Automatic Synthesis of Boolean Networks from Biological Knowledge and Data

Athénaïs Vaginay[1,2][0000−0001−5062−7993], Taha Boukhobza[2][0000−0003−1046−3554], and Malika Smaïl-Tabbone[1][0000−0002−8119−2117]

[1] Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
[2] Université de Lorraine, CNRS, CRAN, F-54000 Nancy, France
athenais.vaginay@loria.fr

Abstract. Boolean Networks (BNs) are a simple formalism used to study complex biological systems when the prediction of exact reaction times is not of interest. They play a key role to understand the dynamics of the studied systems and to predict their disruption in case of complex human diseases. BNs are generally built from experimental data and knowledge from the literature, either manually or with the aid of programs. The automatic synthesis of BNs is still a challenge for which several approaches have been proposed. In this paper, we propose `ASKEeD-BN`, a new approach based on Answer-Set Programming to synthesise BNs constrained in their structure and dynamics. By applying our method on several well-known biological systems, we provide empirical evidence that our approach can construct BNs in line with the provided constraints. We compare our approach with three existing methods (`REVEAL`, `Best-Fit` and `caspo-TS`) and show that our approach synthesises a small number of BNs which are covering a good proportion of the dynamical constraints, and that the variance of this coverage is low.

Keywords: Boolean Network Synthesis · Answer-Set Programming.

1 Introduction

Models of biological systems are important to understand the underlying processes in living organisms [10]. Once built, the model is an artefact that can be used to study a system through simulation. Several formalisms have been proposed to model biological systems [11], and they all have their own strengths and weaknesses. The choice of a formalism is guided by the question at hand: the best formalism is the most abstract formalism which can answer the question [3]. For example, differential equations are a formalism suited to run detailed dynamic simulations because they contain information on kinetic parameters. However, they do not scale to large systems.

Boolean Networks (BNs) are a formalism used to study complex biological systems where prediction of exact reaction times is not of interest [1]. They play a key role to understand the dynamics of biological systems and predict their disruption in case of complex human diseases [2]. The key notions of BNs are presented in Section 2.2. BNs are built from available knowledge about the structure of the system and data about the behaviour of its components (Section 2.3). The knowledge and data are used as constraints for the BN synthesis. The automatic synthesis of BNs from biological data and knowledge is still a challenge for which several methods have been developed. In Section 3, we review three state-of-the-art approaches: `REVEAL`, `Best-Fit` and `caspo-TS`.

In Section 4, we present `ASKEeD-BN`, a new automatic approach for the synthesis of BNs constrained in their structure and dynamics. We rely on the Answer-Set Programming framework to generate non-redundant BNs fulfilling the given constraints. We compare the performances of our approach with `REVEAL`, `Best-Fit` and `caspo-TS` on several biological systems with experimental and synthetic data (Section 5). Finally, we discuss the results and conclude.

2 Boolean Networks and their Synthesis

2.1 Prior Knowledge Network (PKN)

Part of the knowledge one has about a biological system is the list of components (genes, proteins...) constituting the system and how these components influence each other. Influences have a **polarity**: activation (polarity "+") or inhibition (polarity "−"). The **parents** of a component X are the components which influence X. A **Prior Knowledge Network** (PKN) encodes this knowledge. The nodes of the network are the components of the system. The edges are directed from parent components to child components and labelled "+" or "−" according to the polarity of the influences. Fig. 1 shows an example PKN for a system of three components. In this PKN, C and A are the parents of C.

- "A activates C"
- "B interacts with itself"
- "C activates A"
- "C interacts with B"
- "C inhibits itself"

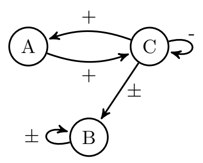


Fig. 1. PKN example of a three-components system.

2.2 Boolean Networks (BNs)

BNs were introduced by Kauffman [7] to model genetic regulatory networks. Concepts used in BNs are described in a recent review [17]. Two examples of BNs are given in Fig. 2.

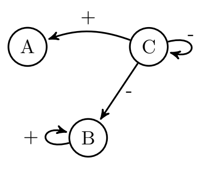
The components of a BN are the components of the considered biological system. For example, a BN modelling a system of three proteins called A, B and C has three components named A, B and C. A **configuration** of a BN

$$\mathcal{B}_1 = \begin{cases} f_\mathrm{A} := \mathrm{C} \\ f_\mathrm{B} := \mathrm{B} \land \neg\mathrm{C} \\ f_\mathrm{C} := \neg\mathrm{C} \end{cases}$$

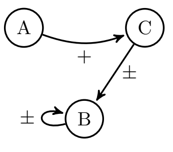
(a) Transition functions of $\mathcal{B}_1$

$$\mathcal{B}_2 = \begin{cases} f_\mathrm{A} := 0 \\ f_\mathrm{B} := (\mathrm{B} \land \neg\mathrm{C}) \lor (\neg\mathrm{B} \land \mathrm{C}) \\ f_\mathrm{C} := \mathrm{A} \end{cases}$$

