Sequential reprogramming of biological network fate

Jérémie Pardo, Sergiu Ivanov, Franck Delaplace

Sequential Reprogramming of Biological Network Fate

Jérémie Pardo, Sergiu Ivanov, and Franck Delaplace

IBISC, Univ Évry, Paris-Saclay University, 91025, Évry, France
{jeremie.pardo,sergiu.ivanov,franck.delaplace}@ibisc.univ-evry.fr
http://www.ibisc.univ-evry.fr

Abstract. Network controllability is a major challenge in network medicine. The problem is to rewire the molecular network for reprogramming the cell fate. The reprogramming action is considered as a control usually performed once. However, in some cases, a therapy has to follow a time-scheduled drug administration protocol. Furthermore, some diseases are induced by a sequence of mutations leading to a sequence of actions on molecules. In this paper, we extend the single control action method by investigating the sequential control of Boolean networks. We present a novel theoretical framework for formal study of control sequences, leading to algorithms resolving the PSPACE-hard problem of inferring minimal parsimonious control sequences under the synchronous dynamics.

Keywords: dynamical systems reprogramming · Boolean control network · control sequence · abductive reasoning · drug target prediction · sequential therapy.

1 Introduction

Biological network control consists in modifying gene expression to reprogram cells naturally or artificially. Applications of network controllability will have valuable benefits in essential challenges of health: cancerous targeted therapy, complex disease etiology, regenerative medicine, stem cells monitoring, etc. In spite of the impressive progress in cell reprogramming during the last decade, more breakthroughs are required before cellular reprogramming yields routine clinical use [27]. The main issues lie in the discovery of reliable ways to trigger the reprogramming process and to understand exactly how the process works. In this endeavour, the definition of suitable theoretical frameworks and computational methods are crucial for enabling the analysis and the design of *control patterns* responsible for the phenotypic switch.

Cell reprogramming mechanisms are based on the control of molecular processes to monitor the dynamics of the network fate. In [29], the authors relate mutations to their network effect: nonsense mutation[1], out-of-frame insertion or

[1] Substitution of a single base pair leading to the appearance of a stop codon where previously there was a codon specifying an amino acid.

deletion and defective splicing are interpreted as node or arc deletions whereas missense mutation[2] and in–frame insertion or deletion can be modelled as node or arc additions. Moreover, in [9], the authors classify mutations according to the way they affect signalling networks and distinguish mutations that constitutively activate or inhibit enzymes (nodes) and mutations that rewire the interactions (arcs). Accordingly, the action of targeted therapies is interpreted as network rewiring [10]. The effect of mutations and drugs can thus be described as elementary topological actions on the network: deletion or insertion of nodes and arcs. Network control is based on these topological actions. The impact of the network actions should be evaluated from a model of dynamics translating the topological actions into dynamical change of the trajectories. Accordingly, the phenotypic changes are assessed at molecular level by the measurement of the state of peculiar molecules called *biomarkers*—observable indicators of biological processes whose molecular signature variation discriminates the phenotypes [10, 25]. The signatures must be observed in a significant period of time for testifying their relevance, and thus assumed to be met at stability condition of the biological system. This approach is part of Network Medicine [3] which is an emerging field of precision medicine aiming to address drug target discovery and the elucidation of disease mechanism based on network analysis by considering the the phenotype-genotype relationship as an association of phenotype and network perturbations [24].

Recent research in computational biology provides novel inference methods for reprogramming a system to make the dynamics converge towards an expected fate. These works use the *Boolean Control Network* (BCN) specifying the controls on Boolean network (Section 2). In [15], the authors apply a stuck-at fault model for simulating drug and disease processes. A Max-SAT based algorithm is then used for inferring node actions. In [16], the authors study the trajectory and state controllability of delayed BCNs. Using the semi-tensor product of matrices, the delayed BCNs are converted into an equivalent algebraic description from which are derived the necessary and sufficient conditions for the trajectory controllability of delayed BCNs. In [28], the authors propose a heuristic method focused on the control of key-nodes stabilizing the state of "motifs" that correspond to specific sub-networks. In [13], the authors define and give a general algorithm to find a minimal set of network components named "control kernels" that must be regulated to make the cell reach a desired stable state. In [23, 20], the authors study the problem of computing a minimal subset of nodes of a given asynchronous [23] and synchronous [20] Boolean network that need to be controlled to drive its dynamics from an initial steady state to a target steady state. The general idea of the proposed algorithms is a decomposition-based solution to the minimal control problem. In [26], the authors extend [23] by including a notion of temporary and permanent perturbations. In [21], the authors propose a method based on Gröbner basis computation to find the node actions for generating or avoiding particular stable states. In [4, 5], the authors use an

[2] Change of a single base pair causing the substitution of a different amino acid in the resulting protein.

abductive method based on prime implicants for the inference, and cover actions on nodes and arcs. These works were validated on real biological cases showing their adequacy for drug therapy prediction. The first method is restricted to acyclic networks whereas the others admit any kind of networks. In summary, the state of the art related to Boolean control network shows that the majority of the works use a similar methodology consisting in computing a single network action modelled as control input for monitoring the dynamics in order to reach stable states meeting some expected properties assessed at molecular level.

However in some biological cases, a sequence of mutations is observed or a therapy involves a scheduled protocol for administering drugs. Tumorigenesis is the result of a multi-step process governed by sequential genetic alterations. Colorectal tumors can be considered as a paradigmatic example illustrating this sequential progression shifting from a benign tumor (adenoma) to a malignant one (carcinoma) following a sequence of four gene mutations ending with the appearance of metastases [11]. Furthermore, in [14] the authors describe a systematic approach to identifying efficient drug combinations in killing cancer cells depending on changes in the order and duration of drug exposure. They found that some drug combinations (EGFR inhibitor) can synergise the apoptotic response to DNA damaging chemotherapy for a subset of triple negative breast cancers if the drugs are given sequentially but not simultaneously, leading to an appropriate dynamics rewiring of oncogenic signalling networks.

Therefore, to widen the scope of potential applications in network medicine, we extend the previous works by investigating control sequences for investigating the sequences of perturbations as disease causes and for discovering therapeutic regimen as a long term perspective. A control sequence is composed of a list of control/topological network actions whose sequential application routes the dynamics of the network by steps to the expected fate. The control sequence analysis is still in its infancy. In [17], the authors study temporal reprogramming of Boolean networks based on Petri net analysis. Given a trajectory, they identify the appropriate states at which a control should be applied, and deduce the corresponding controls (perturbations) to reach an expected state. The algorithm explores the extended state graph encompassing perturbation representation, implying that the number of possible controls has to remain low to be tractable. An improved and generalised version of this algorithm is given in [18, 19]. In [22], we lay the foundations of a formal framework for describing control sequences, and we propose a first algorithm for inferring sequences satisfying several minimality properties. The present paper is a reworked and extended version of [22], including a streamlined formal framework, simplified proofs, a new inference algorithm, and a benchmark section showcasing the performance of our methods.

In this article, we define a computational method for inferring the minimal sequence of controls to reach some expected properties met at stable states under the synchronous dynamics.More specifically, we propose a theoretical framework describing a controlled dynamics enabling us to characterize a bound on the length of control sequences of minimal size.

The article is organised as follows. Section 2 recalls the principles of Boolean networks and introduces the extension to Boolean Control Networks by defining the main notions of controlled dynamics. In Section 3 we examine the properties of control sequences required to control network fate. In this section we also detail two approaches to inferring minimal control sequences under the synchronous dynamics.In Section 4 we compare the performance of the approaches.

2 Boolean Control Network

Boolean Control Networks (BCN) extend Boolean networks by adding Boolean controls which can alter the dynamics. In this section, we briefly recall the main definitions of Boolean networks (Section 2.1), and then we define the extension to BCN (Section 2.2). We more specifically focus on a particular class of control called the *freezing control* where a control input freezes a variable state to a specific value definitively. This category faithfully models the consequences of the perturbations of genetic and signalling networks blocking the gene expression in a particular regulation state, that are notably the consequences of mutations or drug effects.

Notations. We use the following notations. Let $E' \subseteq E$. We denote: $-E' = E \backslash E'$. Let f be a function with E as domain, $f_{\downarrow E'}$ defines the restriction/projection of the function to E' such that f is only defined for the elements of E'.

2.1 Boolean network

A Boolean network is a discrete dynamical system defined on Boolean variables X. A state s belonging to the set of states S_X is an interpretation assigning a Boolean value to the variables (*i.e.*, $s : X \to \mathbb{B}$). A Boolean network is defined by a collection of Boolean functions,

$$F = \{x_i = f_i(x_1, \ldots, x_n) \mid 1 \leq i \leq n\},$$

where each f_i is a propositional formula computing the state of x_i.

The *model of dynamics* describes the evolution of states for all variables by a labeled transition system $\langle \longrightarrow, M, S_X \rangle$, where the states are updated according to an updating policy $M \subseteq \mathbf{2}^X$, called the *mode*, which is a cover of X ($\bigcup_{m \in M} m = X$). Each transition relation ($\longrightarrow \subseteq S_X \times M \times S_X$) is labeled by the set of updated variables m:

$$s \xrightarrow{m} s' \stackrel{\text{def}}{=\!=} s' = (F_{\downarrow m}(s) \cup s_{\downarrow -m}). \tag{1}$$

The complement $-m$ is taken with respect to X. The global transition relation is defined as: $\longrightarrow = \bigcup_{m \in M} \xrightarrow{m}$. A path[3] $s \longrightarrow^* s'$ characterizes a trajectory from s to s'. In biological modeling two modes are preferentially used: the *synchronous*

[3] $\longrightarrow^*$ is the reflexive and transitive closure of the transition relation.

mode where all the variables are updated during a transition ($M = \{X\}$) or the *asynchronous* mode where one variable only is updated per transition ($M = \{\{x_i\}\}_{x_i \in X}$).

An *equilibrium* s is a particular state of the system which is infinitely often visited once this state is reached *i.e.*, $\forall s' \in S_X : s \longrightarrow^* s' \implies s' \longrightarrow^* s$. A *stable state* s is a particular equilibrium satisfying the stability condition: $\text{STBL}_F(s) \stackrel{\text{def}}{=\!=} F(s) = s$. The picture on the left of Figure 1 describes a Boolean dynamics under the synchronous mode.

2.2 Boolean control network

A BCN F_μ is a function generating a Boolean network from an interpretation $\mu \in S_U$ of control parameters $u_i \in U$, called a *control input*. It is defined as follows:

$$F_\mu = \{x_i = f_i(x_1, \ldots, x_n, u_1, \ldots, u_m\} \mid 1 \leq i \leq n\},$$

where the values u_j, $1 \leq j \leq m$, are given by μ. Each application of a control input F_μ leads to a Boolean network with a particular dynamics.

