EURASIP Journal on Bioinformatics and Systems Biology
a SpringerOpen Journal

RESEARCH Open Access

Detecting controlling nodes of boolean regulatory networks

Steffen Schober[1*], David Kracht[1], Reinhard Heckel[1,2] and Martin Bossert[1]

Abstract

Boolean models of regulatory networks are assumed to be tolerant to perturbations. That qualitatively implies that each function can only depend on a few nodes. Biologically motivated constraints further show that functions found in Boolean regulatory networks belong to certain classes of functions, for example, the unate functions. It turns out that these classes have specific properties in the Fourier domain. That motivates us to study the problem of detecting controlling nodes in classes of Boolean networks using spectral techniques. We consider networks with unbalanced functions and functions of an average sensitivity less than $\frac{2}{3}k$, where k is the number of controlling variables for a function. Further, we consider the class of 1-low networks which include unate networks, linear threshold networks, and networks with nested canalyzing functions. We show that the application of spectral learning algorithms leads to both better time and sample complexity for the detection of controlling nodes compared with algorithms based on exhaustive search. For a particular algorithm, we state analytical upper bounds on the number of samples needed to find the controlling nodes of the Boolean functions. Further, improved algorithms for detecting controlling nodes in large-scale unate networks are given and numerically studied.

1 Introduction

The reconstruction of genetic regulatory networks using (possibly noisy) expression data is a contemporary problem in systems biology. Modern measurement methods, for example, the so-called *microarrays*, allow measuring the expression levels of thousands of genes under particular conditions. A major problem is to predict the structure of the underlying regulatory network. The overall goal is to understand the processes in cells, for example, how cells execute and control operations required for the functions performed by the cell. In the Boolean model, this implies that based on a given set of observed state-transition pairs (samples), the Boolean functions attached to each node need to be identified. In general, this problem is quite hard, due to the large number of possible Boolean functions. First results for the noiseless case appeared 1998 in the work of Liang et al. [1]. Their *Reverse Engineering Algorithm* (REVEAL) tries in a first step to find the *controlling nodes* of each node by estimating the mutual information between possible variables and the regulatory function's output. After the inputs have been identified, the *truth table* of the Boolean functions can be determined from the samples. If the number of variables for each function is at maximum K, the REVEAL algorithm considers any of the $\binom{n}{K}$ combinations of variables, where n is the number of nodes in the network.

The numerical results in [1] suggest that it is possible to identify a Boolean network using a small number of samples. Akutsu et al. [2] gave an analytical and constructive proof that it is possible to identify the network using only $\mathcal{O}(\log n)$ samples with high probability. For constant values of K, the given algorithm, BOOL, has *time complexity* $\mathcal{O}(n^{K+1} \cdot m)$ where m is the number of samples. Later it was shown that a similar algorithm also works in the presence of (low-levela) noise [3]. These algorithms are based on exhaustive search in two ways. First, they search through all $\binom{n}{K}$ possible combinations of controlling nodes. Second, they search through all of the 2^{2^K} possible Boolean functions. Lähdesmäki et al. [4] overcame the problem to search through all possible Boolean functions, reducing the double exponential factor to roughly 2^K. But their algorithm still searches through all $\binom{n}{K}$ possible variable combinations, hence, runs roughly in time n^K.

* Correspondence: steffen.schober@uni-ulm.de
[1]Institute of Telecommunications and Applied Information Theory, Ulm University, Ulm, Germany
Full list of author information is available at the end of the article

If n is large, applying such an algorithm is prohibitive even for moderate values of K.

The algorithms above implicitly solve two distinct problems. First, the controlling nodes of all nodes have to be detected, and second, each function has to be determined. This paper is dedicated to algorithms for detecting controlling nodes in Boolean networks. In general, this problem can be solved by exhaustive search in time n^K. By exploiting structural properties of certain classes of functions, the time and *sample complexity* of the algorithms can be reduced. The sample complexity of an algorithm is the number of samples needed to detect the controlling nodes with a predefined probability. In fact, one can readily apply methods stemming from the area of PAC (probably approximately correct) learning theory [5], as the network identification problem can be reduced to the problem of learning Boolean *juntas*, i.e., Boolean functions that depend[b] only on a small number of their arguments. This problem was studied by Arpe and Reischuk [6] extending earlier work of Mossel et al. [7,8].

The particular inference problem studied here is the following. Given a synchronous Boolean network and a set of input/output patterns, i.e.,

$$\{(\mathbf{X}'_1, \mathbf{Y}'_1), (\mathbf{X}'_2, \mathbf{Y}'_2), \ldots, (\mathbf{X}'_m, \mathbf{Y}'_m)\},$$

where $\mathbf{X}'_l$ and $\mathbf{Y}'_l$ describe noisy observations of two successive network states $\mathbf{X}_l$ and $\mathbf{Y}_l$ at some time t_l and $t_l + 1$, respectively. The networks state $\mathbf{X}_l$ at time t_l is modeled using a uniformly distributed random variable $\mathbf{X}$.

