Output feedback stabilization of Boolean control networks ⋆

Nicoletta Bof, Ettore Fornasini, Maria Elena Valcher [a]

[a] *Dipartimento di Ingegneria dell'Informazione, Università di Padova, Italy.*

Abstract

In the paper output feedback control of Boolean control networks (BCNs) is investigated. First, necessary and sufficient conditions for the existence of a time-invariant output feedback (TIOF) law, stabilizing the BCN to some equilibrium point, are given, and constructive algorithms to test the existence of such a feedback law are proposed. Two sufficient conditions for the existence of a stabilizing time-varying output feedback (TVOF) are then given. Finally, an example concerning the *lac* Operon in the bacterium *Escherichia Coli* is presented, to illustrate the effectiveness of the proposed techniques.

Key words: Boolean control networks, output feedback, system biology.

1 Introduction

Research in Boolean networks (BNs) and Boolean control networks (BCNs) has a long tradition. The renewed interest witnessed in recent times, however, must be mainly credited to two reasons: on the one hand, BNs and BCNs have proved to be effective modeling tools for a number of rapidly evolving research topics, like genetic regulation networks and consensus problems Lou and Hong (2010); Shmulevich *et al.* (2002). On the other hand, the algebraic framework developed by D. Cheng and co-authors Cheng (2009); Cheng and Qi (2010); Cheng *et al.* (2011a) has allowed to cast both BNs and BCNs into the framework of linear state-space models (operating on canonical vectors), thus benefitting of a number of powerful algebraic tools, in addition to traditional graph-based techniques. By resorting to this approach, many properties of BCNs such as the stability of an equilibrium point or a limit cycle, controllability, observability, and reconstructibility have been thoroughly investigated, and important control problems, such as state feedback stabilization, state-observer design, finite and infinite horizon optimal control have been solved (see, e.g. Cheng and Liu (2009); Cheng *et al.* (2011b); Fornasini and Valcher (2013a,b); Li *et al.* (2013)). The purpose of this contribution is to investigate output feedback control and to single out structural conditions that guarantee the existence of a stabilizing output feedback control law for a BCN. Static output feedback involves only memoryless operations on measurable quantities, and hence, when applicable, it provides an extremely simple and reliable tool. In particular, when modeling a biological regulation network, introducing a feedback loop from some specific (output) variables can be viewed either as an attempt to achieve a more complete representation of the network dynamics, since feedback loops are intrinsic of the network functioning, or as an external action aimed at modifying the network behavior. The stabilizing output feedback problem has been addressed in Li and Wang (2013), where an algebraic characterization of a logical matrix that describes a stabilizing time invariant output feedback (TIOF) control law has been provided (Theorem 1). This condition is not computationally meaningful, as it does not provide a practical way to test whether a stabilizing output feedback law exists or not. On the other hand, the necessary and sufficient condition for output feedback stabilization provided in Theorem 2 of Li and Wang (2013) is actually only sufficient. The reason for this is that the class of state feedback matrices K the Authors consider in order to obtain output feedback matrices K_y, by making use of the equation $K = K_y H$, are those obtained through the algorithm given in Li *et al.* (2013), an algorithm that provides only those state feedback matrices K that guarantee that each state reaches the equilibrium state, $\mathbf{x}_e$, along the shortest possible path. As we shall see in Example 4, in some cases stabilization to a given state $\mathbf{x}_e$ is achievable through state feedback, but not through output feedback. In other cases, a stabilizing output feedback does exist, but the associated state feedback law does not implement a shortest path strategy (see Example 1), that is always achievable when a state feedback is carefully designed. So stabilizing TIOF, when available, often achieves the stabilization

⋆ This paper was not presented at any IFAC meeting. Corresponding author: Maria Elena Valcher, Tel: +39 049 8277795; Email: `meme@dei.unipd.it`

to the equilibrium point at the price of a slower dynamics. In other words, there is no way of interpreting the associated control law as a shortest path state feedback law Fornasini and Valcher (2013b); Li *et al.* (2013).
The paper is organized as follows. In section II, we investigate under what conditions there exists a TIOF law, stabilizing a BCN to some equilibrium state $\mathbf{x}_e$. In section III, we provide sufficient conditions for the existence of stabilizing time-varying output feedback (TVOF) laws. Finally, section IV outlines the main features of the *lac* Operon model, and discusses the stabilization to a fixed point when TIOF or TVOF laws are implemented. A preliminary version of part of the results of sections II and III appeared in Fornasini and Valcher (2014).
Notation. $\mathbb{Z}_+$ denotes the set of nonnegative integers. Given $k, n \in \mathbb{Z}_+$, with $k \le n$, the symbol $[k, n]$ denotes the integer set $\{k, k+1, \dots, n\}$. We consider Boolean vectors and matrices, taking values in $\mathcal{B} := \{0,1\}$, with the usual operations (sum $\vee$, product $\wedge$ and negation $\neg$). δ_k^i denotes the ith canonical vector of size k, $\mathcal{L}_k$ the set of all k-dimensional canonical vectors, and $\mathcal{L}_{k\times n} \subset \mathcal{B}^{k\times n}$ the set of all $k \times n$ matrices whose columns are canonical vectors. Any matrix $L \in \mathcal{L}_{k\times n}$ can be represented as a row vector whose entries are canonical vectors, i.e. $L = [\delta_k^{i_1}\ \delta_k^{i_2}\ \dots\ \delta_k^{i_n}]$, for some indices $i_1, i_2, \dots, i_n \in [1, k]$. The k-dimensional vector with all entries equal to 1 is denoted by $\mathbf{1}_k$. The (ℓ, j)th entry of a matrix M is denoted by $[M]_{\ell j}$, while the ℓth entry of a vector $\mathbf{v}$ is $[\mathbf{v}]_\ell$.
Given a matrix $L \in \mathcal{B}^{k\times k}$ (in particular, $L \in \mathcal{L}_{k\times k}$), we associate with it a *digraph* $\mathcal{D}(L)$, with vertices $1, \dots, k$. There is an arc (j, ℓ) from j to ℓ if and only if $[L]_{\ell j} = 1$. A sequence $j_1 \to j_2 \to \dots \to j_r \to j_{r+1}$ in $\mathcal{D}(L)$ is a *path* of length r from j_1 to j_{r+1} provided that $(j_1, j_2), \dots, (j_r, j_{r+1})$ are arcs of $\mathcal{D}(L)$. A closed path is called a *cycle*. In particular, a cycle γ with no repeated vertices is called *elementary,* and its length $|\gamma|$ coincides with the number of (distinct) vertices appearing in it.
There is a bijective correspondence between Boolean variables $X \in \mathcal{B}$ and vectors $\mathbf{x} \in \mathcal{L}_2$, defined by

$$\mathbf{x} = \begin{bmatrix} X \\ \neg X \end{bmatrix}.$$

$\mathbf{x}$ is called *the vector form* of the Boolean variable X.
The *(left) semi-tensor product* $\ltimes$ between matrices $L_1 \in \mathbb{R}^{r_1\times c_1}$ and $L_2 \in \mathbb{R}^{r_2\times c_2}$ (in particular, $L_1 \in \mathcal{L}_{r_1\times c_1}$ and $L_2 \in \mathcal{L}_{r_2\times c_2}$) is defined as

$$L_1 \ltimes L_2 := (L_1 \otimes I_{T/c_1})(L_2 \otimes I_{T/r_2}), \quad T := \text{l.c.m.}\{c_1, r_2\},$$

where l.c.m. denotes the least common multiple. The semi-tensor product represents an extension of the standard matrix product, by this meaning that if $c_1 = r_2$, then $L_1 \ltimes L_2 = L_1 L_2$. If $\mathbf{x}_1 \in \mathcal{L}_{r_1}$ and $\mathbf{x}_2 \in \mathcal{L}_{r_2}$, then $\mathbf{x}_1 \ltimes \mathbf{x}_2 \in \mathcal{L}_{r_1 r_2}$. For the properties of the semi-tensor product we refer to Cheng *et al.* (2011a). By resorting to the semi-tensor product, we can extend the previous correspondence to a bijective correspondence between $\mathcal{B}^n$ and $\mathcal{L}_{2^n}$. Given $X = [X_1\ X_2\ \dots\ X_n]^\top \in \mathcal{B}^n$, just set

$$\mathbf{x} := \begin{bmatrix} X_1 \\ \neg X_1 \end{bmatrix} \ltimes \begin{bmatrix} X_2 \\ \neg X_2 \end{bmatrix} \ltimes \dots \ltimes \begin{bmatrix} X_n \\ \neg X_n \end{bmatrix}.$$