The freezing control assigns a definite value to each variable. The two possible freezing outcomes, 0 or 1, are supported by two parameters with two distinct regimes: either they freeze the variable or remain idle. By convention, inspired by the freezing temperature of water $0°C$, the freezing action is triggered when the control parameter is set to 0 whereas 1 stands for the idle situation. The implementation of the freezing control on a Boolean network augments the formulas of a Boolean network by adding the control parameter to obtain the expected control behavior. For a formula f_i, the addition of the control parameters $u_i^0 \in U^0$ and $u_i^1 \in U^1$ for respectively freezing the variable x_i to 0 or 1 leads to the following specification:

$$x_i = f_i(x_1, \ldots, x_n) \wedge u_i^0 \qquad \text{for freezing to 0,} \tag{2}$$

$$x_i = f_i(x_1, \ldots, x_n) \vee \neg u_i^1 \qquad \text{for freezing to 1.} \tag{3}$$

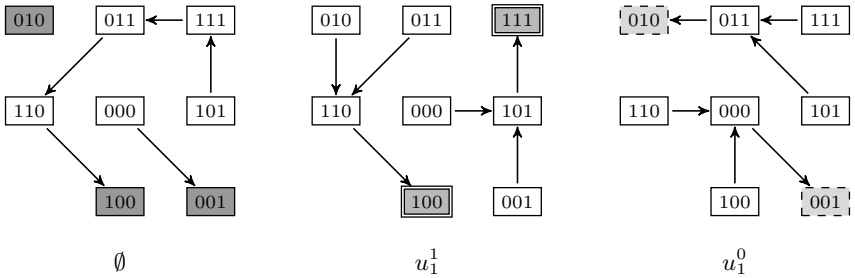
U^0 and U^1 control parameters can be combined to trigger the freeze to different values (*i.e.*, $x_i = f_i(x_1, \ldots, x_n) \wedge u_i^0 \vee \neg u_i^1$). In the sequel $U = U^0 \cup U^1$ will represent the set of indiscriminate freezing control parameters, and $u_i \in U$ a generic freezing control parameter (u_i^0 or u_i^1). The model can be extended to arc freezing [4, 5]. Figure 1 depicts the application of a control to variable x_1. Three different dynamics respectively corresponding to the absence of control, the freeze of variable x_1 to 1 ($u_1^1 = 0$) , and the freeze to 0 ($u_1^0 = 0$) are shown. The dynamics changes by the application of a control and leads to different equilibria.

The *active control* set of a control input, $\dot{\mu}$, represents the set collecting all the activated controls: $\dot{\mu} = \{u \mid \mu(u) = 0\}$. Notice, that μ and $\dot{\mu}$ are equivalent descriptions of the control since we can define one from the other one.

It is worth noticing that some variables are purposely uncontrolled to play the role of observers used for freely reporting the state evolution of a system. In

biology, biomarkers play the role of these observers. Therefore, the uncontrolled variables are important for assessing the fate of the dynamical system. The set of controlled variables is denoted C_X and the set of uncontrolled variables is $\bar{C}_X = X \setminus C_X$. Throughout the article, the profile of uncontrolled variables is denoted "$\bar{C}_X-$profile", and "C_X-profile" for controlled variables.

$$F = \begin{cases} x_1 = (x_1 \wedge \neg x_2) \vee (x_1 \wedge \neg x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3) \\ x_2 = (x_1 \wedge x_3) \vee (\neg x_1 \wedge x_2) \\ x_3 = (x_1 \wedge x_3) \vee (\neg x_1 \wedge \neg x_2) \end{cases}$$



Legend: The Boolean network F is completed by the formulas of the freezing controls to produce the equivalent BCN. From left to right the respective controls are: no freeze, x_1 is frozen to 1, x_1 is frozen to 0. The active control parameter are mentioned below each dynamics. The dynamics is synchronous and the self loops on states are not shown. The stable states of each dynamics are coloured in 3 shades of gray, and their contours are drawn in different styles. Each contour style is associated with a different control input.

Fig. 1. The synchronous dynamics of a Boolean control network.

2.3 Control sequence dynamics

The *controlled dynamics* extends the Boolean network dynamics by showing how the system evolves through a sequence of control inputs. A sequence of controls is formally defined by a function $\mu : \mathbb{N}^+ \to (U \to \mathbb{B})$ indexing control inputs, where $\mu_i, i \geq 1$, is the i-th control input in the sequence and $\mu_{[k]}$ stands for the sequence of size k starting with μ_1 and ending in μ_k: $\mu_{[k]} = (\mu_1, \ldots, \mu_k)$.

Controlled dynamics definition. Given a Boolean control network F_μ, the model of controlled dynamics is defined as a labeled transition system including the control inputs as labels $\langle S_X, S_U \times M, \longrightarrow \rangle$ such that a transition is defined by:

$$s \xrightarrow{\mu_i, m} s' \stackrel{\text{def}}{=\!=} s' = (F_{\mu_i})_{\downarrow m}(s) \cup s_{\downarrow -m}. \tag{4}$$

A control sequence $\mu_{[k]}$ leads therefore to the following trajectory (path):

$$s \xrightarrow{\mu_1, m_1} \ldots s_i \xrightarrow{\mu_i, m_i} s_{i+1} \ldots s_k \xrightarrow{\mu_k, m_k} s_{k+1}.$$

For the sake of clarity we omit the mode if it is not needed for the explanation[4]. For example, sequential application of the different controls described in Figure 1 leads to the following trajectory in the controlled dynamics by starting at state 000. The control inputs are represented by their active control sets, with elements indexed by the freeze value, and the stable states traversed by the trajectory are in bold face.

$$000 \xrightarrow{\emptyset} \mathbf{001} \xrightarrow{\{u_1^1\}} 101 \xrightarrow{\{u_1^1\}} \mathbf{111} \xrightarrow{\{u_1^0\}} 011 \xrightarrow{\{u_1^0\}} \mathbf{010}. \tag{5}$$

Although no paths connect 000 to 010 initially, the controlled trajectory enables the creation of a path between these two states by successive application of the controls u_1^1 and u_1^0.

State trace. The trace defines the sequence of visited states $(s_i)_{1 \leq i \leq k+1}$. For the example of Figure 1, the state trace of Trajectory (5) is:

$$(000, \mathbf{001}, 101, \mathbf{111}, 011, \mathbf{010})$$

Control Evolution based on Stable–State dynamics. The model of controlled dynamics is said to be *Control Evolution based on Stable–State dynamics* (ConEvs) if the modification of the control only occurs at a stable state only. The application of a control modifies the dynamics and releases the stability. Hence the ConEvs dynamics fulfills the following property:

$$\forall s_1 \xrightarrow{\mu_{[k]}} s_{k+1} : \mu_i \neq \mu_{i+1} \iff \mathrm{STBL}_{F_{\mu_i}}(s_{i+1}), \quad \text{given that } s_i \xrightarrow{\mu_i} s_{i+1} \xrightarrow{\mu_{i+1}} s_{i+2}, 1 \leq i < k. \tag{6}$$

In ConEvs dynamics, changing the control is the only way to evolve the dynamics since a stable state is reached with the current instance of the Boolean network resulting from the application of a control input to the BCN. The trajectory described in (5) is ConEvs.

Contracted control sequence. The *contracted control sequence* keeps only one instance of the control input for each sub-sequence having identical control inputs. For ConEvs dynamics, the contracted control sequence can be considered as the sequence making the dynamics evolve from stable states to stable states, and the initial control sequence can be easily retrieved by connecting the encountered stable states for each F_{μ_i} by a trajectory controlled by μ_i, and ending in the corresponding stable state. In the case of example (5), the contracted sequence represented by the active controls is $(\emptyset, \{u_1^1\}, \{u_1^0\})$.

Classes of sequences. Control sequences can be categorized into families based on the evolution of the control between steps:

[4] Formally, we consider the union relation: $\xrightarrow{\mu} = \bigcup_{m \in M} \xrightarrow{\mu,m}$.

– **Definition 1.** Total Control Sequence (TCS): *all the controls are triggered during the first phase for all controlled variables, and remain active all along the sequence; the values to which the variables are frozen may change.*
– **Definition 2.** Abiding Control Sequence (ACS): *once a control on a variable is triggered, the variable stays controlled but the value to which it is frozen may vary.*
– **Definition 3.** Open Control Sequence (OCS): *no constraints on control parameters are imposed; a control can be changed or released freely.*

The sequence described in (5) is an ACS sequence and, starting with the state 001, it is a TCS sequence since the only controlled variable x_1 is always controlled. The OCS family corresponds to the largest class of control sequences including ACS, which in its turn includes TCS. Therefore, the following inclusions between these families hold:

$$\text{TCS} \subset \text{ACS} \subset \text{OCS}.$$

The TCS class is mainly used for proofs, but has no particular biological application of its own. The ACS class models the consequences of a disease as mutations forbidding the relaxation of the control definitively, while enabling the possibility to change it or not according to the context of the biological process. The OCS class is the most general class that may represent the action of the drugs on molecular network potentially implying the modification and the relaxation of the actions.

In the rest of this article, we restrict ourselves to the synchronous dynamics of the Boolean networks. This choice is motivated by the fact that, under this update mode, the practical complexity of the reachability problem is reduced (even though reachability is PSPACE-complete for both update modes). Indeed, in the asynchronous update mode, one needs to tackle the fact that state transition becomes a relation inducing non-determinism, that should not be exhaustively explored to ensure the efficiency of an algorithm.

Proposition 1 states the observational equivalence between OCS and TCS classes under the synchronous update mode, namely, that any OCS control sequence can be simulated by a TCS sequence having the same state trace.

Proposition 1. *For any control sequence $\mu_{[k]}$ there exists a total control sequence of the same size $\nu_{[k]} \in$ TCS generating the same state trace under the synchronous mode.*

Proof. Take a control sequence $\mu_{[k]}$ and an initial state. For a transition $s_i \xrightarrow{\mu_i} s_{i+1}, 1 \leq i \leq k$, two cases may occur for the control parameters of the controlled variables, $x_j \in C_X$:

1. If one of the two control parameters u_j^0, u_j^1 is already activated then the configuration remains the same for ν.
2. If the control parameters are both idle ($u_j^0 = 1, u_j^1 = 1$) then we directly fix the expected final state value by setting the control appropriately, namely:
   $\nu_i(u_j^0) = 0, \nu_i(u_j^1) = 1$ if $s_{i+1}(x_j) = 0$ and,
   $\nu_i(u_j^0) = 1, \nu_i(u_j^1) = 0$ if $s_{i+1}(x_j) = 1$.

As the update is synchronous then all the value of the controlled variables, x_j, leads to the state $s_{i+1}(x_j)$ in a controlled way. For uncontrolled variables $x_j \in \bar{\mathcal{C}}_X$, we have: $(f_\nu)_j = (f_\mu)_j$ since no modifications occur, then the update is the same.

Since a transition only depends on the previous state that can be obtained by the application of a TCS control input ν_i, we can define a TCS control input for each step finally leading to a total controlled sequence $\nu_{[k]}$ simulating the trajectory controlled by $\mu_{[k]}$ from a s_1. □

3 Control sequence discovery

Finding a control sequence modifying the dynamics to evolve towards an expected state can be stated as a reachability problem:

Let $S_\alpha, S_\omega \subseteq S_X$ be two set of states, can we find a control sequence:

$$\mu_{[k]} = (\mu_1, \ldots, \mu_k) \text{ such that there exists a path } s_1 \xrightarrow{\mu_{[k]}} s_{k+1},$$

with: $s_1 \in S_\alpha$ and $s_{k+1} \in S_\omega$?

We refer to this problem as *"Controlled Fate in Sequence"* (CoFaSe) problem. Consider for example the Boolean network from Figure 1. The CoFaSe problem for this network with $S_\alpha = \{000\}$ and $S_\omega = \{010, 110\}$ as input parameters, means deciding the existence of a control sequence reaching any state in which $x_2 = 1$ and $x_3 = 0$ from the initial state 000. Figure 1 shows that this is not possible under the uncontrolled dynamics (leftmost graph), but applying the control $x_1 = 1$ to reach 111 (center graph) and then $x_1 = 0$ (rightmost graph) allows reaching 010. This sequence of controls therefore gives a solution to the above CoFaSe problem. Recall that x_1 is the only controllable variable for the network in Figure 1.