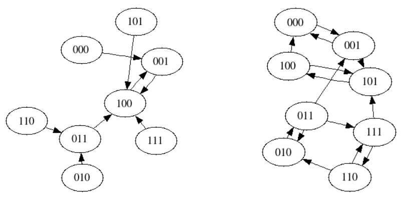
(b) Transition functions of $\mathcal{B}_2$



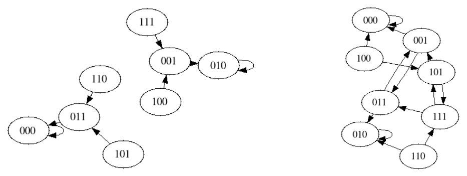
(c) Interaction graph of $\mathcal{B}_1$



(d) Interaction graph of $\mathcal{B}_2$



(e) Synchronous (left) and asynchronous (right) state transition graphs of $\mathcal{B}_1$



(f) Synchronous (left) and asynchronous (right) state transition graphs of $\mathcal{B}_2$

Fig. 2. The transition functions, derived interaction graph, and state transition graphs according to synchronous and asynchronous update schemes of two BNs.

is a vector which associates a Boolean value (1/active or 0/inactive) to each component of the BN. A BN with n components has 2^n possible configurations. For example, the $2^3 = 8$ possible configurations of a BN with 3 components are: 000, 001, 010, 011, 100, 101, 110 and 111.

Each component has an associated **transition function** ($\mathbb{B}^n \to \mathbb{B}$) which maps the configurations of the BN to the next value of the component. The transition functions are usually written as Boolean expressions. In this paper, these expressions are in Disjunctive Normal Form (DNF), *i.e.*, disjunctions of conjunctions. The conjunctions are *satisfiable*, which means they do not contain a literal and its contrary. The operators $\neg$, $\land$, $\lor$ represent respectively negation, conjunction and disjunction. Figs. 2a and 2b show examples of transition functions. The transition function associated to B in $\mathcal{B}_2$ states that the value of B will be 1 if either the value of B or of C was 1 in the previous configuration.

Like for the PKN, the structure of a BN is defined in terms of parent-child relationships between the components. A component P which appears in the transition function of a component X is called a **parent** of X. If the parent is negated in the DNF, we say that the *polarity* of the influence of P on X is negative. Conversely, if the parent is not negated, the polarity is positive. The **Interaction Graph (IG)** summarises these relationships as a directed graph. The directed edge P $\to$ X is labelled with "+" or "$-$" depending on the polarity of the influence P has on X. For example, the IG of $\mathcal{B}_1$ contains B $\xrightarrow{+}$ B and C $\xrightarrow{-}$ B because B appears positively and C appears negatively in the transition function associated to B. As we will see in Section 2.3, the PKN will act as a hard constraint on the IG of the BNs we want to synthesise.

The BN **dynamics** is obtained by applying iteratively the transition functions starting from each possible configuration. The order of application of the transition functions is defined by the **update scheme**. The **synchronous**, **asynchronous** and **mixed** update schemes are the most commonly used. In the synchronous update scheme, the transition functions are applied all at once, while in the asynchronous scheme, they are applied one by one. In the mixed update scheme, any number of components can be updated at each step. Thus, the update possibilities from both the synchronous and asynchronous update schemes are included in this third update scheme.

The **State Transition Graph (STG)** is a directed graph whose nodes are the 2^n possible configurations of the BN. In this graph, there is a directed edge from c to c' if c' is the result of applying to c the transition function(s) according to the chosen update scheme. Fig. 2 shows examples of synchronous and asynchronous STGs. As we will see later, the synthesis of BNs constrained in their dynamics implies to enforce their STG to contain specific edges, corresponding to specific transitions of configuration. We will also see how we use the mixed STG to quantify how the synthesised BNs match the dynamical constraints.

2.3 Synthesis of BNs from PKN and multivariate TS

In general, BNs that model biological systems have to satisfy two categories of *constraints*. On one hand, the BNs have to comply with a PKN. The PKN constrains the structure of the synthesised BNs by defining which components can appear as variables in each transition function and the polarity of those variables. Hence, a component P is allowed to appear in the transition function of a component X with a polarity s if the PKN contains an edge P $\xrightarrow{s}$ X. Formally, a BN is compatible with a PKN if its IG is a *spanning subgraph* of the PKN. In other words, the IG of a BN compatible with a given PKN is formed from the vertices and a subset of the edges of the PKN. For example, the two BNs presented in Figs. 2a and 2b are compatible with the PKN given in Fig. 1. On the contrary, a BN containing the transition function $f_\mathrm{A} := \mathrm{B}$ is not, since the IG of this BN contains the edge B $\xrightarrow{+}$ A, which is not in the PKN. A BN having $f_\mathrm{A} := \neg\mathrm{C}$ is also incompatible: despite C being a possible parent of A, the negative polarity is not allowed, since the PKN does not contain the edge C $\xrightarrow{-}$ A.