2 Static output feedback control of BCNs

A *Boolean control network* (BCN) is described by the following equations

$$\begin{aligned} X(t+1) &= f(X(t), U(t)), \\ Y(t) &= h(X(t)), \qquad t \in \mathbb{Z}_+, \end{aligned} \tag{1}$$

where $X(t)$, $U(t)$ and $Y(t)$ denote the n-dimensional state variable, the m-dimensional input and the p-dimensional output at time t, taking values in $\mathcal{B}^n, \mathcal{B}^m$ and $\mathcal{B}^p$, respectively. f, h are (logic) functions, i.e. $f : \mathcal{B}^n \times \mathcal{B}^m \to \mathcal{B}^n$ and $h : \mathcal{B}^n \to \mathcal{B}^p$. If we replace the state, input and output Boolean variables with their vector forms, namely with canonical vectors in $\mathcal{L}_N$, $N := 2^n$, $\mathcal{L}_M$, $M := 2^m$, and $\mathcal{L}_P$, $P := 2^p$, respectively, we can describe the BCN (1) by means of the following *algebraic representation* Cheng *et al.* (2011a):

$$\begin{aligned} \mathbf{x}(t+1) &= L \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t), \qquad t \in \mathbb{Z}_+, \\ \mathbf{y}(t) &= H\mathbf{x}(t), \end{aligned} \tag{2}$$

where $\mathbf{x}(t) \in \mathcal{L}_N, \mathbf{u}(t) \in \mathcal{L}_M$ and $\mathbf{y}(t) \in \mathcal{L}_P$. $L \in \mathcal{L}_{N\times NM}$ and $H \in \mathcal{L}_{P\times N}$ are matrices whose columns are canonical vectors of size N and P, respectively. For every value δ_M^j of $\mathbf{u}(t)$, $L \ltimes \mathbf{u}(t) =: L_j$ is a matrix in $\mathcal{L}_{N\times N}$, and hence $L = [L_1\ L_2\ \dots\ L_M]$.
Before proceeding, we want to recall the state feedback stabilization problem. To this end we introduce the concepts of reachability and stabilizability.
Definition 1. Cheng *et al.* (2011a) *Given a BCN (2),* $\mathbf{x}_f = \delta_N^j$ *is* reachable *from* $\mathbf{x}_0 = \delta_N^h$ *if there exists* $\tau \in \mathbb{Z}_+$ *and an input* $\mathbf{u}(t) \in \mathcal{L}_M, t \in [0, \tau-1]$*, that leads the state trajectory from* $\mathbf{x}(0) = \mathbf{x}_0$ *to* $\mathbf{x}(\tau) = \mathbf{x}_f$*.*
$\mathbf{x}_f = \delta_N^j$ is reachable from $\mathbf{x}_0 = \delta_N^h$ if and only if Cheng *et al.* (2011a) there exists $\tau \in \mathbb{Z}_+$ such that the Boolean sum of the matrices $L_i, i \in [1, M]$, i.e. $L_{tot} := \bigvee_{i=1}^M L_i$, satisfies $[L_{tot}^\tau]_{jh} > 0$.
Definition 2. *Cheng and Liu (2009); Cheng* et al. *(2011a); Fornasini and Valcher (2013b) A BCN (2) is* stabilizable *to the state* $\mathbf{x}_e \in \mathcal{L}_N$ *if for every* $\mathbf{x}(0) \in \mathcal{L}_N$ *there exist* $\mathbf{u}(t), t \in \mathbb{Z}_+$*, and* $\tau \in \mathbb{Z}_+$ *such that* $\mathbf{x}(t) = \mathbf{x}_e$ *for every* $t \ge \tau$*.*
The characterization of stabilizability is immediate.
Proposition 1. *Cheng* et al. *(2011a); Fornasini and Valcher (2013b); Li* et al. *(2013) A BCN (2) is stabilizable to* $\mathbf{x}_e \in \mathcal{L}_N$ *if and only if the following conditions hold:*
1) $\mathbf{x}_e$ *is an equilibrium point of the ith subsystem* $\mathbf{x}(t+1) = L_i\mathbf{x}(t)$*, for some* $i \in [1, M]$*, i.e.* $\mathbf{x}_e = L \ltimes \delta_M^i \ltimes \mathbf{x}_e$*;*
2) $\mathbf{x}_e$ *is reachable from every initial state* $\mathbf{x}(0)$*.*

What is more interesting is the fact that if a BCN (2) is stabilizable to the state $\mathbf{x}_e$, then stabilization is achievable by means of a time-invariant state feedback $\mathbf{u}(t) = K\mathbf{x}(t)$ Fornasini and Valcher (2013b); Li *et al.* (2013). Furthermore, such a state feedback law can always be chosen in such a way that $\mathbf{x}_e$ is reached from every other state in a minimal number of steps. The question arises: under what conditions can we stabilize the BCN to $\mathbf{x}_e$ by resorting to an *output* feedback?

Definition 3. *A BCN (2) is* TIOF stabilizable *to the state $\mathbf{x}_e \in \mathcal{L}_N$ if there exists $K_y \in \mathcal{L}_{M\times P}$ such that the output feedback law $\mathbf{u}(t) = K_y\mathbf{y}(t)$, $t \in \mathbb{Z}_+$, drives every $\mathbf{x}(0) \in \mathcal{L}_N$ to the state $\mathbf{x}_e$ in a finite number of steps, namely $\exists \tau \in \mathbb{Z}_+$ such that $\mathbf{x}(t) = \mathbf{x}_e$ for every $t \geq \tau$.*

As for linear state-space models, the TIOF stabilization problem is easy to state, but quite challenging to be solved in a computationally tractable way. Clearly, if K_y defines a TIOF law, then $K = K_yH$ defines a state feedback law. So, a possible way could be that of determining whether the set of all stabilizing state feedback matrices includes at least one matrix expressed as $K = K_yH$ for some $K_y \in \mathcal{L}_{M\times P}$ (see Li and Wang (2013)). In general, the search cannot be restricted to the class of state feedback matrices K that implement paths of minimum length from each state to the equilibrium state $\mathbf{x}_e$ Fornasini and Valcher (2013b) and, consequently, the test has to be performed on the whole set of state feedback matrices (see Example 1, below).

Example 1. *Consider a BCN (2), with $N = 8, M = 2$ and $P = 2$, and suppose that*

$$L_1 := L \ltimes \delta_2^1 = [\delta_8^2\ \delta_8^8\ \delta_8^8\ \delta_8^3\ \delta_8^5\ \delta_8^3\ \delta_8^8\ \delta_8^1],$$

$$L_2 := L \ltimes \delta_2^2 = [\delta_8^1\ \delta_8^3\ \delta_8^6\ \delta_8^5\ \delta_8^7\ \delta_8^7\ \delta_8^6\ \delta_8^8],$$

$$H = [\delta_2^1\ \delta_2^2\ \delta_2^1\ \delta_2^1\ \delta_2^2\ \delta_2^1\ \delta_2^2\ \delta_2^2].$$

The BCN can be represented by the labelled digraph in Fig. 1, obtained by overlapping the two digraphs $\mathcal{D}(L_1)$ and $\mathcal{D}(L_2)$. Light blue thick arrows represent arcs of $\mathcal{D}(L_1)$, red thick dashed arrows represent arcs of $\mathcal{D}(L_2)$. Black continuous arrows stem from states whose associated output is δ_2^1, while red dashed lines stem from states whose output is $y = \delta_2^2$. The state $\mathbf{x}_e = \delta_8^8$ is reachable from every state and $\mathbf{x}_e = L \ltimes \delta_2^2 \ltimes \mathbf{x}_e$. Consequently, both conditions of Proposition 1 are satisfied, and the BCN is stabilizable to $\mathbf{x}_e$.
The stabilizing state feedback matrices that correspond to minimum distance paths from each δ_8^j to $\mathbf{x}_e = \delta_8^8$ are

$$K_1 = [\delta_2^1\ \delta_2^1\ \delta_2^1\ \delta_2^1\ \delta_2^2\ \delta_2^1\ \delta_2^1\ \delta_2^2],$$

$$K_2 = [\delta_2^1\ \delta_2^1\ \delta_2^1\ \delta_2^1\ \delta_2^2\ \delta_2^2\ \delta_2^1\ \delta_2^2].$$