A sequence $\mu_{[k]}$ is said *minimal* for the CoFaSe problem with respect to F_u, S_α, and S_ω, if no control sequences $\nu_{[l]}$ satisfying the problem have a lower length, *i.e.*, $l \geq k$ is always true. A sequence $\mu_{[k]}$ is said *parsimonious* if the number of parallel activated controls is minimal to achieve the expected transition $s_i \xrightarrow{\mu_i} s_{i+1}$ for each control input $\mu_i, 1 \leq i \leq k$. Applied to ConEvs dynamics, the CoFaSe problem also imposes that that all states of S_α should be stable for uncontrolled F. Solving it implies that at least a state appearing in S_ω is stable for F_{μ_k}. The contracted control sequence $(\{u_1^1\}, \{u_1^0\})$ of the example (Figure 1) is minimal, parsimonious, and complies to the ConEvs condition for S_ω.

In biological modelling, the outcome of reprogramming can be formulated as a condition on the biomarkers checking whether the system has reached an expected signature. Therefore, by considering that the biomarkers are represented by the uncontrolled variables, S_ω will be defined by a predicate p formalizing the expected biological property as follows: $S_\omega = \{s \in S_X \mid p(s_{\downarrow \bar{C}_X})\}$. Notice that achieving a given state for *controlled* variables is trivial and consists in merely assigning their expected values by setting the appropriate control inputs. Therefore, the main problem lies on the way to indirectly influencing the state variation of the uncontrolled variables by freezing actions on controlled variables.

3.1 Complexity of CoFaSe

Finding a single parsimonious control is known to be NP-complete [4]. In this section we show that the inference of a control sequence satisfying CoFaSe is PSPACE-hard, which makes this problem even less tractable than finding single controls (assuming that PSPACE $\neq$ NP). Since the freezing to 0 and to 1 cannot be triggered simultaneously for a single variable, the cardinality of possible controlled transitions from a state is $3^{|X|} \cdot |M|$, meaning that finding the control sequence by exhaustively exploring these spaces is not practically tractable.

The problem of reachability in Boolean networks working in synchronous mode can actually be formalized as a CoFaSe problem. Indeed, reachability on a Boolean network is precisely the CoFaSe problem for a Boolean control network without controlled variables. Proof of Lemma 1 shows however that this reduction is not merely an artefact specific for such degenerate BCN. We can construct a network with a non-empty set of control variables and reduce the CoFaSe problem for this network to a reachability problem for a standard Boolean network.

Lemma 1. *Deciding whether a control sequence exists for the CoFaSe problem in the synchronous mode is at least as hard as reachability in (uncontrolled) Boolean networks in synchronous mode.*

Proof. Take an n-variable Boolean network F and construct a Boolean control network F' by adding to F the single control variable x_0 and defining the update functions f'_i of F' in terms of the update functions f_i of F in the following way:

$$\begin{aligned} f'_i &= f_i \wedge x_0, 1 \leq i \leq n, \\ f'_0 &= 0, \end{aligned}$$

where f'_0 is the update function for x_0.

Consider the controls $\mu_1 = d_0^1$ and $\mu_0 = d_0^0$ controlling x_0 to 1 and 0 respectively. The previous two properties ensure that the state graph of F'_{μ_1} is that of F, with $x_0 = 1$ added to each state, and that the state graph F'_{μ_0} only contains transitions to the state $\mathbf{0}$, which is the state in which all variables are 0.

Let X be the set of variables of F. The set of variables of F' is thus $X' = X \cup \{x_0\}$. Fix a set of starting states $S_\alpha \subseteq S_{X'}$ and a set of target states $S_\omega \subseteq S_{X'} \backslash (S_\alpha \cup \{\mathbf{0}\})$, such that the states in both sets satisfy $x_0 = 1$. The CoFaSe problem for the tuple (F', S_α, S_ω) has a solution if and only if the states in $S_{\omega\downarrow X}$ are reachable from $S_{\alpha\downarrow X}$ in F. Indeed, by construction of F' and since $\mathbf{0} \notin S_\omega$, the control sequence for this instance of CoFaSe may only be the singleton control sequence consisting of μ_1, and it must ensure the reachability of S_ω from S_α in F_{μ_1}, whose state graph is trivially isomorphic to that of F. Therefore, an oracle for CoFaSe would allow to solve reachability in Boolean networks working in the synchronous mode with at most polynomial overhead, which proves the statement of the lemma. □

The reachability hardness of CoFaSe is based on the result stating the PSPACE-completeness of reachability in 1-safe Petri nets [8], applied to Boolean networks

with the *asynchronous* mode. In the appendix of this paper, we provide an extension of this result to the synchronous mode. As far as we know, no characterization of complexity of reachability in Boolean networks working in the synchronous mode has been established in the literature before.

Lemma 2. *Given a Boolean network F with the variables X, a set of starting states $S_\alpha \subseteq S_X$, the set of target states $S_\omega \subseteq S_X \setminus S_\alpha$, it is PSPACE-hard to decide whether F can reach any of the states in S_ω from a state in S_α.*

Proof. The proof idea is to polynomial-time reduce the acceptance problem of a deterministic linear bounded automaton (a DLBA) to reachability for Boolean networks working in asynchronous mode. An LBA is a Turing machine which is only allowed to use at most $f(n)$ contiguous tape cells, where n is the size of the input and f is a linear function. Deciding whether a DLBA accepts a given input string is a PSPACE-complete problem (e.g., [12]).

Take a DLBA M and construct the Boolean network F simulating M in the following way. Define the Boolean variables $A_{i,j}$ and $Q_{i,k}$, where i indexes the tape cells of M, j indexes the symbols in the tape alphabet of M, and k indexes the states of M. The situation in which the i-th tape cell contains the j-th symbol is represented by setting $A_{i,j}$ to 1. The situation in which M is in the k-th state and the head is on the i-th tape cell is represented by setting $Q_{i,k}$ to 1. F operates by stepwise simulating the evolution of M: rewriting the j_1-th symbol to the j_2-th symbol in the i-th tape cell is done by setting A_{i,j_1} to 0 and A_{i,j_0} to 1, while moving the head from cell i_1 to i_2 and changing the state from k_1 to k_2 is simulated by setting Q_{i_1,k_1} to 0 and Q_{i_2,k_2} to 1. The synchronous dynamics of F can therefore faithfully simulate M, because M is deterministic.

For any input word w, the DLBA M reaches a configuration in the set of accepting configurations C_A if and only F can reach the encoding of one of the configurations C_A from the encoding of the initial configuration of M. The statement of the lemma follows from the facts that the procedure of constructing F from M is polynomial and that acceptance for DLBA is PSPACE-complete. □

Lemma 3. *Given a Boolean network F with the variables X, a set of starting states $S_\alpha \subseteq S_X$, the set of target states $S_\omega \subseteq S_X \setminus S_\alpha$, it is in PSPACE to decide whether F can reach any of the states in S_ω from one of the states in S_α.*

Proof. The proof idea is to construct a DLBA M which accepts the input if and only if the Boolean network F can reach a state in S_ω from a state in S_α. The initial configuration of M consists of the following three segments:

1. the list of binary vectors representing the states in S_α, each vector written in two copies;
2. the list of binary vectors representing the states in S_ω, each vector written in one copy;
3. an $|X|$-bit binary counter initialised to 0, where $|X|$ is the number of binary variables of F.

In the remainder of the proof we implicitly assume that the states of F are represented as Boolean vectors.

Consider a state $s \in S_\alpha$. The initial configuration of M contains a substring ss. M starts by simulating the transitions of F on one copy of s and replacing the other copy by the new state $s' = F(s)$, thereby yielding the new substring ss'. The subsequent operation of M is divided into macrosteps, during which it carries out the following actions :

1. calculate the new state for each pair of states in segment (1);
2. compare each new state with the states written in segment (2); if one of these comparisons is successful, M accepts, otherwise its continues to the following substep;
3. check if all the bits of the binary counter in segment (3) are 1; if yes, reject, otherwise, commence the next macrostep.

Intuitively, M simulates the deterministic synchronous dynamics of F on every state in segment (1), accepts if it sees a target state from S_ω, or rejects after $2^{|X|}$ steps. Counting to $2^{|X|} = |S_X|$ ensures that the entire state graph of F reachable from S_α is visited. Therefore, M accepts if and only if F can reach at least one state in S_ω from at least one state in S_α. Constructing M from the triple (F, S_α, S_ω) is a polynomial-time procedure, meaning that an oracle for DLBA acceptance would allow deciding reachability for Boolean networks working in the synchronous mode with polynomial overhead. This proves the statement of the theorem. □

Theorem 1. *Given a Boolean network F with the variables X, a set of starting states $S_\alpha \subseteq S_X$, and the set of target states $S_\omega \subseteq S_X \setminus S_\alpha$, it is PSPACE-complete to decide whether F can reach any of the states in S_ω from one of the states in S_α.*

This theorem, combined with Lemma 1, gives the following lower bound on the complexity of CoFaSe.

Theorem 2. *Deciding the existence of a control sequence for the CoFaSe problem in the synchronous mode is PSPACE-hard.*

Whether CoFaSe is in PSPACE remains an open question. Two levels of complexity can be considered in CoFaSe: reachability and control discovery, indicating that the upper bound on the complexity of this problem may be high.

3.2 Partitioning heuristic

As inference of a control sequence satisfying CoFaSe is PSPACE-hard, we should rely on approximation heuristics to find some solutions. Therefore we are looking for factors that would significantly reduce the search space in practice. Proposition 1 offers insights into the critical role of the uncontrolled variables in the resolution of the CoFaSe problem. Indeed, as the variables in C_X are fully controlled with the TCS sequence, no evolution of the dynamics may arise from

these variables. Thus, visiting an already encountered $\bar{C}_X$-profile with a different C_X-profile is not relevant for solving CoFaSe problem. In fact, the dynamics of controllable variables can be ignored as it can be reproduced by a sequential control.

We propose a heuristic method focused on the partitioning of the states of the Boolean networks with respect to their $\bar{C}_X$-profiles. Starting from a model of dynamics, we define the quotient graph called *partitioned dynamics* representing the transitions over the partition. The following equivalence relation is used for its definition:

$$\forall s, s' \in S_X : s \sim s' \Leftrightarrow s_{\downarrow \bar{C}_X} = s'_{\downarrow \bar{C}_X}. \tag{7}$$

The partitioned dynamics of the Boolean network F_U is thus a labelled transition system $\langle \mapsto, U, S_{\bar{C}_X} \rangle$ with the set of transitions $\{[s] \mapsto [s'] \mid \exists \mu : s \xrightarrow{\mu} s'\}$, where $[s]$ denotes the equivalence class of the state s according to the equivalence relation (7). The resulting partitioned dynamics models the intrinsic dynamical interactions between signature variations of the biomarkers and provides us an overview of possible sequence-controlled evolution of the network.

Take s a state, if $\exists \mu \in S_U, \exists s, s' \in S_X : s \xrightarrow{\mu} s'$ then $[s']$ is defined to be the *target equivalence class* of s. Propositions 2 and 3 give observational properties of target equivalence classes and their states in synchronous evolution.