On the other hand, the synthesised BNs are expected to reproduce as well as possible the sequence of configurations extracted from an observed continuous multivariate Time Series (TS) of the concentration of the components over time. An example of multivariate TS is given in Table 1. Various strategies for extracting the sequence of configurations and fitting the transition functions to the observations were proposed in the literature.

We focus here on the *automatic synthesis* of BNs that respect the structure of a given PKN and are designed to reproduce as well as possible the observations from one given multivariate TS. For each synthesised BN, this ability of reproducing the observations is measured in terms of *coverage* proportion, *i.e.*, the proportion of transitions observed in the multivariate TS that are retrieved by the BN when computing its STG according to the mixed update scheme. Ideally, an identification method would only return BNs with a perfect coverage proportion (*i.e.*, 1).

Table 1. Multivariate TS of the three-components system given as example. The continuous concentrations of the components have been sampled for 20 time steps. Here, all the observations range from 0 to 100. The value resulting from the binarisation with a threshold of 50 is indicated by the colour of the cells: green if the result of the binarisation is 1 and red if 0. The resulting binary vectors are the configurations. Here there are four configurations (010, 011, 100 and 001) lasting respectively 4, 3, 3 and 10 time steps. Vertical bars indicate a change of configuration.

	configurations sequence:																			
		010		→		011	→		100	→		001								
time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A	0	3	7	13	20	30	49	61	100	63	36	25	2	3	1	1	3	0	0	0
B	100	86	64	57	54	53	51	49	45	37	33	28	22	19	14	12	9	5	2	0
C	0	27	36	42	60	75	54	44	38	48	60	72	88	90	100	100	100	100	100	100

3 State-of-the-Art Methods of BN synthesis from PKN and TS

Several studies have been dedicated to the automatic synthesis of BNs from PKNs and observed multivariate TS. Here, we review three main state-of-the-art approaches: `REVEAL` [12], `Best-Fit` [9] and `caspo-TS` [16].

For each component of the system, `REVEAL` tests all the possible combinations of its parent nodes, and attempts to find the functions that explain *all* the observations of the binarised TS. For example with the multivariate TS from Table 1: `REVEAL` tries to explain 010 → 010 → 010 → 010 → 011 → 011 → 011 → 100 → ... Hence, it cannot handle *inconsistencies* — such as a configuration being associated to distinct successor configurations. Such inconsistencies are frequent when sampling concentration along time because the processes implicated do not all have the same speed. In the example (Table 1), observing both 010 → 010 and 010 → 011 is an inconsistency which causes the failure of `REVEAL`. Furthermore, `REVEAL` cannot use the influence signs from the PKN, and since it uses an already binarised TS, it is possibly biased by the chosen binarisation.

Like `REVEAL`, `Best-Fit` tests every possible combination of the parent nodes of each component. It cannot use the influence signs and works on the binarised TS as well. Unlike `REVEAL` it can manage inconsistencies from the TS since it returns the functions that explain the maximal number of time steps. In Table 1, since $010 \to 010$ is observed three times and $010 \to 011$ only once, `Best-Fit` will focus on explaining the former instead of the latter.

`caspo-TS` was designed to manage several multivariate TS, corresponding to several experiments where the system is perturbed (forced activation or inhibition of some components), and where some measurements are potentially missing. Unlike `REVEAL` and `Best-Fit`, `caspo-TS` takes the influence signs into account. But it can only generate locally monotonous BNs, *i.e.*, BNs for which a parent of a component cannot be both its activator and its inhibitor. $\mathcal{B}_2$ is an example of BN `caspo-TS` cannot generate due to non local monotony. Indeed, in f_{B}, the components B and C act both as activator and inhibitor of B. `caspo-TS` works as the following: first, it derives the set of BNs that are compatible with the given PKN and an over-approximation of the dynamics of the TS, using the so-called most-permissive semantics [4]. Because of this over-approximation, the result can contain many false positive BNs, *i.e.*,BNs optimising the cost function used under the hood of `caspo-TS`, while their asynchronous dynamics is not able to reproduce the configurations sequence of the multivariate TS. These false positive BNs are subsequently ruled out using exact model checking. This filtering is PSPACE-hard, but thanks to the first step, a large set of BNs has already been excluded.

4 Our Approach: `ASKEeD-BN`

4.1 Details of the Approach

We propose an approach for the Automatic Synthesis of Boolean Networks from Knowledge and Data (`ASKEeD-BN`). It computes a non-redundant set of BNs complying with a given PKN and one observed multivariate TS. Unlike `REVEAL` and `Best-Fit`, `ASKEeD-BN` is capable of using the influence signs provided in the given Prior Knowledge Network (PKN) and the raw values of the input multivariate Time-Series (TS). Unlike `caspo-TS`, `ASKEeD-BN` directly fits the behaviour of each component with the TS. Also, it is not limited to the synthesis of locally-monotonous BNs.