Neither of these matrices can be expressed as $K_i = K_{y_i}H$ for some $K_{y_i} \in \mathcal{L}_{2\times 2}$. Indeed, as the last two columns of H are both equal to δ_2^2, every matrix expressed as $K_{y_i}H$ has the last two columns that coincide, but this not the case either for K_1 or for K_2. On the other hand, it is easy to see that $K_y = [\delta_2^1\ \delta_2^2] = I_2$ (corresponding to $K_yH = H$, i.e. $\mathbf{u}(t) = \mathbf{y}(t)$) is the only stabilizing TIOF matrix. Both state feedback matrices K_1 and K_2 drive all states to $\mathbf{x}_e$ in at most 2 steps, while the TIOF associated with K_y drives one state (δ_8^5) to $\mathbf{x}_e$ in 4 steps. ♠

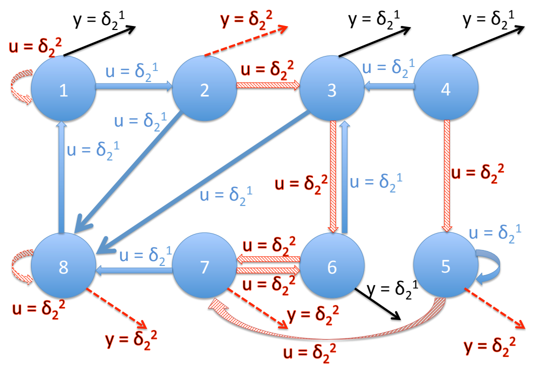

FIG. 1. Digraph corresponding to the BCN of Example 1.

Given $\mathbf{y} \in \mathcal{L}_P$, we denote by $\mathcal{X}(\mathbf{y}) := \{\delta_N^j : H\delta_N^j = \mathbf{y}\}$, the *indistinguishability class in 1 step* corresponding to the output value $\mathbf{y}$ Fornasini and Valcher (2013a), namely the set of all states that produce the output value $\mathbf{y}$. A necessary (but not sufficient) condition for TIOF stabilization is given in the following proposition.

Proposition 2. *Given a BCN (2), a necessary condition for the existence of a TIOF stabilizing the BCN to $\mathbf{x}_e$ is that there exists an input value $\bar{\mathbf{u}} = \delta_M^i$ such that $\mathbf{x}_e$ is the only equilibrium point of the ith subsystem $\mathbf{x}(t+1) = L \ltimes \bar{\mathbf{u}} \ltimes \mathbf{x}(t) = L_i\mathbf{x}(t)$ belonging to $\mathcal{X}(H\mathbf{x}_e)$.*

Example 2. *Consider a BCN (2), with $N = 4, M = 2$, $P = 2$, $L_1 := L \ltimes \delta_2^1 = [\delta_4^1\ \delta_4^2\ \delta_4^4\ \delta_4^1]$, $L_2 := L \ltimes \delta_2^2 = [\delta_4^4\ \delta_4^3\ \delta_4^3\ \delta_4^3]$, $H = [\delta_2^1\ \delta_2^1\ \delta_2^2\ \delta_2^2]$.*

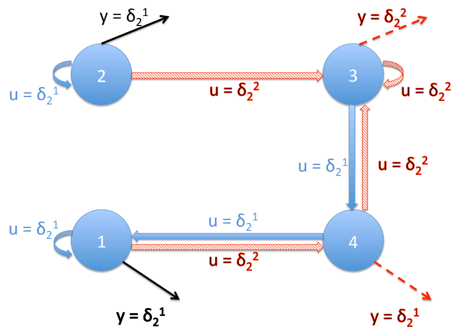

FIG. 2. Digraph corresponding to the BCN of Example 2.

The BCN can be represented by the labelled digraph in Fig. 2. $\mathbf{x}_e = \delta_4^1$ is reachable from every state and $\mathbf{x}_e = L \ltimes \delta_2^1 \ltimes \mathbf{x}_e$. Consequently, both conditions of Proposition 1 are satisfied, and the BCN is state feedback stabilizable to $\mathbf{x}_e$. However, a TIOF stabilizing the BCN to the state δ_4^1 does not exist, since the previous necessary condition is not satisfied. Indeed, the only input value that keeps the system in the equilibrium state is $\bar{\mathbf{u}} = \delta_2^1$. However, $\delta_4^2 \in \mathcal{X}(H\mathbf{x}_e) = \{\delta_4^i : H\delta_4^i = \delta_2^1\}$ is an equilibrium point of the BCN corresponding to the same input value. ♠

To further explore the existence of a TIOF stabilizing the BCN (2) to $\mathbf{x}_e$, we introduce the following non-restrictive assumptions: $\mathbf{x}_e = \delta_N^1$ and $H = \text{diag}\{\mathbf{1}_{n_1}^\top, \mathbf{1}_{n_2}^\top, \dots, \mathbf{1}_{n_P}^\top\}$, with $n_1+n_2+\cdots+n_P = N$. Indeed, every BCN (2) can always be reduced to this situation by resorting to suitable permutations of the state and output components, and possibly by removing output values that never occur (this is the case if H has zero rows)[1]. In this set-up, Theorem 1 in Li and Wang (2013) can be restated, in slightly revised terms, as follows.

Proposition 3. *The BCN (2) is TIOF stabilizable to* $\mathbf{x}_e = \delta_N^1$ *if and only if there exists* $K_y \in \mathcal{L}_{M\times P}$ *such that*

$$(L \ltimes K_y H\Phi_N)^N = [\delta_N^1\ \delta_N^1\ \dots\ \delta_N^1] = \delta_N^1 \mathbf{1}_N^T, \qquad (3)$$

where Φ_N *is the so-called* power reducing matrix *Cheng* et al. *(2011a), i.e. the logical matrix satisfying* $\mathbf{x}(t) \ltimes \mathbf{x}(t) = \Phi_N \mathbf{x}(t)$*, for every* $\mathbf{x}(t) \in \mathcal{L}_N$*.*

It is worth to comment on the meaning of condition (3). The effect of a TIOF is that of converting a BCN into a BN, namely a logic network devoid of external inputs. The $N\times N$ transition matrix of such a BN is just $L \ltimes K_y H\Phi_N$. On the other hand, such a TIOF stabilizes the resulting BN to the state $\mathbf{x}_e$ if and only if $\mathbf{x}_e$ is a global attractor for the BN, which means that in at most N steps every state of the BN reaches $\mathbf{x}_e = \delta_N^1$. This result is analogous to what happens with state feedback (see Remark 1 in Fornasini and Valcher (2013b)).

If we analyze the structure of the logical matrix $L \ltimes K_y H\Phi_N$, we observe that it has the following structure

$$L \ltimes K_y H\Phi_N = [\text{blk}_1(L_{i_1})\ \text{blk}_2(L_{i_2})\ \dots\ \text{blk}_P(L_{i_P})],$$

where $i_1, \dots, i_P \in [1, M]$ are the indices[2] appearing in

$$K_y = [\delta_M^{i_1}\ \delta_M^{i_2}\ \dots\ \delta_M^{i_P}] \qquad (4)$$

and $\text{blk}_k(L_{i_k})$ is the $N \times n_k$ matrix obtained by selecting the columns of L_{i_k} with indices in the interval $[(\sum_{\ell=1}^{k-1} n_\ell)+1, (\sum_{\ell=1}^{k-1} n_\ell)+n_k]$. On the other hand, condition (3) is satisfied (i.e., $\mathbf{x}_e = \delta_N^1$ is globally attractive) if and only if

$$L \ltimes K_y H\Phi_N = \begin{bmatrix} 1 & * \\ 0_{N-1} & \tilde{N} \end{bmatrix},$$

($*$ denoting a Boolean row vector) for some nilpotent matrix $\tilde{N}$ of size $N-1$. Therefore $K_y \in \mathcal{L}_{M\times P}$, described as in (4), stabilizes the BCN to δ_N^1 if and only if

$$[\text{blk}_1(L_{i_1})\ \text{blk}_2(L_{i_2})\ \dots\ \text{blk}_P(L_{i_P})] = \begin{bmatrix} 1 & * \\ 0_{N-1} & \tilde{N} \end{bmatrix} \qquad (5)$$

for some nilpotent $\tilde{N}$. Also, $\tilde{N}$ is nilpotent if and only if its leading principal submatrices Horn and Johnson (1985) are nilpotent[3]. This suggests an algorithm to find the P-tuple (if any) corresponding to a stabilizing K_y.