Proposition 2. *In the synchronous mode, regardless of the control, each transition from a state leads to states belonging to a unique target equivalence class.*

Proof. Assume an initial state s with s_1 and s_2 two states derived from the one step synchronous evolution of respectively F_{μ_1} and F_{μ_2}. Suppose that the following is true:

$$\exists \mu_1, \mu_2 \in S_U : s \xrightarrow{\mu_1} s_1 \wedge s \xrightarrow{\mu_2} s_2 \wedge s_{1 \downarrow \bar{C}_X} \neq s_{2 \downarrow \bar{C}_X}$$

By definition $\bar{C}_X$-variables are not controllable. In synchronous mode, the dynamics is determined by anterior states. Thus, the states s_1 and s_2 cannot have different equivalence classes by means of controls. This contradicts the above equation and substantiates the statement of the proposition. □

Proposition 3. *In the synchronous mode, a state can reach any state of its target equivalence class in a single step under an appropriate control.*

Proof. In synchronous mode all C_X-profiles are reachable by a total control from any state in a single dynamics step. Thus, the equivalence relation (7) and Proposition 2 corroborate the statement of the proposition. □

For a better understanding of the evolution of states with respect to equivalence classes, we distinguish *impermanent states* which remain transient whatever the applied control, from *enduring states* that can be stabilized by an appropriate control. Formally:

$$\forall \mu \in S_U : \neg STBL_{F_\mu}(s) \qquad \text{impermanent state,} \tag{8}$$

$$\exists \mu \in S_U : STBL_{F_\mu}(s) \qquad \text{enduring state.} \tag{9}$$

Any state s must either be impermanent or enduring: the definitions in equations (8) and (9) are logical opposites of each other. We can further derive $\bar{C}_X$-related properties from these definitions. First, if s is impermanent, $F_\mu(s) \neq s$ even for a total control μ which freezes all the controlled variables to some values. This necessarily implies that the uncontrolled variables evolve, for any μ: $F_\mu(s)_{\downarrow \bar{C}_X} \neq s_{\downarrow \bar{C}_X}$. On the other hand, if s is enduring, then $F_\mu(s) = s$ and therefore $F_\mu(s)_{\downarrow \bar{C}_X} = s_{\downarrow \bar{C}_X}$, for some μ. But, since the control effectively sets the values of the controlled variables for the *next* state, this last identity actually holds for *any* μ. Indeed, whatever the choice of μ, the update functions of $\bar{C}_X$-variables will read the values of the variables from s, meaning that the $\bar{C}_X$-profile will stay the same in the next step. Formally:

$$\begin{aligned} &s \text{ is impermanent} && \text{if} : \forall \mu \in S_U : \; s \xrightarrow{\mu} s' \implies s_{\downarrow \bar{C}_X} \neq s'_{\downarrow \bar{C}_X}, \\ &s \text{ is enduring} && \text{if} : \forall \mu \in S_U : \; s \xrightarrow{\mu} s' \implies s_{\downarrow \bar{C}_X} = s'_{\downarrow \bar{C}_X}. \end{aligned}$$

Therefore a self loop in the partitioned dynamics with an identical state of uncontrolled variables reveals the existence of at least an enduring state in the equivalence class. Similarly, an edge between two equivalence classes $[s]$ and $[s']$ in the partitioned dynamics implies the existence of an impermanent state in $[s]$ whose target equivalence class is $[s']$.

3.3 Bounds on sequence size

The properties related to equivalence classes enable us to define an upper bound on the length of minimal control sequences.

Proposition 3 and the definition of an impermanent state give us insights into the resolution of the CoFaSe problem: a minimal sequence solving the problem jumps from an impermanent state to another until it reaches the target equivalence class, which represents the property expected on $\bar{C}_X$-profiles. Based on this observation, Theorem 3 defines an upper bound on the size of minimal sequences that only depends on the number of equivalence classes and thus of uncontrolled variables.

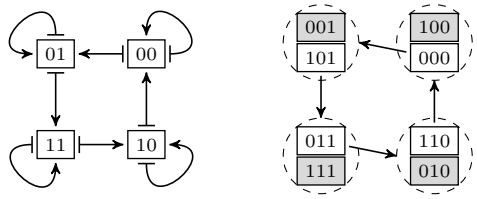
Theorem 3. *The size of the minimal control sequence $\mu_{[k]}$ solving a given CoFaSe problem is bounded by $2^{|\bar{C}_X|}$ for the synchronous mode: $\left|\mu_{[k]}\right| \leq 2^{|\bar{C}_X|}$.*

Proof. Assume that $\mu_{[k]}$, with $k > 2^{|\bar{C}_X|}$, is a minimal sequence solving the CoFaSe problem $\langle F_\mu, S_\alpha, S_\omega \rangle$. The control sequence $\mu_{[k]}$ yields the following sequence of states: $s_1 \xrightarrow{\mu_1} s_2 \dots s_k \xrightarrow{\mu_k} s_{k+1}$, with $s_1 \in S_\alpha$ and $s_{k+1} \in S_\omega$.

From Proposition (3) we know that all states of a target equivalence class are reachable in one step under the appropriate control. Since only the update of impermanent states evolves the uncontrolled variables, it is possible to reach from an impermanent state the next desired impermanent state or property. Then, the main steps for solving CoFaSe consist in finding the minimal path, from a state of S_α to a state of S_ω, by traversing solely impermanent through impermanent states of different equivalence classes. Therefore a minimal control

sequence does not pass through two states of a same equivalence class. Since $\mu_{[k]}$ is minimal, the states from s_2 to s_k are impermanent states and the equivalence classes from $[s_2]$ to $[s_k]$ are different target equivalence classes of their former state. The sequence then spans across $k-1 \geq 2^{|\bar{C}_X|}$ different equivalence classes. Since only $2^{|\bar{C}_X|}$ $\bar{C}_X$-profiles exist, the sequence must have a state $s_i, 2 \leq i \leq k$, with the same $\bar{C}_X$-profile as $s_k + 1$. Note that $2 \leq i$ because the first state s_1 may not be an impermanent state. Thus, from the state s_{i-1} having for target equivalence class $[s_i] = [s_{k+1}]$, it is possible to reach $s_k + 1$ and to yield the following sequence $s_1 \xrightarrow{\mu_1} \cdots s_{i-1} \xrightarrow{\mu'_{i-1}} s_{k+1}$. Hence the sequence $\mu_{[k]}$ is not minimal, thus contradicting the original assumption and proving the statement of the theorem. □

Theorem 3 shows the critical role of uncontrolled variables in determining the control sequence. In practice, the number of uncontrolled variables standing for biomarkers is very low compared to the controlled variables that represent the other molecules of a network [4, 6, 7]. Moreover by the definition of the CoFaSe problem, the evolution of the uncontrolled variables guides the discovery of the control since the objective is to reach an expected final state characterized by a property defined on the uncontrolled variables. In Section 3.5, we show an algorithm based on the exploration of the minimal paths in the partitioned dynamics linking the target equivalence classes of the initial states in S_α to the equivalence classes of those in S_ω.



Legend: On the left is the synchronous partitioned dynamics of the Boolean network of the Figure 1 for $\bar{C}_X = \{x_2, x_3\}$. On the right, the enduring states of each equivalence class of the partitioned dynamics are colored in gray and impermanent states, in white, are connected to their target equivalence class.

Fig. 2. A synchronous partitioned dynamics with impermanent and enduring states.

Figure 2 shows the partitioned synchronous dynamics of the example Boolean network from Figure 1 and the corresponding graph of equivalence classes. By following the minimal path between the equivalence classes [000] and [010] in the equivalence graph, we obtain the following sequence:

$$000 \xrightarrow{\{d_1^1\}} 101 \xrightarrow{\{d_0^1\}} 011 \xrightarrow{\{d_0^1\}} 010.$$

This sequence is a minimal sequence solving the CoFaSe problem for this network with $S_\omega = \{110, 010\}$ and $S_\alpha = \{000\}$.

3.4 Bounds on sequence size for ConEvs dynamics

Finding a minimal parsimonious contracted OCS solving the CoFaSe problem under the ConEvs dynamics appears relevant for biological applications. Indeed, this framework models either the different mutational steps where a mutation rewires the network reaching another phenotype the molecular signature of which is stable, as in the case for the Vogelstein sequence [11], or a therapeutic regimen where the drug administering depends on the therapeutic evaluation modelled by a stable state assessment [14].

Determining low upper bounds on the size of control sequences would indicate that algorithms based on direct exploration of the sequence space can be an efficient solution for sequence inference. Theorem 3 establishes such an upper bound, but it is proved for the particular context under which the control can be changed at any state. This bound is not therefore directly applicable to ConEvs, where control changes are only allowed at stable states. The following theorem establishes an upper bound for this semantics.

Theorem 4. *The size of the minimal contracted control sequence $\mu_{[k]} \in OCS$ solving the CoFaSe problem (F, S_α, S_ω) for the ConEvs model of dynamics under the synchronous mode is at most $2^{|\bar{C}_X|+1}$:*

$$\left|\mu_{[k]}\right| \leq 2^{|\bar{C}_X|+1}.$$

Proof. Consider the CoFaSe problem $(F_u, S_\alpha, S_\omega)$ and assume that $\mu_{[k]} \in \text{OCS}$ is a minimal contracted control sequence solving it for the ConEvs model of dynamics. This control sequence gives rise to the trajectory $T = s_1 \xrightarrow{\mu_1}{}^{*} s_2 \ \ldots \ s_k \xrightarrow{\mu_k}{}^{*} s_{k+1}$, with $s_1 \in S_\alpha$, $s_{k+1} \in S_\omega$, and the states s_i, $1 < i < k+1$, being the stable states at which the control is changed. We will use the symbol τ to refer to the sequence of stable states, plus the initial and the final states: $\tau = (s_i)_{1 \leq i \leq k+1}$.

Now assume that $k > 2^{|\bar{C}_X|+1}$. Since k is greater than the double of the number of all states over $\bar{C}_X$, it must be that τ contains three states having the same target equivalence class. Suppose that these three states are at positions $1 \leq h < i < j \leq k+1$, *i.e.*, for $s_h \xrightarrow{\mu_h} s'_h$, $s_i \xrightarrow{\mu_i} s'_i$ and $s_j \xrightarrow{\mu_j} s'_j$ $[s'_h] = [s'_i] = [s'_j]$. Note that, at least two different controls must appear between the states s_h and s_j because there is at least the stable state s_i in between the two.

By construction, as $1 > i > j$, the stable states s_i, s_j are enduring states and thus have for target equivalence class their one equivalence class. According to Proposition 3, it is possible from s_h to find a control μ'_h reaching s_j in one step. This gives rise to the following trajectory $T' = s_1 \xrightarrow{\mu_1}{}^{*} s_2 \ \ldots \ s_h \xrightarrow{\mu'_h} s_j \ \ldots \ s_k \xrightarrow{\mu_k}{}^{*} s_{k+1}$. This contradicts our initial assumption that $\mu_{[k]}$ is minimal.

Therefore any control sequence solving the CoFaSe problem for the ConEvs model of dynamics and having more than $2^{|\bar{C}_X|+1}$ elements is not minimal, which proves the statement of the theorem. □

Theorem 4 implies the possibility of the occurrence of states with the same equivalence class in the contracted trace, called *duplicates*. Since in minimal contracted control sequences all intermediary states are enduring states, the proof of the theorem also entails that any equivalence class may appear at most twice, and necessarily in consecutive states of the trace, excluding the first state[5]. Intuitively, if duplicate equivalence class appear in non-successive steps i and j, the whole evolution between i and j can be skipped by applying an appropriate control input.