For each component of the studied system, our approach searches among all possible transition functions. All the transition functions that do not respect the given PKN are ruled out. Then, every remaining candidate is evaluated on the basis of both their simplicity and their ability to reproduce the given observations. The candidate transition functions for the component X might not be able to explain all the binary state transitions happening at time $t \to t'$. The set of unexplained t' is denoted $\mathcal{U}$. Every time step t' in $\mathcal{U}$ is associated with a measure stating "how far" the continuous value X'_t is from the binarisation threshold θ: $|\theta - \mathrm{X}'_t|$. These spotted errors are then averaged on the T time steps of the TS through the Mean Absolute Error (MAE):

$$\mathrm{MAE}_{\mathrm{X}} = \frac{\sum_{t' \in \mathcal{U}} |\theta - \mathrm{X}_{t'}|}{T}$$

Among the candidates having the smallest MAE, we select the ones that has the smallest number of influences. Finally, we create all the possible BNs by generating all the combinations of the selected functions.

We implemented our approach using Python and the Answer-Set Programming framework (ASP) with the system *clingo* [6]. ASP is a declarative programming language oriented towards difficult (NP-hard) search problems. The possible solutions of a problem are described with the constraints they must fulfill. These constraints are written as a logic program. The ASP solver is tasked with finding the solutions of the program. To do so, it uses a Conflict-Driven Clause Learning (CDCL) algorithm inspired by SAT solvers. In our case, the CDCL algorithm avoids the evaluation of all the possible transition functions by learning from conflicts: whenever it finds that a candidate is in conflict with the constraints, it creates a new constraint that explains the conflict. These learned constraints subsequently eliminate other conflicting candidates, pruning the search space. Thanks to these pruning heuristics, our approach is efficient. ASP and in particular *clingo*, have already been used in similar contexts including `caspo-TS`.

4.2 Illustration on the Toy Example

Let us illustrate our approach on the toy example consisting of the PKN in Fig. 1 and the multivariate TS in Table 1.

When no PKN is available, the default PKN is a complete graph assuming that each component can inhibit / activate all the others (including itself). In this setup, a component with n parents have 2^{2^n} possible transition functions. In the toy example, each component can be explained by $2^{2^3} = 256$ distinct functions, which correspond to 16 777 216 potential BNs (formed by all the possible combinations of all the candidates of each component). Thanks to the available PKN, the number of candidate functions for each components A, B and C falls respectively to 3, 16 and 6. Besides the CDCL pruning, `ASKEeD-BN` virtually evaluates all the candidates, but for illustration purpose we will focus on the two that are present in $\mathcal{B}_1$ and $\mathcal{B}_2$ (Figs. 2a and 2b).

For the component A, the candidate $f_{\text{A}} := 0$ does not contain any literal and it cannot explain the transition of configuration for A at $t_7 \rightarrow t_8$. Hence, the set $\mathcal{U}$ of unexplained time steps is $\{t_8\}$. The concentration of A at time t_8 is 61, and the candidate's MAE is thus $|50-61|/20 = 0.55$. The candidate $f_{\text{A}} := \text{C}$ involves one literal (which is C). This candidate can explain all transitions. Hence, $\mathcal{U} = \emptyset$ and the MAE associated with this candidate is 0. Despite requiring more literals, $f_{\text{A}} := \text{C}$ is a better candidate than $f_{\text{A}} := 0$ because its MAE is smaller. The comparisons of the candidates proposed for the components B and C in $\mathcal{B}_1$ and $\mathcal{B}_2$ are summarised in Table 2.

For the toy example, our approach returns $\mathcal{B}_1$ as the only solution. It retrieves the 3 configuration transitions extracted from the binarised TS, thus its coverage proportion is 1. `REVEAL` does not find any BN, and the BN returned by `Best-Fit` does not comply with the PKN. `caspo-TS` finds 5 BNs with coverage proportions ranging from 0.33 to 1 (standard deviation of 0.25).

Table 2. Number of influences and MAE for the candidate functions in $\mathcal{B}_1$ (Fig. 2a) and $\mathcal{B}_2$ (Fig. 2b). A checkmark indicates the candidate selected by our approach, and the best for each criterion: (1) minimal MAE and (2) minimal number of influences.

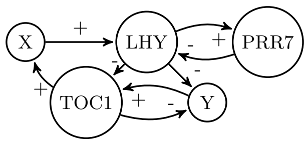
candidate	$f_B := B \land \neg C$ ✓	$f_B := (B \land \neg C) \lor (\neg B \land C)$	$f_C := \neg C$ ✓	$f_C := A$
MAE ($\mathcal{U}$)	0 ($\emptyset$) ✓	0 ($\emptyset$) ✓	0 ($\emptyset$) ✓	0.5 ($\{t_5\}$)
number of influences	2 ✓	4	1 ✓	1 ✓

5 Datasets and Procedure for the Comparative Evaluation

5.1 Datasets

In order to compare our approach with `REVEAL`, `Best-Fit` and `caspo-TS`, we used eight biological systems. For two of these systems (*yeast*'s cell cycle and *A. thaliana*'s circadian clock), their PKN and experimental multivariate TS are taken from [13] and [18] respectively. These two systems are summarised in Table 3. They respectively involve 4 and 5 components.