Algorithm 1: [Initialization] Set $k = 1$ and $i_k := 1$.

[Step 1] If the first column of L_{i_1} is δ_N^1 and the $(n_1-1)\times(n_1-1)$ principal submatrix of L_{i_1} with row and column indices $[2, n_1]$ is nilpotent[4], then set $k := 2$, $i_k := 1$ and go to Step 2. Otherwise set $i_1 := i_1+1$. If $i_1 \le M$, repeat Step 1, otherwise STOP: there is no solution.

[Step 2] Case 1: If $k > P$, then STOP: the problem is solvable and $K_y = [\delta_M^{i_1}\ \delta_M^{i_2}\ \dots\ \delta_M^{i_P}]$ is one solution.
Case 2: If $k \le P$ and $i_k \le M$, then check if the principal submatrix of $\left[\text{blk}_1(L_{i_1})\ \text{blk}_2(L_{i_2})\ \dots\ .\ \text{blk}_k(L_{i_k})\right]$ with row and column indices $[2, \sum_{j=1}^k n_j]$ is nilpotent. If so, set $k := k+1$, $i_k := 1$, otherwise set $i_k := i_k+1$. Repeat Step 2.
Case 3: If $k \le P$ and $i_k > M$, set $k := k-1$ and $i_k := i_k+1$. If $k = 1$ repeat Step 1, otherwise repeat Step 2.

Remark 1. *The algorithm tries to set a value in* $[1, M]$ *for each index* $i_1, i_2, \dots, i_P$*, namely to assign an input value to each output value, so that* $\tilde{N}$ *in (5) is nilpotent. If at some stage* k*, no choice for* i_k *is available that is compatible with* $(i_1, i_2, \dots, i_{k-1})$ *then the value of* i_{k-1} *is updated and hence increased by one. On the other hand if* i_{k-1} *is already equal to the maximum value* M*, we update* i_{k-2}*, and so on. If, at some point, the algorithm backsteps till the level* $k = 1$ *and the value of* i_1 *is already the maximum possible, the algorithm terminates with a negative answer. Finally, note that in Step 2, Case 2, one could preliminarily verify whether the principal submatrix of* L_{i_k} *with row and column indices* $[\sum_{j=1}^{k-1} n_j + 1, \sum_{j=1}^k n_j]$ *is nilpotent. If not, the current value of* i_k *is unsuitable. This allows to discard some indices by performing a check on quite smaller matrices.*

Remark 2. *Algorithm 1 is a branch and bound algorithm that explores all possible assignments for the indices* $i_1, i_2, ..., i_P \in [1, M]$*, until either* one *$P$-tuple is obtained for which* $K_y = [\delta_M^{i_1}\ \delta_M^{i_2}\ \dots.\ \delta_M^{i_P}]$ *stabilizes the BCN to* δ_N^1 *or a negative answer is given. Since there are precise requirements, related to the need for obtaining nilpotent submatrices, some values of the indices may*

[1] If we refer to the original BCN description (1), in terms of Boolean and not canonical vectors, this set-up canot be achieved by means of simple component permutations of the vectors X and Y, but requires state and output coordinate transformations $X' = \phi(X)$ and $Y' = \psi(Y)$, where ϕ and ψ are suitable logic functions (see Cheng *et al.* (2011a)).

[2] In the sequel, we will often identify a matrix K_y described as in (4) with the P-tuple of indices $(i_1, \dots, i_P) \in [1, M]^{[1,P]}$.

[3] Each column of $\tilde{N}$ is either zero or a canonical vector, and nilpotency requires that at least one column is zero. If $\tilde{N}_{i_1,j_1}, \tilde{N}_{i_2,j_2}, \dots, \tilde{N}_{i_k,j_k}$ are the unique nonzero elements of $\tilde{N}$, consider the map ϕ from $\{0, j_1, \dots, j_k\}$ into itself, given by $\phi(0) = 0$, $\phi(j_\nu) = \begin{cases} 0 & \text{if } i_\nu \notin \{j_1, \dots j_k\} \\ i_\nu & \text{if } i_\nu \in \{j_1, \dots j_k\} \end{cases}$. $\tilde{N}$ is nilpotent if and only if ϕ^k maps every element in 0 or, equivalently, if the digraph associated with $\tilde{N}$ has no cycles.

[4] This selection criterion for i_1 is nothing but the necessary condition we have given in Proposition 2.

be rejected at an early stage and hence in the average case only a small subset of the M^P possible choices are considered. It is also clear that, in principle, malicious examples could be constructed for which the branch and bound technique leads to consider all possible P-tuples and hence the worst case may be encountered. Note that, due to the order according to which the indices are considered, when a solution is available the algorithm always provides the specific solution *associated with the P-tuple $(i_1, i_2, \dots, i_P)$ that is the smallest, according to the lexicographic order, among the feasible solutions.*

A different approach to the problem solution, whose merit is that of providing *all the TIOF matrices* that stabilize the BCN to $\mathbf{x}_e$, is based on the labelled digraph associated with the BCN. The underlying idea is that of determining all the elementary cycles in the overall labelled digraph that are "TIOF-compatible". A path (in particular, a cycle) γ of length k in the labelled digraph can be represented as[5] $\mathbf{x}_1 \xrightarrow{\mathbf{u}_1} \mathbf{x}_2 \xrightarrow{\mathbf{u}_2} \dots \xrightarrow{\mathbf{u}_k} \mathbf{x}_{k+1}$, and is called TIOF-compatible if $H\mathbf{x}_i = H\mathbf{x}_j \Rightarrow \mathbf{u}_i = \mathbf{u}_j$.
The TIOF matrices stabilizing the BCN to $\mathbf{x}_e$ are exactly those that ensure that in the resulting BN no TIOF-compatible cycle, except for a self loop in $\mathbf{x}_e$, appears. If $\mathcal{K}$ is the set of all possible P-tuples taking values in $[1, M]$ that correspond to stabilizing TIOF matrices, Algorithm 2 below initializes $\mathcal{K}$ as $[1, M]^{[1,P]}$. Then it evaluates all the cycles, passing through any state of index $j_1 \neq 1$, that are TIOF-compatible, and removes the corresponding P-tuple(s) from the set $\mathcal{K}$. At the end of the algorithm, the set $\mathcal{K}$ will henceforth coincide with the set of (P-tuples corresponding to) *all TIOF matrices* that stabilize the BCN to $\mathbf{x}_e$. In the algorithm, every path γ of length k, say $\delta_N^{j_1} \xrightarrow{\delta_M^{i_1}} \delta_N^{j_2} \xrightarrow{\delta_M^{i_2}} \dots \xrightarrow{\delta_M^{i_k}} \delta_N^{j_{k+1}}$, is associated with two vectors: $\mathcal{X} := [j_1, j_2, \dots, j_{k+1}]$, which represents the sequence of state indices, and $\mathcal{U} := [i_1, i_2, \dots, i_k]$, which represents the sequence of input indices. γ is TIOF-compatible if and only if $H\delta_N^{j_h} = H\delta_N^{j_\ell} \Rightarrow i_h = i_\ell$. Also, if γ is TIOF-compatible, we introduce a P-dimensional vector $\mathcal{T}$ whose elements are set according to the following rule: $[\mathcal{T}]_\nu = 0$ if no state in γ generates the output value δ_P^ν, and $[\mathcal{T}]_\nu = i$ if each state in γ that generates the output value δ_P^ν has an outgoing arc corresponding to the input δ_M^i. If we extend γ by adding a new arc and a new vertex, then we obtain a path γ' of length $k+1$, associated with $\mathcal{X}' := [j_1, j_2, \dots, j_{k+1}, j_{k+2}]$ and $\mathcal{U}' := [i_1, i_2, \dots, i_k, i_{k+1}]$. We can evaluate whether γ' is in turn TIOF-compatible by simply checking that either $[\mathcal{T}]_\nu = 0$ or $[\mathcal{T}]_\nu = i_{k+1}$, where $\delta_N^\nu := H\delta_N^{j_{k+2}}$. We now introduce the algorithm. Further comments will be provided at the end of it.