Corollary 1. *Consider a minimal contracted control sequence $\mu_{[k]} \in OCS$ solving the CoFaSe problem (F, S_α, S_ω) for the ConEvs model of dynamics under the synchronous mode. Take the sequence $\tau = (s_i)_{1 \leq i \leq k+1}$ of states induced by $\mu_{[k]}$, with $s_1 \in S_\alpha$, $s_{k+1} \in S_\omega$, $\mu_{k+1} = \mu_k$, and the states s_i, $1 \leq i < k+1$, being the stable states at which the control is changed. If there exist two indices $1 \leq i < j \leq k+1$ such that $s_i \xrightarrow{\mu_i} s'_i, s_j \xrightarrow{\mu_j} s'_j$ and $[s'_i] = [s'_j]$, then $j = i+1$.*

Proof. As the two target equivalence classes $[s'_i]$ and $[s'_j]$ are equal, and as s_j is an enduring state by hypothesis, it follows from Proposition 3 and Theorem 4 that it is possible to find a control μ'_i reaching s_j from s_i in one step, whenever $j > i+1$. Thus, in a minimal control sequence, the controls appearing between indices $i+1$ and j can be skipped: $j = i+1$. □

Set $S_\alpha = \{010\}$, $\bar{C}_X = \{x_1\}$, and $S_w = \{1\star\star\}$[6] as CoFaSe parameters. The contracted trace of the following trajectory for the example in Figure 1, controlled by the minimal parsimonious contracted control sequence $(\{u_2^0\}, \{u_2^1\})$, contains the duplicate $x_1 = 0$ occurring in the initial state 010 and the stable state 001.

$$010 \xrightarrow{\{u_2^0\}} 000 \xrightarrow{\{u_2^0\}} \mathbf{001} \xrightarrow{\{u_2^1\}} 011 \xrightarrow{\{u_2^1\}} \mathbf{110}.$$

3.5 Inference of minimal parsimonious contracted control sequences

The upper-bound of the sequence depends on $\bar{C}_X$-profiles only. Therefore the uncontrollable variables are central for the inference of a control sequence motivation a $\bar{C}_X$ state partitioning approach for the sequence inference. Accordingly, we define an algorithm based on the partitioning with regards to uncontrolled variable states that infers all minimal parsimonious control sequences solving the CoFaSe problem for the ConEvs model of dynamics. The algorithm will find a sequence of controls evolving from stable states to stable states. By convention motivated by the biological application, we assume the property on the expected final states to only concern the uncontrolled variables.

As the number of uncontrolled variables is in practice markedly lower than the number of controlled variables (*e.g.*, [4, 6, 7]), the exhaustive exploration of

[5] The equivalence class of the fist state may appear at most three times if it is an impermanent state *i.e.*, its target equivalence class and its own equivalence class are distinct.

[6] $1\star\star$ corresponds to the set of states $\{100, 101, 110, 111\}$.

all possible profiles for these variables constitutes an efficient approach for control sequence computation. Furthermore, our algorithm is optimized to avoid redundant operations. Informally the algorithm builds a tree describing the possible paths reaching a state of S_ω from the initial set of states S_α. Since the algorithm solves the CoFaSe problem for the ConEvs model of dynamics, the set of states S_α should be composed of stable states under the original dynamics of the studied network. The shortest paths/trajectories are found in the tree, from which the minimal parsimonious control sequences are directly derived by keeping their control inputs.

Phases of Algorithm 1. Algorithm 1 comprises two major phases. The first phase (steps 1 and 2) corresponds to the search for a control allowing to directly attain a state of S_ω. The second phase (steps 3 and 4) corresponds to searching a trajectory visiting an intermediary state with an unexplored equivalence class. At each step a parsimonious control input is inferred with the method presented in [4, 5]. The evolution of the main steps is detailed in Figure 3.

Data structures. The algorithm relies on the following data structures:

1. *the list* Δ of equivalence classes according to the equivalence relation 7;
2. *the exploration tree* G with nodes labelled by sets of stable states and edges labelled by controls;
3. *the sets* Γ_l, Γ_{l+1}, *and* Γ_{l+2} of unexplored nodes of the tree for the current level of depth l, and the next two levels, respectively.

At the beginning, Δ is initialized to $[S_X] \setminus ([S_\alpha \cup S_\omega])$, the depth is $l = 1$, Γ_1 contains the root node $\{S_\alpha\}$ of the exploration tree, and all the other data structures are empty. Remark that we extend the notation of equivalence class to sets of states: $[A] = \{[a] \mid a \in A\}$.

Duplicate states. A specific treatment is applied to take into account the case where a trajectory passes through duplicates. As all states in the explored sequence are stable and thus enduring states (*i.e.*, their target equivalence class is equal to their equivalence class), a pair of duplicate states can be described as two states belonging to the same equivalence class. By considering that duplicates can only occur in 2 successive states of any given minimal contracted sequence (Corollary 1), we propose two strategies:

1. Let γ be a set of initial states and γ' the set of stable states of F_μ reachable from γ. Infer the set of parsimonious control inputs μ validating the following equation:

$$\exists s \in \gamma,\ \exists s' \in S_X : s \xrightarrow{\mu}{}^{*} s' \wedge \mathrm{STBL}_{F_\mu}(s') \wedge [s'] \subseteq [\gamma] \wedge s' \notin \gamma. \tag{10}$$

   For each such control μ, infer a parsimonious control input μ' such that the set of stable states $S_{\mu'}$ of $F_{\mu'}$ reachable from γ' contains some elements satisfying the property p.

Algorithm 1 Inference of minimal parsimonious contracted control sequences

1. *Direct reachability of S_ω:* For all $\gamma \in \Gamma_l$, infer the control μ taking the BCN from γ to a state with an equivalence class included in $[S_\omega]$. If such a μ exists, add the arc labelled by μ to G and go to step 6.
2. *Reachability of S_ω via a duplicate:* For all $\gamma \in \Gamma_l$, infer a pair of controls (μ, μ') such that μ takes the BCN to a state with an equivalence class included in $[\gamma]$, and μ' takes the BCN from there to some of the target equivalence classes. If such a pair of controls exists, add two chained arcs labelled by μ and μ' to G and go to step 6.
3. *Direct reachability of Δ:* For every $\gamma \in \Gamma_l$, infer a set of controls $\mathcal{U}$ taking the BCN from γ to a state with an equivalence class included in Δ. If $\mathcal{U}$ is non-empty, add the arcs labelled by the controls from $\mathcal{U}$ to G, delete the equivalence class reached in Δ , and store the sets of stable states that allow reaching in Γ_{l+1}.
4. *Reachability of Δ via a duplicate:* For every $\gamma \in \Gamma_l$, infer a set of pairs of controls $\mathcal{D} = \{(\mu, \mu') \mid \mu, \mu' \in S_U\}$ such that μ takes the BCN to a state with an equivalence class included in $[\gamma]$, and μ' takes the BCN from there to some of the equivalence classes in Δ which we could not be directly reached at the previous step. If $\mathcal{D}$ is not empty, add chained arcs labelled by the pairs of controls from $\mathcal{D}$ to G, delete the equivalence class reached in Δ and store the sets of stable states they allow to reach in Γ_{l+2}.
5. *Continue if states left:* If one of Γ_l, Γ_{l+1}, or Γ_{l+2} is non-empty, go to step 1 with $l = l + 1$.
6. *Produce the result:* Find the sequence of controls by backtracking G from a leaf found in steps 1 or 2 to the root S_α. If no such leaf was found, return $\emptyset$.

2. Let γ be a set of initial states. Infer a pair of parsimonious control inputs μ and μ' validating the following equation:

$$\begin{aligned} \exists s \in \gamma,\ \exists s', s'' \in S_X : s \xrightarrow{\mu}{}^{*}\ s' \wedge \mathrm{STBL}_{F_\mu}(s') \wedge [s'] \subseteq [\gamma] \\ \wedge\ s' \xrightarrow{\mu'}{}^{*}\ s'' \wedge \mathrm{STBL}_{F_{\mu'}}(s'') \wedge p(s''). \end{aligned} \tag{11}$$

The first strategy may in some cases not finds the intermediary state needed to reach the property p. For example, let us consider the following Boolean network:

$$F = \begin{cases} x_1 = (x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3 \wedge x_4) \\ x_2 = \neg x_4 \wedge x_2 \\ x_3 = \neg x_4 \wedge x_3 \\ x_4 = x_4 \end{cases}$$

with $\bar{C}_X = \{x_1\}, C_X = \{x_2, x_3, x_4\}, S_\alpha = \{0000\}$, and $S_\omega = 1\star\star\star$ as CoFaSe parameters. The minimal parsimonious control sequence of size 2 reaching a state in S_ω from S_α is $\mu_{[2]} = \{\{u_2^1, u_3^1\}, \{u_4^1\}\}$ inducing the following trajectory:

$$0000 \xrightarrow{\{u_2^1, u_3^1\}} \mathbf{0110} \xrightarrow{\{u_4^1\}} 0111 \xrightarrow{\{u_4^1\}} \mathbf{1001}.$$

Strategy 1 will not find this sequence. In fact, for $s = 0000$ equation (10) only yields the parsimonious controls $\{\{u_2^1\}, \{u_3^1\}\}$ giving the states $\{0100\}$ and $\{0010\}$, from which no state satisfying the target property $x_1 = 1$ can be reached.

Conversely, the second strategy finds all duplicates. In fact, equation (11) completely formalizes the necessity of having a duplicate to validate the target property. However, the second strategy is more costly computationally speaking than the first. In the inference of the pair of controls μ and μ', the dynamics of both F_μ and $F_{\mu'}$ need to be evaluated at the same time, which effectively doubles the number of variables for which an assignment must be found.

Correctness. Algorithm 1 closely follows the proofs of Theorem 4 and of Corollary 1, which guarantees the correctness and the minimal parsimony of the result. In other words, the sequences found by Algorithm 1 solve the CoFaSe problem for the ConEvs model of dynamics under the synchronous mode, and they are minimal and parsimonious.

Theorem 5. *Algorithm 1 returns all minimal parsimonious control sequences $\mu_{[k]}$ solving the CoFaSe problem for the ConEvs model of dynamics under the synchronous mode.*

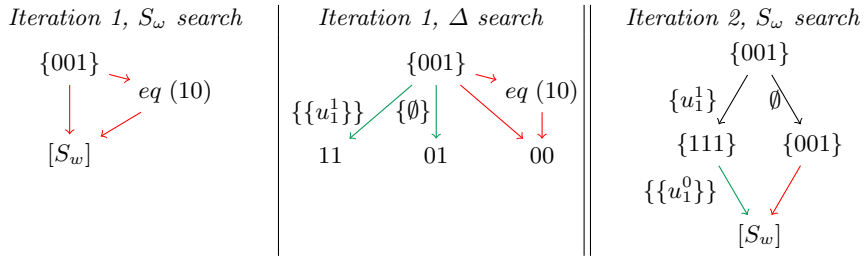
Proof. The compliance of $\mu_{[k]}$ with the ConEvs model of dynamics and its minimal parsimony are guaranteed by the use of the parsimonious one-shot inference algorithm from [4, 5]. This algorithm yields parsimonious controls allowing to reach stable states satisfying certain properties. The fact that $\mu_{[k]}$ solves the given CoFaSe problem follows from the end condition in step 6. □

Example. Figure 3 illustrates the computation of a control sequence. The data structures are initialized as follows: $l = 1$, $\Gamma_1 = \{S_\alpha\}$, $\Delta = \{[11], [01], [00]\}$, and all the other data structures are empty. On the left, we seek to reach a state in S_ω. As no state with an equivalence class included in $[S_\omega]$ is reachable from S_α, we search for a potential trajectory via a duplicate. Since no states validate (10), and thus no other enduring states exist in $[S_\omega]$, we know that such a trajectory does not exist.