Table 3. Summary of two biological systems and their corresponding datasets

System	Genes	PKN	TS	Source
yeast (cell cycle)	Fkh2, Swi5, Sic1 & Clb1	Sic1 does not influence itself nor Fkh2	14 time steps 6 transitions	[18]
A. thaliana (circadian clock)	LHY, PRR7, TOC1, X & Y	 X LHY PRR7 TOC1 Y + - + - - + + -	50 time steps 11 transitions	[13]

For the six other systems[3], we conducted our experiments on multivariate TS that we simulated from existing BNs taken from the repository of example BNs of the package `PyBoolNet` [8]. For these systems, the number of components ranges from 3 to 10. For each system, the used PKN is the IG of the associated BN. As for the generation of the multivariate TS, three parameters are taken into consideration: the update scheme (in $\{\text{synchronous}, \text{asynchronous}\}$), the maximum number of introduced repetitions of each configuration (in $\{1, 4\}$) and the standard deviation of the added noise (in $\{0, 0.1\}$). For each setting of these parameters, we follow a procedure similar to what is implemented in the `generateTimeSeries` function of the R package `BoolNet` [15]:

1. choose randomly a configuration of the considered BN,

[3] `raf`, `randomnet_n7k3`, `xiao_wnt5a`, `arellano_rootstem`, `davidich_yeast` and `faure_cellcycle`

2. on this configuration, apply the update function(s) 20 times w.r.t the chosen update scheme,
3. duplicate randomly each configuration in the obtained sequence (added in contrast to `generateTimeSeries`),
4. add a Gaussian noise with a standard deviation of N.

For a given setting of the 3 parameters and a given system, we run the procedure 7 times (with different random seed). In the following, we denote *ARN* the setting with the Asynchronous update scheme, Repetitions (of 4) and Noise (of 0.1). We believe that this setting allows us to obtain multivariate TS which are quite close to real TS.

We illustrate here how to generate a synthetic multivariate TS in the ARN setting for $\mathcal{B}_1$ (Fig. 2a). We would start from a random configuration. Let it be 010. Then we apply 20 times the transition functions of $\mathcal{B}_1$ with the asynchronous update scheme. This process is not deterministic as any path from Fig. 2e (right) starting from 010 and of length 20 is valid. Let's say we obtain a path starting with $010 \rightarrow 011 \rightarrow 010 \rightarrow 011 \rightarrow 111 \rightarrow 101 \rightarrow \ldots$ Then we add a random number of duplications (in bold). The beginning of the sequence could for example look like $010 \rightarrow 011$ **→ 011→** $010 \rightarrow 011$ **→ 011→ 011→** $111 \rightarrow 101$ **→ 101→ 101→ 101→** … Finally, we add a random Gaussian noise with a standard deviation of 0.1. The synthetic multivariate TS could now start with $(0.02; 0.92; -0.16) \rightarrow (0.04; 0.8; 0.7) \rightarrow (-0.05; 1.06; 0.7) \rightarrow \ldots$

5.2 Details on the Evaluation Procedure

For `REVEAL` and `Best-Fit` we use the implementation from the R package `BoolNet` [15]. `caspo-TS` is ran with the option `mincard`, that asks for BNs with functions minimising the number of influences. Note that this is also what our method optimises.

In the following, we define an *experiment* as a BN identification method applied on a system with *one* multivariate TS. The unicity of the multivariate TS makes the problem under-specified and allows us to evaluate the performances of the different approaches in this context.

`REVEAL`, `Best-Fit` and our approach need the binarised multivariate TS in their inputs. We use a simple form of binarisation: the binarisation threshold is defined as $min + (\max - \min)/2$. All values from the multivariate TS greater or equal to the threshold are binarised to 1, and to 0 otherwise. For the two systems with real TS, the theoretic range of the values is not know in advance, so the binarisation threshold is determined component-wise: the components are binarised taking into account their observed minimum and maximum. For the six systems with the synthetic multivariate TS, we know *a priori* that the values of all the components are between 0 and 1 (± the noise). In case of noisy data, the fluctuations of a constant component are interpreted as state change when using a threshold computed component-wise. However, the identification methods are not capable to detect these spurious transition in the binarized TS.

Hence, we compute the binarisation threshold globally, on all the observation of all the components.