Algorithm 2: [Initialization] The set of all P-tuples describing potentially stabilizing TIOF matrices (4) is initialized as $\mathcal{K} := \{(i_1, i_2, \dots, i_P) \; : \; i_k \in [1, M]\}$, $|\mathcal{K}| = M^P$. Let $\mathcal{I}_e := \{i \in [1, M] \; : \; L \ltimes \delta_M^i \ltimes \delta_N^1 = \delta_N^1\}$ be the set of the indices of the inputs that correspond to self-loops on $\mathbf{x}_e = \delta_N^1$. As $H\delta_N^1 = \delta_P^1$, all P-tuples $(i_1, i_2, \dots, i_P)$ with $i_1 \not\in \mathcal{I}_e$ are not stabilizing. Therefore set $\mathcal{K} := \mathcal{K} \setminus \{(i_1, i_2, \dots, i_P) \in \mathcal{K} \; : \; i_1 \not\in \mathcal{I}_e\}$.
Note that no elementary cycle including δ_N^1, apart from self-loops, is compatible with any of the remaining P-tuples. We now determine, for every $j_1 \neq 1$, all the TIOF-compatible cycles passing through the state of index j_1, and remove the corresponding P-tuples from $\mathcal{K}$. Set $j_1 := 2$, $\mathcal{X} := [j_1]$, $\mathcal{U} := []$, and $\mathcal{T} := [0, \; 0, \dots, \; 0]$. Introduce the variables L, the current length of the path (i.e. of vector $\mathcal{X}$), and u_{curr}, the next input index to be used in the search for the cycles, and initialize them as follows: $L := 1$ and $u_{curr} := 1$.

[Step 1]: Set $\ell := [\mathcal{X}]_L$ (this means that ℓ is the last vertex of our current path) and let $\nu \in [1, P]$ and $f \in [1, N]$ be such that $\delta_P^\nu = H\delta_N^\ell$ and $\delta_N^f = L \ltimes \delta_M^{u_{curr}} \ltimes \delta_N^\ell$ (we identify the output associated with δ_N^ℓ and the next state, corresponding to the input index u_{curr}).
Case 1: $(f = j_1) \wedge ([\mathcal{T}]_\nu = u_{curr} \; \vee \; [\mathcal{T}]_\nu = 0)$ (i.e., the path is a TIOF-compatible cycle). Update $\mathcal{K} := \mathcal{K} \setminus \{(i_1, i_2, \dots, i_P) \in \mathcal{K} \mid i_\nu = u_{curr}, i_r = [\mathcal{T}]_r, \; \forall r \in [1, P], r \neq \nu$, with $[\mathcal{T}]_r \neq 0\}$ and go to Step 2.
Case 2: $(f > j_1) \; \wedge \; (f$ is not an element of $\mathcal{X}) \; \wedge \; ([\mathcal{T}]_\nu = u_{curr} \; \vee \; [\mathcal{T}]_\nu = 0)$ (i.e., we can extend our TIOF-compatible path to a new TIOF-compatible one, by adding the arc corresponding to $\delta_M^{u_{curr}}$ and the vertex f). Update $\mathcal{X} := [\mathcal{X}, \; f]$, $\mathcal{U} := [\mathcal{U}, \; u_{curr}]$, $[\mathcal{T}]_\nu := u_{curr}$, $L := L + 1$, and $u_{curr} := 1$ and repeat Step 1.
Case 3: $(f < j_1) \; \vee \; (f > j_1$ and f is already an element of $\mathcal{X}) \; \vee \; ([\mathcal{T}]_\nu \neq u_{curr} \; \wedge \; [\mathcal{T}]_\nu \neq 0)$. Go to Step 2.

[Step 2]: Set $u_{curr} := u_{curr} + 1$.
Case 1: $u_{curr} \leq M$ (and hence the extension of the path ending at $[\mathcal{X}]_L$ using the input indexed by u_{curr} has still to be explored). Go to Step 1.
Case 2: $(u_{curr} = M + 1) \wedge (L > 1)$ (all paths extension from $[\mathcal{X}]_L$ have been explored but there may be other path extensions to be explored from $[\mathcal{X}]_{L-1}$). Then set $\mathcal{X} := [\mathcal{X}]_{1:L-1}$ and $L := L - 1$ (the last vertex is eliminated from the path). Set $u_{curr} := [\mathcal{U}]_L$, $\mathcal{U} := [\mathcal{U}]_{1:L-1}$, (also the last arc is eliminated). If $f = [\mathcal{X}]_L$ is the only state in $\mathcal{X}$ such that $H \ltimes \delta_N^f = \delta_P^\nu$, set $[\mathcal{T}]_\nu := 0$. Go to Step 2.
Case 3: $(u_{curr} = M + 1) \; \wedge \; (L = 1)$ (all paths starting from j_1 have been analyzed). Set $j_1 := j_1 + 1$, $\mathcal{X} := [j_1]$, $\mathcal{U} := []$, $\mathcal{T} := [0, \; 0, \dots, \; 0]$, $L := 1$ and $u_{curr} := 1$. If $j_1 \leq N$ go to Step 1, otherwise go to Step 3.

[Step 3]: If $\mathcal{K}$ is void the problem is unsolvable, otherwise $\mathbf{K} := \{[\delta_M^{i_1} \; \dots \; \delta_M^{i_P}] \mid (i_1, \dots, i_P) \in \mathcal{K}\}$ is the set of *all TIOF stabilizing matrices.*

After the initialization phase, the set $\mathcal{K}$ contains only the P-tuples corresponding to TIOF matrices K_y whose resulting BN has $\mathbf{x}_e = \delta_N^1$ as an equilibrium point. Subsequently, the algorithm searches for all elementary cycles passing through the vertices j_1, with $j_1 \in [2, N]$. To this end it constructs all paths leaving from j_1. Case 1 in Step 1 corresponds to the situation when the sequence

[5] This notation means that for every $i \in [1, k]$ we have $\mathbf{x}_{i+1} = L \ltimes \mathbf{u}_i \ltimes \mathbf{x}_i$. If γ is a cycle then $\mathbf{x}_{k+1} = \mathbf{x}_1$.

$\mathcal{X}$ of states represents a TIOF-compatible elementary cycle, and hence the associated P-tuples (obtained from $\mathcal{T}$) are removed from $\mathcal{K}$. Case 2 of Step 1 corresponds to the case when the sequence $\mathcal{X}$ is TIOF-compatible but it does not represent a cycle, so Step 1 is iterated. In Case 3 of Step 1, the vertex f cannot be added to the sequence $\mathcal{X}$ because one of the following problems arises: (a) the resulting path is not TIOF-compatible; (b) $f \neq j_1$ already belongs to $\mathcal{X}$, but then the path cannot be completed to an elementary cycle; (c) $f < j_1$ and hence it has been considered in a previous iteration. In Step 2 the input index is updated and a check is performed to understand whether a path extension from the current f is possible, or a step back to the previous vertex in $\mathcal{X}$ is required, or, finally, a new starting vertex j_1 has to be considered. Finally, Step 3 returns *all P-tuples corresponding to stabilizing TIOF matrices K_y*, if any.

3 Time-varying output feedback stabilization

As we have seen, output feedback stabilization is quite challenging and often difficult to achieve. In order to derive alternative means to solve the feedback stabilization problem when only output values are available, it is worthwhile to provide a graph theoretic interpretation of the effects that various feedback laws produce on a given BCN. State feedback converts a BCN, in which each state exhibits M outgoing arcs (describing the state transitions corresponding to the M different values of the input), into a BN by selecting for each state which outgoing arc to preserve. This choice is in principle completely arbitrary for every state. If the target of the state feedback is the stabilization to a given state $\mathbf{x}_e$, all arcs must be chosen (if possible) in such a way that, in the final BN, all paths lead to $\mathbf{x}_e$, and $\mathbf{x}_e$ exhibits a self loop. In order to achieve this goal, conditions 1) and 2) of Proposition 1 must be satisfied. On the other hand, these conditions are also sufficient for the existence of a time-invariant state feedback law stabilizing the BCN to $\mathbf{x}_e$ and, in addition, for implementing a shortest path strategy. So, a time-varying state feedback law does not offer any advantage, in particular it does not allow to stabilize when a time-invariant solution does not exist.
When dealing with output feedback, on the other hand, the selection of the outgoing arc for each state is not arbitrary. Indeed, once an outgoing arc, and hence an input value $\mathbf{u} = \delta_M^j$, has been chosen for a specific state $\mathbf{x} = \delta_N^i$, the arc corresponding to the same value of $\mathbf{u}$ must be chosen for all the states belonging to the indistinguishability class in 1 step $\mathcal{X}(H\delta_N^i)$. Again, if the target is the stabilization to a given state $\mathbf{x}_e$, in the final BN, all paths must lead to $\mathbf{x}_e$, and $\mathbf{x}_e$ must exhibit a self loop. However, as the arc selection procedure is much more restrictive, output stabilization may not be achievable even if conditions 1) and 2) of Proposition 1 hold. Moreover, in this case a time varying selection of which kind of arc to associate with each indistinguishability class may offer more flexibility, and hence allow for stabilization even in cases when TIOF would not. Indeed, there are situations, when the output feedback stabilization problem cannot be solved by resorting to a time-invariant solution, but it can be solved by adopting a TVOF law $\mathbf{u}(t) = K_y(t)\mathbf{y}(t), t \in \mathbb{Z}_+$.