As no sequences were found, the second phase starts. Here, all equivalence classes in Δ will be used as new targets. In the middle section of Figure 3, we see that $11 \in \Delta$ is reachable from s_α with the control $\{u_1^1\}$, and $01 \in \Delta$ is reachable from s_α with the control $\emptyset$, while $00 \in \Delta$ is not reachable. We add to Γ_2 the stable states γ' of $F_{\{u_1^1\}}$ belonging to the searched equivalence class $[11] \in \Delta$. We also add the stable states γ'' of $F_\emptyset$ belonging to the searched equivalence class $[01] \in \Delta$. Accordingly, we update Δ by deleting [11] and [01].

After creating the new edges of the tree $s_\alpha \xrightarrow{\{u_1^1\}} \gamma'$ and $s_\alpha \xrightarrow{\emptyset} \gamma''$, we test for new trajectories which would attain an equivalence class in Δ via a duplicate. Since no states validate (10), we know that such a trajectory does not exist.

The right section of Figure 3 shows that the second iteration of the algorithm is started after updating all the sets of initial variables with new values: $l = 2$,



Legend: Steps of the construction of the tree built by Algorithm 1. For the Boolean control network F_U of Figure 1 and the set of initial states $S_\alpha = \{001\}$, Algorithm 1 infers the control sequence allowing to reach $S_\omega = \star 10$. The only controlled variable is $C_X = \{x_1\}$ and the uncontrolled ones are $\bar{C}_X = \{x_2, x_3\}$.
A black edge is a branch of the tree G connecting the previous initial states to the new set of stable states of F_U, under the control given in the annotation. A green edge represents the possibility to reach a state validating the target property by a set of parsimonious controls. A red edge corresponds to a failure to find a control input leading to the targeted property.
The left and the middle sections depict the first iteration of the algorithm. For $\Gamma_1 = \{S_\alpha\}$, it respectively tries to reach S_ω, and as this is not possible, tries to reach states from an equivalence class included in Δ. The rightmost section depicts the first phase of the second iteration where the algorithm tries to reach S_ω for the new set of unexplored nodes $\Gamma_2 = \{\{111\}, \{001\}\}$ found in the middle section. The rightmost section is also the final step of this execution of the algorithm as S_ω is reachable from $\{111\}$. The returned sequential control is $\{\{u_1^1\}, \{u_1^0\}\}$.

Fig. 3. Iterations of Algorithm 1 on the Boolean network of Figure 1.

$\Gamma_2 = \{\{111\}, \{001\}\}$, and $\Delta = \{00, 01\}$. Observe now that a state with an equivalence class included in $[S_\omega]$ is reachable from the initial state 111 under the control $\{u_1^0\}$. The main loop of the algorithm stops here. After adding the last edge to G, the sequence $\{\{u_1^1\}, \{u_1^0\}\}$ is constructed by backtracking the resulting tree.

Parsimony & optimality. Minimality is proved by remarking that the parsimony condition is satisfied for the control inputs at each step of the inference. However, it is worth noticing that it may be possible to find shorter control sequences by relaxing the parsimony constraint. Consider the following Boolean network:

$$F = \begin{cases} x_1 = x_1 \vee (x_2 \wedge x_3 \wedge \neg x_4) \\ x_2 = (\neg x_1 \wedge x_2) \vee (\neg x_1 \wedge x_4) \vee (x_2 \wedge \neg x_3) \vee (x_2 \wedge x_4) \\ x_3 = (x_1 \wedge x_3) \vee (\neg x_2 \wedge x_3) \vee (x_3 \wedge x_4) \\ x_4 = x_4 \end{cases}$$

with $\bar{C}_X = \{x_1, x_2\}$, $C_X = \{x_3, x_4\}$, $S_\alpha = \{0000\}$, and $S_\omega = 11\star\star$ as CoFaSe parameters. The minimal parsimonious control sequence of size 3 inferred by Algorithm 1 is $\mu_{[3]} = \{\{u_4^1\}, \{u_3^1\}, \{u_4^0\}\}$. However, the following sequence of

size 2: $\nu_{[2]} = \{\{u_3^1, u_4^1\}, \{u_4^0\}\}$ also converges to the same final state 1100. Their trajectories are respectively:

$$T_\mu = 0000 \xrightarrow{\{u_4^1\}} 0001 \xrightarrow{\{u_4^1\}} \mathbf{0101} \xrightarrow{\{u_3^1\}} \mathbf{0111} \xrightarrow{\{u_4^0\}} 0110 \xrightarrow{\{u_4^0\}} \mathbf{1100},$$
$$T_\nu = 0000 \xrightarrow{\{u_3^1, u_4^1\}} 0011 \xrightarrow{\{u_3^1, u_4^1\}} \mathbf{0111} \xrightarrow{\{u_4^0\}} 0110 \xrightarrow{\{u_4^0\}} \mathbf{1100}.$$

Since the target equivalence class $11\star\star$ cannot be directly attained from the starting state 0000, Algorithm 1 infers a control sequence driving the BCN through some intermediary states. In the example above, the equivalence class $01\star\star$ is picked first, and the state 0101 is then reached by the single parsimonious freezing of x_4 to 1. The next stable state 0111, in the same equivalence class, is reached by parsimonious freezing x_3 to 1. The stable state 1100 in the target equivalence class is then reached by parsimoniously controlling x_4 to 0.

In contrast, the application of the non-parsimonious control $\{u_3^1, u_4^1\}$ at the starting state 0000 reaches the stable state 0111 directly, thereby reducing the length of the control sequence ν to 2. The first control in ν can be seen as a union of the first two controls appearing in μ, but such compression does not work in general.

3.6 TCS-based inference

Since the dynamics of any controlled variable can be adjusted arbitrarily by the control input, inferring a control sequence can be seen as a problem of driving the subnetwork formed by the uncontrolled variables to the target configuration. In this section, we show a two-step approach to inferring a control sequence: first, infer a total control sequence (TCS) driving the $\bar{C}_X$ subnetwork to satisfying the target property, then, secondly, derive a control sequence with smaller-size controls, *i.e.*, controls acting on as few control variables as possible. By first inferring a TCS control sequence, we significantly reduce the complexity of the problem for networks with a small number of uncontrolled variables. We will later show that this approach yields solutions of roughly the same quality as those given by Algorithm 1, but considerably more efficient (Figure 5).

We start by giving an upper bound on the size of minimal *total* control sequences solving a given CoFaSe problems for the ConEvs model of dynamics. This theorem is a refinement of Theorem 3 for TCS.

Theorem 6. *The size of the minimal contracted total control sequence $\mu_{[k]} \in TCS$ solving the CoFaSe problem (F, S_α, S_ω) for the ConEvs model of dynamics under the synchronous mode is at most $2^{|\bar{C}_X|}$:*

$$\left|\mu_{[k]}\right| \leq 2^{|\bar{C}_X|}.$$

Proof. Consider the CoFaSe problem $(F_u, S_\alpha, S_\omega)$ and assume that $\mu_{[k]} \in$ TCS is a minimal contracted total control sequence solving it for the ConEvs model of dynamics. This control sequence gives rise to the trajectory $T = s_1 \xrightarrow{\mu_1}{}^* s_2 \ \ldots \ s_k \xrightarrow{\mu_k}{}^* s_{k+1}$, with $s_1 \in S_\alpha$, $s_{k+1} \in S_\omega$, and the states s_i, $1 < i < k+1$,

being the stable states at which the control is changed. We will use the symbol τ to refer to the sequence of stable states, plus the initial and the final states: $\tau = (s_i)_{1 \leq i \leq k+1}$.

Now assume that $k > 2^{|\bar{C}_X|}$. Since k is greater than the number of all states over $\bar{C}_X$, it must be that τ contains two states having the same target equivalence class. Suppose that these two states are at positions $1 \leq i < j < k+1$, *i.e.*, for $s_i \xrightarrow{\mu_i} s'_i$, $s_j \xrightarrow{\mu_j} s'_j$ and $[s'_i] = [s'_j]$.

According to Proposition 2 and since no dynamics evolution can result from the update of C_X-variables (the control is total), controlling s_i with μ_j reaches the state s'_j in one step. This gives rise to the trajectory $T' = s_1 \xrightarrow{\mu_1}{}^* s_2 \ \ldots \ s_i \xrightarrow{\mu_j}{}^* s_{j+1} \ \ldots \ s_k \xrightarrow{\mu_k}{}^* s_{k+1}$, contradicting our initial assumption that $\mu_{[k]}$ is minimal.

Therefore any control sequence solving the CoFaSe problem for the ConEvs model of dynamics and having more than $2^{|\bar{C}_X|}$ elements is not minimal, which proves the statement of the theorem. □

We now introduce Algorithm 2 which exhaustively explores all possible $\bar{C}_X$-profiles by building a tree describing the possible paths from the initial states to a target state. As in Algorithm 1, the control sequences are directly derived from this tree by collecting the controls annotating its edges. In contrast to Algorithm 1, TCS inference does not need to explore duplicate states, because Theorem 6 shows that no duplicate states can occur in minimal TCS sequences. Thus, no solutions are found by Algorithm 2 when the property can only be reached by control sequences having a duplicate occurrence.

Phases of Algorithm 2. Algorithm 2 comprises two major phases. The first phase (step 1) corresponds to the search for a total control allowing to directly attain a state of S_ω. The second phase (step 2) corresponds to, for a trajectory, visiting intermediary equivalence states. The total controls are inferred with a SAT solver.

Data structures. The algorithm relies on the following data structures:

1. *the list* Δ of equivalence classes according to the equivalence relation 7;
2. *the exploration tree* G with nodes labelled by equivalence classes and edges labelled by total controls;
3. *the sets* Γ_l, Γ_{l+1} of unexplored nodes of the tree for the current level of depth l, and the next level.

At the beginning, Δ is initialized to $[S_X]/([S_\alpha \cup S_\omega])$, the depth is 1, Γ_1 contains the root equivalence classes $\{[s] \mid \forall s \in S_\alpha\}$ of the exploration tree, and all the other data structures are empty.

Correctness. Algorithm 2 closely follows the proof of Theorem 6, which guarantees the correctness and the minimality of the result. More concretely, the

Algorithm 2 Inference of minimal contracted total control sequences

1. *Direct reachability of S_ω:* For all $\gamma \in \Gamma_l$, infer the total control μ^t taking the BCN from the equivalence class γ to some of the target equivalence classes in $[S_\omega]$. If such a μ^t exists, add the arc labelled by μ^t to G and go to step 4 .
2. *Direct reachability of Δ:* For every $\gamma \in \Gamma_l$, infer a set of total controls $\mathcal{U}^t$ taking the BCN from the equivalence class γ to some of equivalence classes in Δ. If $\mathcal{U}^t$ is non-empty, add the arcs labelled by the controls from $\mathcal{U}^t$ to G, delete from Δ and store in Γ_{l+1} the set of reached equivalence classes.
3. *Continue if states left:* If Γ_{l+1} is non-empty, go to step 1 with $l = l + 1$.
4. *Produce the result :* Find the trajectory of the sequence of total controls by backtracking G from a leaf found in step 1 to a root equivalence class in Γ_0. If no such leaf was found, return $\emptyset$.

sequences found by Algorithm 2 solve the CoFaSe problem for the totally controlled ConEvs model of dynamics under the synchronous mode. These sequences are minimal.