In order to have a fair comparison of the methods, and since `caspo-TS` is making the binarisation itself and is not aware that the theoretical minimum and maximum of the components are 0 and 1 ($\pm$ the noise), we correct *a posteriori* the transition functions it returned. The value of the constant is set to the binarised value that is the most present in the binarised TS of the component concerned. Also, since `caspo-TS` does not return a function for the components without parents in the PKN nor for the components that it founds constant for all the TS (in the case where no noise is involved), we use the same technique to set the transition functions to their correct values. We also added a step to filter out BNs returned by `REVEAL` and `Best-Fit` which do not respect the polarities given in the PKN.

For all the BNs returned by the four methods (and after the PKN-based filtering for `REVEAL` and `Best-Fit`), we use `PyBoolNet` [8] to compute the STG of each retrieved BN according to the mixed update scheme. Finally, we evaluate the results of each experience according to three criteria:

- the number of BNs returned;
- the median of the coverage ratios: the proportion of configuration transitions extracted from the input TS that are present in the mixed STG;
- the standard deviation of the coverage ratios.

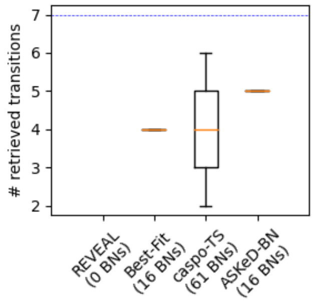
All data and programs needed to reproduce the presented results are accessible at `https://gitlab.inria.fr/avaginay/OLA2021`.

6 Results

6.1 Results on Systems with Real PKN and Experimental Multivariate TS

Yeast (Fig. 3 left). For this system `caspo-TS` find 61 BNs while `Best-Fit` and `ASKEeD-BN` both find 16 BNs. As for `REVEAL`, due to inconsistencies in the TS, it does not return any BN. Concerning the coverage, on the 7 transitions observed in the TS, the BNs synthesised by `Best-Fit` recover 4 and the BNs synthesised by `ASKEeD-BN` recover five. The best coverage ratio (6 retrieved transitions over 7) is obtained for 8 BNs synthesised by `caspo-TS` (among the total of 61). Nevertheless, as the box plot shows, the BNs synthesised by `caspo-TS` present a large variance in their coverage.

A. thaliana (Fig. 3 right). For this system, `REVEAL` returns no BN. The only BN returned by `Best-Fit` has all the components set to 1 and recovers 5 transitions over the 10 observed. `ASKEeD-BN` also returns a single BN with a perfect coverage since the BN recovers all the 10 transitions. As for the 5 BNs synthesised by `caspo-TS`, we can make the same observation as before: they present a variability in their coverage. The best coverage obtained by `caspo-TS` are from 2 different BNs including the one synthesised by `ASKEeD-BN`.



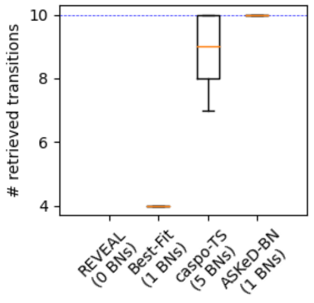


Fig. 3. Number of transitions retrieved by the BNs synthesised using the different methods on the systems *yeast* (left) and *A. thaliana* (right). The blue dashed line indicates the number of transitions that were observed in the multivariate TS.

To sum up, the results on these two real examples show that:

– `REVEAL` constantly fails to return any BN. At the opposite, `caspo-TS` returns more BNs than the other methods;
– the coverage of the BNs returned by both our approach and `caspo-TS` are better than for `Best-Fit`;
– `caspo-TS` presents worse variability in the coverage ratio of its BNs compared to our approach.

6.2 Results on Systems with generated multivariate TS

Number of Synthesised BNs: The total number of BNs returned on the synthetic datasets and the number of times the identification methods failed returning any BNs are reported in Table 4. The table shows that a large proportion of BNs generated by `REVEAL` and `Best-Fit` were not complying with the influence signs from the input PKN. The following reported results do not take into account these non-compliant BNs. `REVEAL` is the method which returns the smallest number of BNs, in particular in the ARN setting. This is due to the inconsistencies in the TS, which are frequent in the ARN setting (as in real TS). On the opposite, `caspo-TS` is the method that returned the largest number of BNs. Moreover, when considering all experiments, there are 18 experiments for which `caspo-TS` generated more than 100 BNs. In these cases, we stopped the enumeration and analysed the 100 first BNs `caspo-TS` returned. Despite this limit, `caspo-TS` returned between 5 and 7 times more BNs than our method.

From here on, we focus on the results of the experiments corresponding to the ARN setting (Asynchronous update scheme, random Repetition of configurations, and Noise addition) after having remove the BNs from `REVEAL` and `Best-Fit` which does not respect the given PKN.

Coverage ratio: To assess the coverage ratio criterion, instead of plotting the boxplots for the 42 experiments of this setting (6 systems times 7 replicates), we

Table 4. Number of experiments for which each method failed to return any BN, number of BNs returned over all 336 experiments with synthetic TS and number of BNs returned over the 42 experiments with the ARN setting. The "before / after filter" refers to the step which rules out the BNs not respecting the signs of the given PKN (see Section 5.2).

