. W. Ho, Y. Tang, and J. Cao, "On controllability of delayed Boolean control networks," *SIAM Journal on Control and Optimization*, vol. 54, no. 2, pp. 475–494, 2016.

[37] J. Zhong, Y. Liu, K. Kou, L. Sun, and J. Cao, "On the ensemble controllability of boolean control networks using stp method," *Applied Mathematics and Computation*, vol. 358, pp. 51–62, 2019.

[38] Y. Wu, X. Sun, X. Zhao, and T. Shen, "Optimal control of boolean control networks with average cost: A policy iteration approach," *Automatica*, vol. 100, pp. 378–387, 2019.

[39] H. Chen, L. Albergante, J. Y. Hsu, C. Lareau, G. L. Bosco, J. Guan, S. Zhou, A. N. Gorban, D. E. Bauer, M. Aryee, D. Langenau, A. Zinovyev, J. Buenrostro, G. Yuan, and L. Pinello, "Single-cell trajectories reconstruction, exploration and mapping of omics data with stream," *Nature Communications*, vol. 10, 2019.

[40] M. Meng and J. Feng, "Function perturbations in boolean networks with its application in a D. melanogaster gene network," *European Journal of Control*, vol. 20, pp. 87–94, 2014.

[41] X. Qian and E. R. Dougherty, "Effect of function perturbation on the steady-state distribution of genetic regulatory networks: Optimal structural intervention," *IEEE Transactions on Signal Processing*, vol. 56, pp. 4966–4976, 2008.

[42] X. Li, H. Li, Y. Li, and X. Yang, "Function perturbation impact on stability in distribution of probabilistic boolean networks," *Mathematics and Computers in Simulation*, vol. 177, pp. 1–12, 2020.