Total controls are impractical for biological applications, because they act on all controlled variables all the time, even when it is not strictly necessary. To tackle this problem, we introduce Algorithm 3, which reduces a sequence of total controls by testing, for each total control, whether one of its subsets allows reaching a stable state validating the desired property.

Phases of Algorithm 3. Algorithm 3 comprises only one iterated phase: for every total control μ_i^t appearing in a contracted control trajectory $(s_i^t \xrightarrow{\mu_i^t} s_{i+1}^t)_{1 \leq i \leq n}$ induced by a TCS sequence $[\mu_n^t]$, it considers all subsets of μ_i^t, and picks the smallest ones which allow reaching the same equivalence class $[s_{i+1}^t]$. As all previous algorithms, Algorithm 3 focuses mainly on the ConEvs model of dynamics.

Data structures. Algorithm 3 relies on the following data structures:

1. *the input contracted total control trajectory* $(s_i^t \xrightarrow{\mu_i^t} s_{i+1}^t)_{1 \leq i \leq n}$;
2. *the resulting reduced control sequence* $\mu_{[n]}$;
3. *the sets* S_l, S_{l+1} of unexplored initial states of the reduced sequence for the current level of depth l, and the next level;
4. *the set* S_μ of stable states the BCN can reach under a given control μ.

At the beginning, $(s_i^t \xrightarrow{\mu_i^t} s_{i+1}^t)_{1 \leq i \leq n}$ is initialized with a contracted total control trajectory (e.g., one found by Algorithm 2), the depth l is 1, S_1 is equal to S_α, and all the other data structures are empty.

Parsimony & optimality. Algorithm 3 always picks the smallest control subsets. Therefore, if every control in a given TCS sequence is a superset of a corresponding control in a control sequence composed of minimal control, Algorithm 3 will

Algorithm 3 Iterative reduction of total control sequences

1. *Reduce the current total control:* For every control subset μ of the total control μ_l^t, construct the set of stable states S_μ the BCN reaches from S_l under μ. Pick the smallest control μ for which $[s_{l+1}^t] \in [S_\mu]$. Append μ to $\mu_{[n]}$ and set $S_{l+1} = \{s \mid \forall\, s \in S_\mu : [s] = [s_{l+1}^t]\}$.
2. *Continue if the sequence is not finished:* If $l \leq n$, go to step 1 with $l = l + 1$. Otherwise return the reduced sequence $\mu_{[n]}$.

always find it. However, if the input control sequence is *not* a superset of such control sequence, the result of Algorithm 3 will be a sequence of parsimonious controls, but will not be composed of only minimal controls.

3.7 Summary of the algorithms

For the reference, we give here a short summary of the approaches to inference and of the algorithms we presented in this section.

The first approach (also introduced in [22]), consists in an exhaustive exploration of all possible equivalence classes of $\bar{C}_X$-profiles and controlled trajectories between them. It is implemented in Algorithm 1, which organises the explored equivalence classes in a tree, stores the found controls on its edges, and then derives a minimal parsimonious control sequence from a path in the tree connecting the root—the starting equivalence classes in $[S_\alpha]$—to the first leaf corresponding to one of the target equivalence classes in $[S_\omega]$.

The second approach is also based on exhaustive exploration of equivalence classes and control trajectories, but this time only *total* controls are considered. Algorithm 2 works rather similarly to Algorithm 1: it constructs the tree of explored equivalence classes, whose edges are annotated by the found total controls. Since freezing every single C_X-variable is not always necessary, the controls in the output of Algorithm 2 are further individually reduced (Algorithm 3).

4 Benchmarks

To assess the efficiency of the proposed approaches for inference of control sequences, we benchmark them on a set of randomly generated Boolean networks. We compare their computational time, memory used and the quality of their results. In the below subsections, we refer to the first approach as *OCS-based inference* and the second approach as *TCS-based inference*.

4.1 Experimental protocol

The experiments consist in comparing the effectiveness and the performance of *one-step* control inference by the two proposed approaches on random Boolean networks. Indeed, inferring a single control is the core part of both methods and

also the principal difference between them. Evaluating the performance of one-step inference is therefore a good indicator for comparing their computational costs. Furthermore, considering one-step inference only eliminates the potential bias in the observed resource consumption which may be introduced by the presence of sequences of different lengths in the sample networks.

As regulatory networks appear to be scale-free [1], we generate random Boolean networks from random scale-free interaction graphs obtained with the *Barabási-Albert model* [2]. We generate networks of 10 to 35 variables with an increment of 5. For each of the sizes, we test on 100 different random Boolean networks the inference of a one-step control, reaching a set of target states S_ω from a set of initial states S_α. The initial set of states S_α originates from the set of stable states of the random Boolean network. The initial set of target states S_ω originates from the enduring states. Around 30% of the network variables are randomly picked to be the uncontrollable variables in $\bar{C}_X$. Table 4.1 gives the exact values of different parameters, as well as the technical specifications of the machine used to run the benchmarks.

Characteristics of the computer used to perform the experiments	Model	Macbook Pro				
	CPU	*Intel Core i7* of *2.8GHz*				
	RAM	*16Gb* of *DDR3*				
Implementation language	Wolfram Mathematica					
Parameters of the generation of random Boolean networks	*Barabási-Albert model* distribution parameter					2.1
	probability to have a self-loop					0.1
	probability of having positive regulation and not a negative regulation					0.6
Generated Boolean networks size	10	15	20	25	30	35
$\bar{C}_X$-profile size	3	4	6	8	9	10

Table 1. Parameters of the benchmarks.

The Boolean formula for reachability is obtained by 20-iterated symbolic composition of the random Boolean network formula—preliminary experimentation revealed that 20 compositions were enough to reach stable states. The inference of one-step controls is time-constrained to *1 hour* of computation time. Due to excessive computational time of OCS-based inference, the method is only applied to networks of size 10 and 15 (see Figure 4). The results will be defined in relation to the median since outliers present in the dataset risk biasing the average.

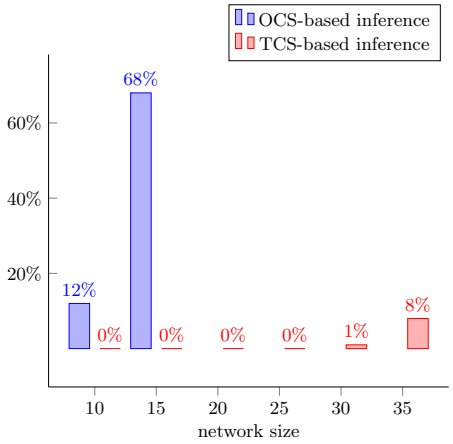


Fig. 4. Percentage of the OCS-based and TCS-based one-step inference runs, aborted after an hour of computation.

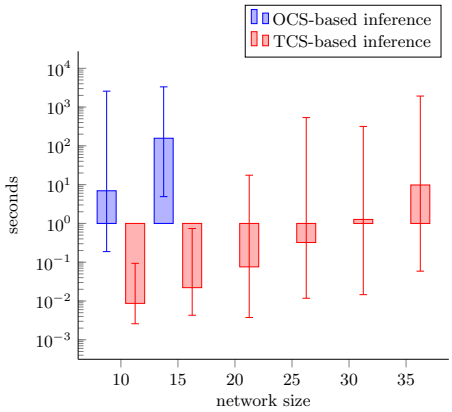


Fig. 5. Median of the computation time of the OCS-based and TCS-based one-step inference, excluding aborted runs.

4.2 Results

Figure 4 describes the percentage of timed out computations. Figures 5, 6, and 8 are realized without taking into account the aborted inferences. Figure 5 describes the median of the total computational time of the two approaches, including the minimum and the maximum. Figure 6 describes the median, with the minimum and the maximum, of the total memory used in bytes of the two approaches. Figure 7 describes the median, with the minimum and the maximum, of the total memory used in bytes by the timed-out inferences. Figure 8 describes the median, with the minimum and maximum, of the number of variables controlled by the inferred one-step controls. Figure 9 describes the percentage of times the TCS-based inference finds a parsimonious but not optimal control *i.e.*, the inferred control is bigger than the OCS-based inferred control. We recall that as OCS-based inference is optimal by definition, a control inferred by the TCS-based inference cannot be smaller than the corresponding OCS-based inferred control.

4.3 Discussion of the benchmarks

The benchmark clearly shows a performance gap between the computation time of the two approaches (Figure 5). The disparity becomes more obvious when taking into account that for networks of sizes 10 and 15, respectively 10% and 70% (Figure 4) of computations are aborted after 1 hour and thus are omitted in Figure 5.

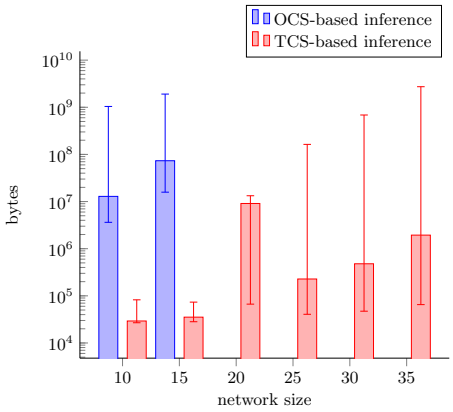


Fig. 6. Median of memory usage by OCS-based and TCS-based one-step inference, excluding aborted computations.

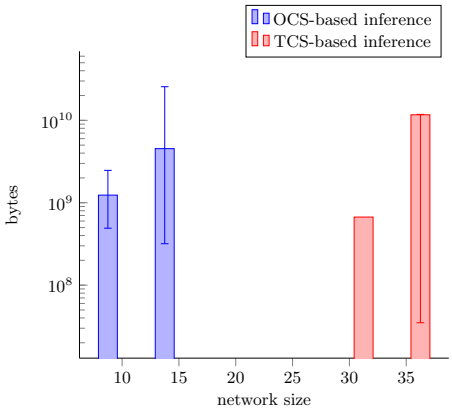


Fig. 7. Median of memory usage by OCS-based and TCS-based inferences in aborted computations.

Figure 4 shows that, on networks of size 35, the TCS-based inference results in a similar number of aborted computations as the OCS-based inference for networks of size 10. Furthermore, Figure 5 shows that the times of the TCS-based inference on networks of size 35 are similar to the times of the OCS-based inference on networks of size 10. This correspondence is due to the fact that the TCS-based inference discards the dynamics of the controlled variables by freezing all of them, thereby considerably reducing the complexity of reachability for a given control. Since we pick around a third of variables to be uncontrollable, the TCS-based inference only considers the dynamics of about 10 variables for networks of size 35.

In Figure 6, we observe a jump in median memory usage for networks of size 20. As minimum and maximum values seem to be within normal range, this variation is probably due to the algorithm generating the random Boolean networks and therefore not significant for the algorithm.