Example 3. *In the BCN of Example 2 we have seen that a TIOF stabilizing the BCN to $\mathbf{x}_e = \delta_4^1$ does not exist. However, the TVOF $K_y(0) = [\delta_2^2\ \delta_2^2]$, $K_y(t) = [\delta_2^1\ \delta_2^1], \forall\, t \geq 1$, stabilizes the BCN to $\mathbf{x}_e$.* ♠

The idea behind the previous example can be generalized. Let $\mathcal{S}$ be the set of all the states that can reach $\mathbf{x}_e$ by resorting to a constant input $\bar{\mathbf{u}}$ that leaves $\mathbf{x}_e$ invariant. If $\mathcal{S}$ can in turn be reached in exactly T steps from all the N states, by resorting to some assigned input sequence (which is independent of the initial state), then TVOF stabilization is possible.

Proposition 4. *Assume that $\mathbf{x}_e = \delta_N^1$ is an equilibrium point of the BCN (2) corresponding to $\bar{\mathbf{u}} = \delta_M^{i^*}$, and let $\mathcal{A}_{\bar{\mathbf{u}}}(\mathbf{x}_e)$ be the domain of attraction[6] of $\mathbf{x}_e$ in the BN $\mathbf{x}(t+1) = L \ltimes \bar{\mathbf{u}} \ltimes \mathbf{x}(t) = L_{i^*}\mathbf{x}(t), t \in \mathbb{Z}_+$.*
If there exist $T \in \mathbb{Z}_+$ and an input sequence $\tilde{\mathbf{u}}(t), t \in [0, T-1]$, such that for every $\mathbf{x}(0) \in \mathcal{L}_N$ the state trajectory stemming from $\mathbf{x}(0)$ under the action of the previous input sequence satisfies $\mathbf{x}(T) \in \mathcal{A}_{\bar{\mathbf{u}}}(\mathbf{x}_e)$, then there exists a TVOF stabilizing the BCN to $\mathbf{x}_e$.

Proof. Let $i_0, i_1, \dots, i_{T-1}$ be indices in $[1, M]$, such that $\tilde{\mathbf{u}}(t) = \delta_M^{i_t}, t \in [0, T-1]$. Clearly, the input sequence

$$\mathbf{u}(t) := \begin{cases} \delta_M^{i_t}, & t \in [0, T-1]; \\ \delta_M^{i^*}, & t \geq T; \end{cases}$$

ensures that, for every $\mathbf{x}(0) \in \mathcal{L}_N$ and every $t \geq T + |\mathcal{A}_{\bar{\mathbf{u}}}(\mathbf{x}_e)| - 1$, we have $\mathbf{x}(t) = \mathbf{x}_e$. Then the TVOF law

$$K_y(t) = \begin{cases} [\delta_M^{i_t}\ \delta_M^{i_t}\ \dots\ \delta_M^{i_t}], & t \in [0, T-1]; \\ [\delta_M^{i^*}\ \delta_M^{i^*}\ \dots\ \delta_M^{i^*}], & t \geq T, \end{cases}$$

stabilizes the system to $\mathbf{x}_e$. □

Example 4. *Consider a BCN (2), represented by the labelled digraph of Fig.3, with $N = 8, M = 2$ and $P = 2$*

$$L_1 := L \ltimes \delta_2^1 = [\delta_8^2\ \delta_8^6\ \delta_8^4\ \delta_8^5\ \delta_8^3\ \delta_8^7\ \delta_8^3\ \delta_8^2],$$

$$L_2 := L \ltimes \delta_2^2 = [\delta_8^8\ \delta_8^2\ \delta_8^3\ \delta_8^3\ \delta_8^4\ \delta_8^5\ \delta_8^6\ \delta_8^7],$$

$$H = [\delta_2^1\ \delta_2^2\ \delta_2^1\ \delta_2^1\ \delta_2^2\ \delta_2^2\ \delta_2^1\ \delta_2^2]$$

$\mathbf{x}_e := \delta_8^3$ is a fixed point of the BCN corresponding to the constant input $\bar{\mathbf{u}} = \delta_2^2$, i.e. $\mathbf{x}_e = L \ltimes \delta_2^2 \ltimes \mathbf{x}_e$, and it can be reached from all states of the BCN. Therefore a stabilizing state feedback exists. On the other hand, $\mathbf{x}_e$ is not a fixed point corresponding to $\bar{\mathbf{u}}' = \delta_2^1$, so every stabilizing TIOF matrix K_y should satisfy $K_y\delta_2^1 = K_y(H\mathbf{x}_e) = \bar{\mathbf{u}} = \delta_2^2$. On the other hand, both $K_{y,1} = [\delta_2^2\ \delta_2^1]$ and $K_{y,2} = [\delta_2^2\ \delta_2^2]$ lead to BNs with a second attractor in addition to $\mathbf{x}_e$. So, no stabilizing TIOF exists. However we have $\mathcal{A}_{\delta_2^2}(\delta_8^3) = \{\delta_8^i; i \in [1, 8], i \neq 2\}$ and, if we apply at $t = 0$ and $t = 1$ the input values $\tilde{\mathbf{u}}(0) = \tilde{\mathbf{u}}(1) = \delta_2^1$, independently of the initial state $\mathbf{x}(0)$ we get $\mathbf{x}(2) \in \mathcal{A}_{\delta_2^2}(\delta_8^3)$.

[6] The set of all initial states whose associated state trajectory eventually becomes equal to $\mathbf{x}_e$.

Therefore the TVOF

$$K_y(t) = \begin{cases} [\delta_2^1 \ \delta_2^1], & t \in [0,1]; \\ [\delta_2^2 \ \delta_2^2], & t \geq 2; \end{cases}$$

stabilizes the BCN to $\mathbf{x}_e$*, while no TIOF law does.* ♠

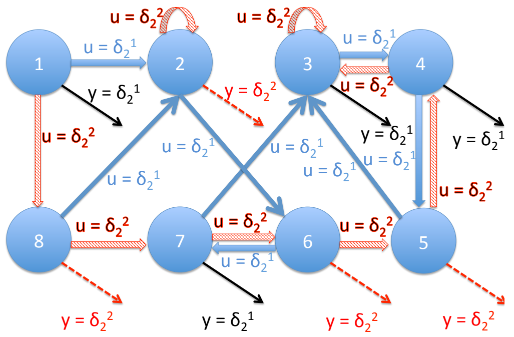

FIG. 3. Digraph corresponding to the BCN of Example 4.

Another sufficient condition for TVOF stabilization assumes that $\mathbf{x}_e$ is reachable from every state by making use only of input sequences whose values "do not alter the equilibrium of $\mathbf{x}_e$" every time the system state belongs to the class $\mathcal{X}(H\mathbf{x}_e)$.