		`REVEAL`		`Best-Fit`		`caspo-TS`	`ASKEeD-BN`
	setting	before filter	after filter	before filter	after filter		
# failing experiments	all	230	240	0	64	20	0
# total returned BNs	all	100 677 500	406	100 678 198	724	8481	1210
# total returned BNs	ARN	3	3	51	35	720	85

summarised them in Fig. 4. In the scatter plot, each experiment is represented by a point whose coordinates are the coverage ratio median of the synthesised BNs and the associated standard deviation (std). The more top-right a point is, the better the corresponding identification method is (*i.e.*, it produces BNs with high coverage ratio and low std). We can see that for the few experiments for which `REVEAL` was able to return BNs, the median coverage is actually excellent. The median coverage of the BNs returned by `Best-Fit` is almost uniform: `Best-Fit` lacks regularity in finding BNs with good coverage. But the high pick around 0 on the plot of std distribution shows that for a given experiment, the BNs returned by `Best-Fit` have similar coverage rates. `caspo-TS` and our approach have a very similar distribution of median coverage. They are both good at finding BNs with very good coverage. But here again, for a given experiment, the BNs synthesised by `caspo-TS` present a bigger variation of their coverage proportions than the ones synthesised by our approach.

7 Conclusion and Perspectives

We presented `ASKEeD-BN`, a novel method to create BNs from a PKN and a multivariate TS. The results on 8 biological systems showed that our approach has the best trade-off on the evaluation criteria: it returns a small set of BNs with a high coverage median and low variance. Our results actually confirm that although `caspo-TS` finds good BNs, too many sub-optimal BNs are also retrieved. Indeed a new version of `caspo-TS` was recently proposed to tackle this problem [5].

We now present two perspectives to improve our approach and the study. First of all, real datasets may contain outlier measurements which could mislead the computation procedure of the binarisation thresholds we used in this paper. It would be interesting to see how such cases impact the performances of the identification methods and to propose a better binarisation procedure with prior outliers detection for instance. Second, contrarily to `REVEAL`, `Best-Fit` and `caspo-TS`, our approach does not handle multiple multivariate TS. However, biologists often have several multivariate TS generated with perturbations forcing some components to stay either active or inactive. However, exploiting such

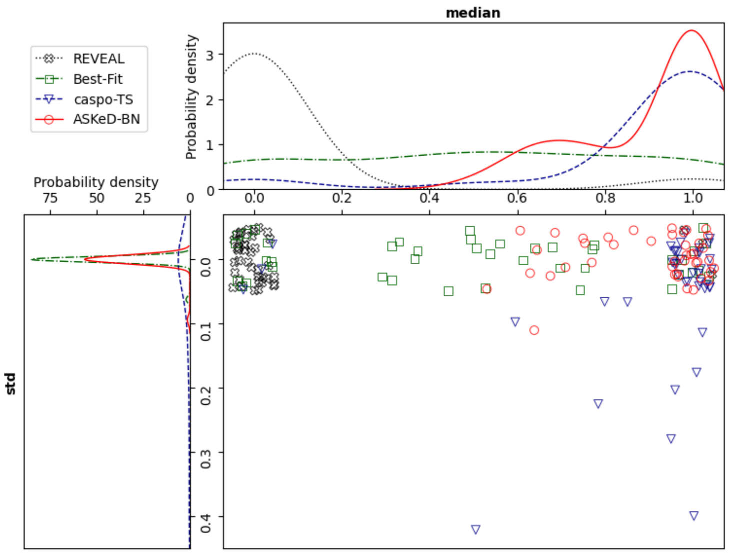


Fig. 4. On the scatter plot, each point represents an experiment in the ARN condition for which a given method potentially returned several BNs with different coverage ratios. The horizontal coordinate of the point is the median of these ratios. The vertical coordinate is their standard deviation (std). For a better visualisation, the coordinates have been jittered with a variance of 0.1 on both axes. The curves on the top (resp. on the left) of the scatter plot are the probability densities of the median (resp. the std) of the points in the scatter plots. The densities have been estimated from the non-jittered coordinates of the points with the Gaussian kernel density estimation method. The smoothing parameter of the estimator was determined automatically (with the Scott method). The areas under all these curves are 1, and the picks show where the points are the most concentrated.

supplementary data gives more information about the behaviour of the studied system in specific conditions (*e.g.*, pathological states). This knowledge allows to constrain even more the space of solutions.

Finally, we are currently working on an automatic pipeline for BN synthesis from a curated mathematical model repository, namely BioModels [14]. This requires (i) automatic extraction of the PKN from the model structure encoded in the SBML[4] file format and (ii) generation of a multivariate TS by simulation of these models.

Acknowledgements We thank Julie Lao and Hans-Jörg Schurr for their valuable comments and suggestions.