In Figure 6 and Figure 7, for networks of size 10 and 35, we observe the same pattern as in Figures 4 and 5 while noticing that the median memory used by the OCS-based inference is bigger than it counterpart. This can be attributed to the fact that Boolean formulas for reachability of the TCS-based inference may generally be less complex for similar numbers of variables, because the sets $\bar{C}_X$ are randomly selected and thus do not follow the power law distribution of scale-free networks. In Figure 8, we can see that the number of controlled variables remains small, even when the size of the network increases. From this, we can conclude that the reduction carried out by the Algorithm 3 generally results in rather small controls.

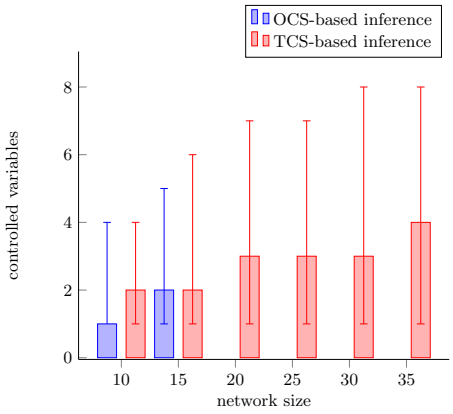


Fig. 8. Median of the sizes of controls found by the OCS-based and TCS-based one-step inference, excluding aborted computations.

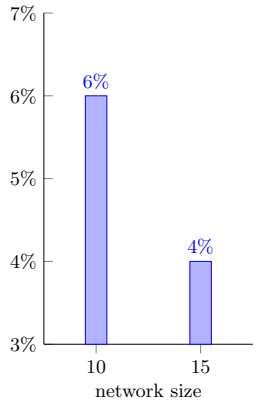


Fig. 9. Percentage of times OCS-based inference yields a smaller control than TCS-based inference.

Since the size of the Boolean formula for reachability increases as network size increases, the time needed for the inference of a one-step control by the solver also increases. By only taking into account the dynamics of $\bar{C}_X$-profiles, the TCS-based inference postpones the computational explosion , in contrast to the OCS-based inference. As we can see in all figures, this entails a better performance in terms of time and memory used by the second approach compared to the first one. In Figure 9 we can observe that the number of times where the TCS-based algorithm gives a parsimonious but not optimal solution is low.

In these benchmarks, we compared the effectiveness and the performance of one-step control inference of the OCS-based approach and the TCS-based approach on random Boolean networks. From these experiments, we observe that TCS-based inference in comparison to the OCS-based inference is less costly in computation time and memory usage. By definition the OCS-based inference always returns the minimal control. The experimentation revealed that TCS-based inference returns minimal controls most of the time. From these observations, we can conclude that the TCS-based approach infers solutions of good quality and is a rather viable alternative to the more computationally costly OCS-based approach.

5 Conclusion

In this article, we study the sequential control applied to Boolean networks. We propose a theoretical framework aiming at discovering minimal control sequences evolving from stable states to stable states (ConEvs model of dynamics), for

parsimonious control actions at each step. Inference of control sequences can be used in precision medicine for the causal analysis of such complex diseases as cancer [11], stressing the evolution of the tumorigenesis from benign to malignant tumoral states. The control sequence inference can also be used to determine efficacious order of drug administering for chemotherapy [14].

Our analysis emphasizes nontrivial complex features of control sequences, such as the occurrence of duplicates where only the controlled variables evolve without changing the states of uncontrolled variables. Such occurrences are interpreted as the need to evolve to different stable states with the same profiles for uncontrolled variables, but which could not be reached previously.

We proposed two approaches for inferring control sequences, both based on exhaustive exploration of the possible intermediate profiles of uncontrolled variables. The first approach is implemented in Algorithm 1 and applies the single control inference algorithm from [5] at most $2^{|\bar{C}_X|}$ times. This bound makes it tractable, because the uncontrolled variables typically constitute a rather small set. The second approach is implemented as a chaining of Algorithms 2 and 3, and focuses on inferring a *total* control sequence first, and then reducing the sizes of individual controls. This implies that the second approach does not take into account the occurrence of duplicates. Our benchmark shows that inferring total controls is much cheaper because the dynamics of uncontrolled variables is totally neglected, while the quality of the solutions stays pretty close to the quality of the sequences inferred by Algorithm 1.

The perspective of our work is twofold: applying the method to biological cases for investigating complex treatment schemes, more specifically for cancer, and designing an algorithm guaranteeing the minimality in a broader context, notably by relaxing the parsimony requirement and by considering other modes, such as the asynchronous one. For the asynchronous mode, one needs to tackle the fact that the transition becomes a relation inducing non-determinism that should not be exhaustively explored for ensuring the efficiency of the algorithm.

References

1. Réka Albert. Scale-free networks in cell biology. *Journal of Cell Science*, 118(21):4947–4957, 2005.
2. Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.
3. Albert-László Barabási, Natali Gulbahce, and Joseph Loscalzo. Network medicine: a network-based approach to human disease. *Nature reviews. Genetics*, 12:56–68, 2011.
4. Célia Biane and Franck Delaplace. Abduction based drug target discovery using Boolean control network. In *Proceedings Computational Methods in Systems Biology - 15th International Conference, CMSB 2017, Darmstadt, Germany, September 27-29*, pages 57–73, 2017.
5. Célia Biane and Franck Delaplace. Causal reasoning on Boolean control networks based on abduction: theory and application to cancer drug discovery. *IEEE/ACM transactions on computational biology and bioinformatics*, 2018.

6. Laura N. Burga, Hai Hu, Ashish Juvekar, Nadine M. Tung, Susan L. Troyan, Erin W. Hofstatter, and Gerburg M. Wulf. Loss of BRCA1 leads to an increase in epidermal growth factor receptor expression in mammary epithelial cells, and epidermal growth factor receptor inhibition prevents estrogen receptor-negative cancers in BRCA1-mutant mice. *Breast Cancer Research*, 13(2):R30, 2011.
7. Cindy H. Chau, Olivier Rixe, Howard McLeod, and William D. Figg. Validation of analytic methods for biomarkers used in drug development. *Clinical Cancer Research*, 14(19):5967–5976, 2008.
8. Allan Cheng, Javier Esparza, and Jens Palsberg. Complexity results for 1-safe nets. *Theoretical Computer Science*, 147(1):117 – 136, 1995.
9. Pau Creixell, Erwin M. Schoof, Craig D. Simpson, James Longden, Chad J. Miller, Hua Jane Lou, Lara Perryman, Thomas R. Cox, Nevena Zivanovic, Antonio Palmeri, Agata Wesolowska-Andersen, Manuela Helmer-Citterich, Jesper Ferkinghoff-Borg, Hiroaki Itamochi, Bernd Bodenmiller, Janine T. Erler, Benjamin E. Turk, and Rune Linding. Kinome-wide decoding of network-attacking mutations rewiring cancer signaling. *Cell*, 163(1):202–217, 2015.
10. Peter Csermely, Tamàs Korcsmàros, Huba J. M. Kiss, Gàbor London, and Ruth Nussinov. Structure and dynamics of molecular networks: A novel paradigm of drug discovery: A comprehensive review. *Pharmacology and Therapeutics*, 138(3):333–408, 2013.
11. Eric R. Fearon and Bert Vogelstein. A genetic model for colorectal tumorigenesis. *Cell*, 61(5):759–767, 1990.
12. Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.
13. Junil Kim, Sang-Min Park, and Kwang-Hyun Cho. Discovery of a kernel for controlling biomolecular regulatory networks. *Scientific reports*, 3:2223, 2013.
14. Michael Lee, Albert S. Ye, Alexandra K. Gardino, Anne Heijink, Peter Sorger, Gavin Macbeath, and Michael Yaffe. Sequential application of anti-cancer drugs enhances cell death by re-wiring apoptotic signaling networks. *Cell*, 149:780–794, 05 2012.
15. Pey-Chang Kent Lin and Sunil P Khatri. Application of Max-SAT-based ATPG to optimal cancer therapy design. *BMC Genomics*, 13(Suppl 6):S5, 2012.
16. Jianquan Lu, Jie Zhong, Daniel WC Ho, Yang Tang, and Jinde Cao. On controllability of delayed boolean control networks. *SIAM Journal on Control and Optimization*, 54(2):475–494, 2016.
17. Hugues Mandon, Stefan Haar, and Loïc Paulevé. Temporal reprogramming of Boolean networks. In Jérôme Feret and Heinz Koeppl, editors, *Computational Methods in Systems Biology - 15th International Conference, CMSB 2017, Darmstadt, Germany, September 27-29, 2017, Proceedings*, volume 10545 of *Lecture Notes in Computer Science*, pages 179–195. Springer, 2017.
18. Hugues Mandon, Cui Su, Stefan Haar, Jun Pang, and Loïc Paulevé. Sequential reprogramming of Boolean networks made practical. In *International Conference on Computational Methods in Systems Biology*, pages 3–19. Springer, 2019.
19. Hugues Mandon, Cui Su, Jun Pang, Soumya Paul, Stefan Haar, and Loïc Paulevé. Algorithms for the Sequential Reprogramming of Boolean Networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16, Issue 5:1610–1619, 2019.
20. Mohammad Moradi, Sama Goliaei, and Mohammad-Hadi Foroughmand-Araabi. A boolean network control algorithm guided by forward dynamic programming. *PloS one*, 14(5):e0215449, 2019.

21. David Murrugarra, Alan Veliz-Cuba, Boris Aguilar, and Reinhard Laubenbacher. Identification of control targets in Boolean molecular network models via computational algebra. *BMC Systems Biology*, 10(1):94, 2016.
22. Jérémie Pardo, Sergiu Ivanov, and Franck Delaplace. Sequential reprogramming of biological network fate. In *International Conference on Computational Methods in Systems Biology*, pages 20–41. Springer, 2019.
23. Soumya Paul, Cui Su, Jun Pang, and Andrzej Mizera. A decomposition-based approach towards the control of boolean networks. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 11–20, 2018.
24. Nidhi Sahni, Song Yi, Quan Zhong, Noor Jailkhani, Benoit Charloteaux, Michael E. Cusick, and Marc Vidal. Edgotype: a fundamental link between genotype and phenotype. *Current opinion in genetics & development*, 23(6):649–657, 2013.
25. Kyle Strimbu and Jorge a Tavel. What are Biomarkers? *Current Opinion in HIV and AIDS*, 5(6):463–466, 2011.
26. Cui Su, Soumya Paul, and Jun Pang. Controlling large boolean networks with temporary and permanent perturbations. In *International Symposium on Formal Methods*, pages 707–724. Springer, 2019.
27. Gretchen Vogel. Reprogramming cells. *Science*, 322(5909):1766–1767, 2008.
28. Jorge G.T. Zañudo and Réka Albert. Cell fate reprogramming by control of intracellular network dynamics. *PLoS Computational Biology*, 11(4):e1004193, 2015.
29. Quan Zhong, Nicolas Simonis, Qian-Ru Li, Benoit Charloteaux, Fabien Heuze, Niels Klitgord, Stanley Tam, Haiyuan Yu, Kavitha Venkatesan, Danny Mou, Venus Swearingen, Muhammed a Yildirim, Han Yan, Amélie Dricot, David Szeto, Chenwei Lin, Tong Hao, Changyu Fan, Stuart Milstein, Denis Dupuy, Robert Brasseur, David E Hill, Michael E. Cusick, and Marc Vidal. Edgetic perturbation models of human inherited disorders. *Molecular Systems Biology*, 5(321):321, 2009.