Proposition 5. *Given a BCN (2), let* $\mathbf{x}_e = \delta_N^1$ *be an equilibrium point for the set of input values* $\mathcal{U}_e \subseteq \mathcal{L}_M$ *and assume that* $H\mathbf{x}_e = \delta_P^1$*. If* $\mathbf{x}_e$ *is reachable from every initial state* $\mathbf{x}(0) = \delta_N^j, j \in [1,N]$*, using input sequences* $\mathbf{u}^{(j)}(0), \mathbf{u}^{(j)}(1), \dots$ *that satisfy the constraint*

$$\mathbf{x}^{(j)}(t) \in \mathcal{X}(\delta_P^1) \quad \Rightarrow \quad \mathbf{u}^{(j)}(t) \in \mathcal{U}_e, \tag{6}$$

then there exists a TVOF $K_y(t), t \in \mathbb{Z}_+$*, that drives every* $\mathbf{x}(0) \in \mathcal{L}_N$ *to the state* $\mathbf{x}_e$ *in a finite number of steps.*

Proof. Under the proposition assumptions, a positive integer τ can be found such that every initial state $\mathbf{x}(0) = \delta_N^j, j \in [2,N]$, can be driven to $\mathbf{x}_e$ in τ steps, by resorting to an input sequence $\mathbf{u}^{(j)}(t), t \in [0, \tau-1]$, satisfying (6). Consequently, for each $\mathbf{x}(0) = \delta_N^j,\ j \in [1,N]$, the time-varying output feedback law, defined for $t \in [0, \tau-1]$,

$$K_y^{(j)}(t) = \begin{cases} [\mathbf{u}^{(j)}(t) \ \mathbf{u}^{(j)}(t) \ \dots \ \mathbf{u}^{(j)}(t)], \text{if } \mathbf{x}^{(j)}(t) \in \mathcal{X}(\delta_P^1), \\ [\bar{\mathbf{u}} \ \mathbf{u}^{(j)}(t) \ \dots \ \mathbf{u}^{(j)}(t)], \text{if } \mathbf{x}^{(j)}(t) \notin \mathcal{X}(\delta_P^1), \end{cases}$$

where $\bar{\mathbf{u}}$ is any value in $\mathcal{U}_e$, induces for every $t \in [0, \tau-1]$ a state evolution that satisfies $\mathbf{x}(t) = \mathbf{x}^{(j)}(t)$, when $\mathbf{x}(0) = \delta_N^j$, and $\mathbf{x}(t) = \delta_N^1$, when $\mathbf{x}(0) = \delta_N^1$. Set now

$k = 0, \quad \mathcal{S}_0 := \{\delta_N^i, i \in [2,N]\}$

$j_0 := 2, \quad K_y(t) := K_y^{(j_0)}(t), \ t \in [0, \tau-1].$

Next set

$k = 1, \quad \mathcal{S}_1 := \{\delta_N^i, i \in [2,N] : \delta_N^i \text{is reachable from some } \delta_N^j \in \mathcal{S}_0 \text{ using } \mathbf{u}^{(j_0)}(t), t \in [0, \tau-1]\}.$

Note that $|\mathcal{S}_1| < |\mathcal{S}_0|$. If $\mathcal{S}_1 \neq \emptyset$, set

$j_1 := \min\{i : \delta_N^i \in \mathcal{S}_1\},\ K_y(t) := K_y^{(j_1)}(t-\tau), t \in [\tau, 2\tau-1].$

Next set

$k = 2, \quad \mathcal{S}_2 := \{\delta_N^i, i \in [2,N] : \delta_N^i \text{is reachable from some } \delta_N^j \in \mathcal{S}_1 \text{ using } \mathbf{u}^{(j_1)}(t), t \in [0, \tau-1]\},$

Note that $|\mathcal{S}_2| < |\mathcal{S}_1|$. If $\mathcal{S}_2 \neq \emptyset$, set

$j_2 := \min\{i : \delta_N^i \in \mathcal{S}_2\},\ K_y(t) := K_y^{(j_2)}(t-2\tau), t \in [2\tau, 3\tau-1].$

For some $k \leq N-1$ we have $\mathcal{S}_k = \emptyset$ and the TVOF $K_y(t), t \geq 0$, constructed in this way drives every $\delta_N^i, i \in [2,N]$, to $\mathbf{x}_e$ (and leaves $\mathbf{x}_e$ invariant at every $t \geq 0$).

Remark 3. *The proof of Proposition 5 is elementary and constructive, however the strategy for choosing the matrices* $K_y(t)$ *is not very efficient, as some initial states are driven to* $\mathbf{x}_e$ *in a high number of steps. A more efficient, but more involved, algorithm is presented in Fornasini and Valcher (2014).*

Remark 4. *The sufficient condition given in the previous proposition corresponds to saying that* $\mathbf{x}_e$ *is reachable from every* $\mathbf{x}(0) \in \mathcal{L}_N$*, even if we constrain the input values to belong to* $\mathcal{U}_e \subseteq \mathcal{L}_M$ *every time we encounter a state* $\delta_N^j \in \mathcal{X}(H\mathbf{x}_e)$*. This condition can be verified as follows. Assume, as in the previous section,* $\mathbf{x}_e = \delta_N^1$*,* $H\mathbf{x}_e = \delta_P^1$*,* $\mathcal{X}(\delta_P^1) = [1, n_1]$*, and* $\mathcal{I}_e = \{i \in [1,M] : L \ltimes \delta_M^i \ltimes \mathbf{x}_e = \mathbf{x}_e\}$*. Arbitrarily choose* $i^* \in \mathcal{I}_e$*. Introduce the matrices*

$$\tilde{L}_i := \begin{cases} L_i, & \text{if } i \in \mathcal{I}_e; \\ [\text{blk}_1(L_{i^*}) \ \text{blk}_{2,\dots,P}(L_i)], & \text{if } i \notin \mathcal{I}_e; \end{cases} \quad i \in [1,M].$$

and set $\tilde{L} := [\tilde{L}_1 \ \tilde{L}_2 \ \dots \ \tilde{L}_M]$*. The sufficient condition of Proposition 5 holds if and only if for the BCN*

$$\mathbf{x}(t+1) = \tilde{L} \ltimes \mathbf{v}(t) \ltimes \mathbf{x}(t), \tag{7}$$

with $\mathbf{v}(t) \in \mathcal{L}_M$*,* $\mathbf{x}_e$ *is reachable from every other state. This is true (see section II) if and only if for each* $j \in [1,N]$ *there exists* $\tau \in \mathbb{Z}_+$ *such that* $[\tilde{L}_{tot}^\tau]_{1j} > 0$*, where* $\tilde{L}_{tot}$ *is the Boolean sum* $\tilde{L}_{tot} := \tilde{L}_1 \vee \tilde{L}_2 \vee \dots \vee \tilde{L}_M$*.*

4 *Lac* Operon model

In this section we show how the output feedback techniques previously introduced can be applied to control the dynamics of the *lac* Operon in the bacterium *Escherichia Coli.* When this bacterium has lactose, but no glucose, at its disposal, the genes of the Operon are expressed, and the bacterium is able to metabolize the lactose. This Operon has been extensively investigated in the literature and different types of models have been proposed to describe it. The model adopted in this section can be found in Veliz-Cuba and Stigler (2011) and involves 13 variables:

- M: the *lac* Operon mRNA;
- C: the catabolite activator protein (CAP);
- P: the permease protein;
- B: the β-galactosidase protein;
- G_e: the extracellular glucose;
- R and R_m: are used to represent different concentration levels of the *lac* Operon repressor;
- A and A_m: are used to represent different concentration levels of the allolactose;
- L and L_m: are used to represent different concentration levels of the intracellular lactose;
- L_e and L_{em}: are used to represent different concentration levels of the extracellular lactose.

Each of the last four pairs of variables is used to represent three different concentration levels (low, medium and high) of a specific substance, and hence only three of the four values that the pair may take are allowed. The model derived in Veliz-Cuba and Stigler (2011) is the following one:

$$\begin{cases} M(t+1) = C(t) \wedge \bar{R}(t) \wedge \bar{R}_m(t) \\ P(t+1) = M(t) \\ B(t+1) = M(t) \\ C(t+1) = \bar{G}_e(t) \\ R(t+1) = \bar{A}(t) \wedge \bar{A}_m(t) \\ R_m(t+1) = (\bar{A}(t) \wedge \bar{A}_m(t)) \vee R(t) \\ A(t+1) = B(t) \wedge L(t) \\ A_m(t+1) = L(t) \vee L_m(t) \\ L(t+1) = P(t) \wedge L_e(t) \wedge \bar{G}_e(t) \\ L_m(t+1) = [(L_{em}(t) \wedge P(t)) \vee L_e(t)] \wedge \bar{G}_e(t). \end{cases}$$