[4] Systems Biology Markup Language

References

[1] R. Albert and J. Thakar. "Boolean Modeling: A Logic-Based Dynamic Approach for Understanding Signaling and Regulatory Networks and for Making Useful Predictions". In: *Wiley Interdisciplinary Reviews. Systems Biology and Medicine* 6.5 (2014), pp. 353–369. DOI: 10.1002/wsbm.1273.

[2] C. Biane, F. Delaplace, and T. Melliti. "Abductive Network Action Inference for Targeted Therapy Discovery". In: *Electronic Notes in Theoretical Computer Science* 335 (2018), pp. 3–25. DOI: 10.1016/j.entcs.2018.03.006.

[3] S. Bornholdt. "Less Is More in Modeling Large Genetic Networks". In: *Science* 310.5747 (2005), pp. 449–451. DOI: 10.1126/science.1119959.

[4] T. Chatain, S. Haar, J. Kolčák, and L. Paulevé. *Most Permissive Semantics of Boolean Networks.* Research Report. 2020. URL: https://hal.archives-ouvertes.fr/hal-01864693.

[5] S. Chevalier, V. Noël, L. Calzone, A. Zinovyev, and L. Paulevé. "Synthesis and Simulation of Ensembles of Boolean Networks for Cell Fate Decision". In: *Computational Methods in Systems Biology.* Ed. by A. Abate, T. Petrov, and V. Wolf. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 193–209. DOI: 10.1007/978-3-030-60327-4_11.

[6] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub. *Answer Set Solving in Practice.* Morgan & Claypool Publishers, 2012. ISBN: 978-1-60845-971-1.

[7] S. A. Kauffman. "Metabolic Stability and Epigenesis in Randomly Constructed Genetic Nets". In: *Journal of Theoretical Biology* 22.3 (1969), pp. 437–467. DOI: 10.1016/0022-5193(69)90015-0.

[8] H. Klarner, A. Streck, and H. Siebert. "PyBoolNet: A Python Package for the Generation, Analysis and Visualization of Boolean Networks". In: *Bioinformatics* (2016). DOI: 10.1093/bioinformatics/btw682.

[9] H. Lähdesmäki, I. Shmulevich, and O. Yli-Harja. "On Learning Gene Regulatory Networks under the Boolean Network Model". In: *Machine Learning* 52.1 (2003), pp. 147–167. DOI: 10.1023/A:1023905711304.

[10] Y. Lazebnik. "Can a Biologist Fix a Radio?—Or, What I Learned While Studying Apoptosis". In: *Cancer Cell* 2.3 (2002), pp. 179–182. DOI: 10.1016/S1535-6108(02)00133-2.

[11] N. Le Novère. "Quantitative and Logic Modelling of Molecular and Gene Networks". In: *Nature Reviews Genetics* 16.3 (2015), pp. 146–158. DOI: 10.1038/nrg3885.

[12] S. Liang, S. Fuhrman, and R. Somogyi. "REVEAL, a General Reverse Engineering Algorithm for Inference of Genetic Network Architectures". In: *Pacific Symposium on Biocomputing.* (1998), pp. 18–29. ISSN: 2335-6928.

[13] J. C. W. Locke et al. "Experimental Validation of a Predicted Feedback Loop in the Multi-Oscillator Clock of Arabidopsis Thaliana". In: *Molecular Systems Biology* 2.1 (2006), p. 59. DOI: 10.1038/msb4100102.

[14] R. S. Malik-Sheriff et al. “BioModels—15 Years of Sharing Computational Models in Life Science”. In: *Nucleic Acids Research* 48.D1 (2020), pp. D407–D415. DOI: 10.1093/nar/gkz1055.

[15] C. Müssel, M. Hopfensitz, and H. A. Kestler. “BoolNet—an R Package for Generation, Reconstruction and Analysis of Boolean Networks”. In: *Bioinformatics* 26.10 (2010), pp. 1378–1380. DOI: 10.1093/bioinformatics/btq124.

[16] M. Ostrowski, L. Paulevé, T. Schaub, A. Siegel, and C. Guziolowski. “Boolean Network Identification from Perturbation Time Series Data Combining Dynamics Abstraction and Logic Programming”. In: *Biosystems* 149 (2016), pp. 139–153. DOI: 10.1016/j.biosystems.2016.07.009.

[17] J. D. Schwab, S. D. Kühlwein, N. Ikonomi, M. Kühl, and H. A. Kestler. “Concepts in Boolean Network Modeling: What Do They All Mean?” In: *Computational and Structural Biotechnology Journal* 18 (2020), pp. 571–582. DOI: 10.1016/j.csbj.2020.03.001.

[18] P. T. Spellman et al. “Comprehensive Identification of Cell Cycle-Regulated Genes of the Yeast Saccharomyces Cerevisiae by Microarray Hybridization”. In: *Molecular Biology of the Cell* 9.12 (1998), pp. 3273–3297. DOI: 10.1091/mbc.9.12.3273.