Extracellular glucose (G_e) and extracellular lactose (L_e and L_{em}) are not influenced by the other variables, and represent the system inputs. To obtain the algebraic representation of this Boolean model, we introduce the state and input variables:

$$\begin{aligned} \mathbf{x}(t) &= M^v(t) \ltimes B^v(t) \ltimes R^v(t) \ltimes A^v(t) \ltimes L^v(t) \ltimes P^v(t) \\ &\quad \ltimes C^v(t) \ltimes R^v_m(t) \ltimes A^v_m(t) \ltimes L^v_m(t), \\ \mathbf{u}(t) &= G^v_e(t) \ltimes L^v_{em}(t) \ltimes L^v_e(t), \end{aligned}$$

where the superscript v has been used to denote that each Boolean variable has been replaced by the corresponding vector form. Since for some Boolean pairs only 3 of the 4 values are admissible, the state vector $\mathbf{x}(t)$ belongs to $\mathcal{L}_{432}$ (instead of $\mathcal{L}_{1024}$), while the input $\mathbf{u}(t)$ belongs to $\mathcal{L}_6$ (instead of $\mathcal{L}_8$) (see Veliz-Cuba and Stigler (2011) for more details on this aspect). The model can therefore be rewritten as in (2), with $L \in \mathcal{L}_{432\times 432\cdot 6}$, a matrix which is not given explicitly here due to its size. Since our goal is the output feedback stabilization of the BCN to an equilibrium point, it is necessary to preliminarily find all the equilibrium points. To this purpose it is sufficient to determine the equilibrium points of the 6 BNs one obtains by setting $\mathbf{u}$ to a specific value.
Table 1 contains all the equilibria of the BNs, along with the number of states starting from which the state evolution (for that specific value of the input) converges to each of them. Corresponding to $\mathbf{u} = \delta_6^5$, the resulting BN exhibits two distinct equlibria, whose domains of attraction provide a partition of the state set. Each of the BNs corresponding to the other input values has a unique global equilibrium.
We first assume as measurable variables $M(t)$ and $L_m(t)$, and hence the output of the BCN is (see Li and Wang (2013)) $y(t) = M^v(t) \ltimes L^v_m(t)$. Using this output we try to find a stabilizing TIOF law for each fixed point:
• TIOF stabilizing to state δ_{432}^{360}: using the method suggested in Li and Wang (2013), 81 matrices K_y are found. One of them is $K_y = [\,\delta_6^1\ \delta_6^1\ \delta_6^1\ \delta_6^1\,]$, a solution that ensures to reach the equilibrium state at time $t = 3$. If we solve the same problem using a slightly modified version of Algorithm 1, in order to find all the solutions, instead of stopping at the first one, we get 426 different TIOF matrices that stabilize the system to state δ_{432}^{360} (obviously, including those found with the previous method). However, none of them is able to reduce the minimum number of steps below 3.

Input	Equilibrium point	# Domain of attraction
δ_6^1	δ_{432}^{360}	432
δ_6^2	δ_{432}^{360}	432
δ_6^3	δ_{432}^{360}	432
δ_6^4	δ_{432}^{38}	432
δ_6^5	δ_{432}^{81}	8
δ_6^5	δ_{432}^{356}	424
δ_6^6	δ_{432}^{356}	432

TABLE 1: Equilibrium points of the different subsystems of the *lac* Operon model.

• TIOF stabilizing to state δ_{432}^{38}: using the method suggested in Li and Wang (2013), only one TIOF matrix K_y is found, namely $K_y = [\,\delta_6^4\ \delta_6^4\ \delta_6^4\ \delta_6^4\,]$. The same problem has 6 different solutions if it is solved using Algorithm 1. One solution found using the time-varying algorithm presented in Fornasini and Valcher (2014) is:

$K_y(0) = [\,\delta_6^4\ \delta_6^4\ \delta_6^4\ \delta_6^4\,]$;

$K_y(1) = K_y(2) = K_y(3) = K_y(4) = [\,\delta_6^4\ \delta_6^1\ \delta_6^4\ \delta_6^1\,]$;

$K_y(t) = [\,\delta_6^4\ \delta_6^1\ \delta_6^1\ \delta_6^1\,], \qquad \forall\, t \geq 5.$

The matrix K_y derived in Li and Wang (2013), as well as the matrices derived through Algorithm 1 and the TVOF matrices, lead to the equilibrium in $t = 8$ steps.
• TIOF stabilizing to state δ_{432}^{356}: the method proposed in Li and Wang (2013) does not give any solution, whereas the method proposed in this article gives 251 solutions, among them the TIOF matrix: $K_y = [\,\delta_6^1\ \delta_6^1\ \delta_6^1\ \delta_6^5\,]$, that leads to the equilibrium point in (at most) $t = 8$ steps, and this is the best possible performance.
• output feedback stabilizing to state δ_{432}^{81}: this problem cannot be solved by a TIOF. However, Proposition 4 applies, thus leading to the following TVOF law: $K_y(t) = [\,\delta_6^4\ \delta_6^4\ \delta_6^4\ \delta_6^4\,]$ for $0 \leq s \leq 7$ and $K_y(t) = [\,\delta_6^5\ \delta_6^5\ \delta_6^5\ \delta_6^5\,]$ for $t \geq 8$. With this solution, the BCN is stabilized to δ_{432}^{81} at $t = 9$.
However, if the measurable variables are $M(t)$, $L_m(t)$ and $P(t)$, and hence the output is $\mathbf{y}(t) = M^v(t) \ltimes L^v_m(t) \ltimes P^v(t)$, even for state δ_{432}^{81} we can find a TIOF. The method proposed in Li and Wang (2013) gives 8 different TIOF matrices. One of them is $K_y = [\,\delta_6^5\ \delta_6^4\ \delta_6^4\ \delta_6^4\ \delta_6^4\ \delta_6^4\ \delta_6^4\ \delta_6^4\,]$, that drives every state to the equilibrium point in at most 6 steps. Algorithm

1 provides 432 solutions.

Remark 5. *Computations were run using a quad-core processor (each with 8 threads) with a frequency of 3.6 GHz. Algorithm 1 is executed in 8.1s for state 360, in 1.5s for state 38, in 12.2s for state 356. Computing the TVOF for state 81 and for the first input required 2318.9s.*

References

Cheng, D. (2009). Input-state approach to Boolean Networks. *IEEE Trans. Neural Netw.*, **20, (3)**, 512 – 521.

Cheng, D. and Liu, J. (2009). Stabilization of BCNs. In *Proc. Joint 48th IEEE CDC and 28th CCC*, pages 5269–5274, Shanghai, China.

Cheng, D. and Qi, H. (2010). State-space analysis of Boolean networks. *IEEE Trans. Neural Netw.*, **21, (4)**, 584 – 594.

Cheng, D., Qi, H., and Li, Z. (2011a). *Analysis and control of Boolean networks.* Springer-Verlag, London.

Cheng, D., Qi, H., Li, Z., and Liu, J. (2011b). Stability and stabilization of Boolean networks. *Int. J. Robust Nonlin. Contr.*, **21**, 134–156.

Fornasini, E. and Valcher, M.E. (2014). Feedback stabilization, regulation and optimal control of BCNs. In *Proc. 2014 ACC*, pages 1993-1998, Portland, OR.

Fornasini, E. and Valcher, M.E.. (2013a). Observability, reconstructibility and state observers of BCNs. *IEEE Trans. Aut. Contr.*, **58 (6)**, 1390 – 1401.

Fornasini, E. and Valcher, M.E. (2013b). On the periodic trajectories of BCNs. *Automatica*, **49**, 1506–1509.

Horn, R. and Johnson, C. (1985). *Matrix Analysis.* Cambridge Univ. Press, Cambridge (GB).

Li, H. and Wang, Y. (2013). Output feedback stabilization control design for BCNs. *Automatica*, **49**, 3641–3645.

Li, R., Yang, M., and Chu, T. (2013). State feedback stabilization for BCNs. *IEEE Trans. Aut. Contr.*, **58 (7)**, 1853–1857.

Lou, Y. and Hong, Y. (2010). Multi-agent decision in Boolean networks with private information and switching interconnection. In *Proc. 29th CCC*, pages 4530 – 4535, Beijing, China.

Shmulevich, I., Dougherty, E., Kim, S., and Zhang, W. (2002). From Boolean to probabilistic Boolean networks as models of genetic regulatory networks. *Proc. IEEE*, **90, no. 11**, 1778–1792.

Veliz-Cuba, A. and Stigler, B. (2011). Boolean models can explain bistability in the *lac* Operon. *J.Compu. Biology*, **18 (6)**, 783–794.