The task to detect the controlling nodes can be reduced to the problem to find the *essential variables* of the Boolean functions. This problem is easier to solve for some classes of functions, namely for nearly all unbalanced functions and functions of an average sensitivity less then $\frac{2}{3}k$, where k is the number of controlling variables for a function. Further the class of 1-low networks, which include unate networks, linear threshold networks, and networks with nested canalyzing functions, is considered. The application of spectral learning algorithms leads to both better time and sample complexity for the detection of controlling nodes compared with exhaustive search. In particular, a slight improvement in the algorithm given in [6] is presented, for which analytical bounds on the number of samples needed to find the controlling nodes are derived. It is notable that for the class of 1-low networks, the time complexity of the resulting algorithms is roughly n^2. The algorithm is further improved, where the main focus lies on the identification of controlling nodes in a large-scale unate network.

Finally, the performance of the improved algorithms is evaluated for large-scale unate networks with 500 nodes using numerical simulations. Further, the problem is studied in a Boolean network model of a control network of the central metabolism of *Escherichia coli* with 583 nodes [9]. Preliminary results of this work were presented in [10,11].

The outline of the paper is as follows. In Section 2, Boolean networks are defined and the detection problem is formally stated. The two classes of functions considered here are introduced and discussed. Section 3 gives a brief introduction to the Fourier analysis of Boolean functions and discusses the spectral properties of the two classes of functions. Further, the algorithms are stated and analyzed in 3.3 and 3.4. Simulation results are presented in 3.5.

2 Regulatory networks and inference

2.1 Boolean regulatory networks

A Boolean network (BN) of n nodes can be described by a numbered list $F = \{f_1, f_2, ..., f_n\}$ of Boolean functions (BFs) $f_i : \{-1, +1\}^n \to \{-1, +1\}$. Each node i in the network has a binary state variable $x_i(t) \in \{-1, +1\}$ assigned, which may vary in time $t \in \mathbb{N}$. The networks state at time t is given by $\mathbf{x}(t) = (x_1, x_2, ..., x_n)(t) \in \{-1, +1\}^n$. The state of a node i at time $t + 1$ is given as

$$x_i(t + 1) = f_i(\mathbf{x}(t)),$$

i.e., given by the pre-state of the network $\mathbf{x}(t)$ and the Boolean functions f_i.

In general not all of the possible n variables of a function f_i are *essential*. The ith variable is called essential to f if and only if there exists at least one $\mathbf{x} \in \{-1, +1\}^n$ such that $f(x_1, ..., x_i, ..., x_n) \neq f(x_1, ..., -x_i, ..., x_n)$. An equivalent terminology is that the function f *depends* on the ith variable. For any function f, the set $\mathrm{var}(f) \subseteq \{1, ..., n\}$ is defined by

$$i \in \mathrm{var}(f) \quad \text{if and only if the } i\text{th variable is essential to } f;$$

hence, $\mathrm{var}(f)$ is called the *set of essential variables* of f. If $\mathrm{var}(f) \leq k$, a function f with n variables is usually called a (n, k)-junta.

Finally note that each BN can be associated with a directed graph that allows describing the network using graph theoretic terms. Let $G(V, E)$ be a directed graph, where $V = \{1, 2, ..., n\}$ is the set of nodes and $E \subseteq V \times V$ is the set of edges. The set E is defined by

$$(i, j) \in E \quad \text{if and only if } i \in \mathrm{var}(f_j).$$

2.2 The detection problem

Assume that there exists an unknown BN that is an appropriate description of an underlying dynamical

process, for example, a regulatory network. An experiment generates state-transition pairs by observing the process, but in general, the *measurements* of the state-transitions are noisy. The challenge is now to detect the functional dependencies between the nodes of the network.

This problem can be restated as follows: Assume that a function f is chosen at random from a subset of functions $\mathcal{F}$. A single state-transition contains a pre-state $\mathbf{X}_l \in \{-1, +1\}^n$, chosen according to a well defined distribution and the corresponding output of the function $Y_l = f(\mathbf{X}_l)$. Each component $X_{l,\,i}$ and Y_l is independently flipped with probability ϵ. In the following, ϵ is called the noise rate. In this way, a set of m noisy observations or samples,

$$\mathcal{X}^m = \{(\mathbf{X}'_1, Y'_1), (\mathbf{X}'_2, Y'_2), \ldots, (\mathbf{X}'_m, Y'_m)\},$$

is obtained. In the following, it is assumed that $\mathbf{X}$ is uniformly distributed. Some comments on choosing $\mathbf{X}$ uniformly distributed will be given in the last section. Given a set of samples, the task is to detect the set of essential variables of f. This should be achieved in an efficient way, since the number of nodes can be very large in realistic problems. Further, the probability of a *detection error* should be as small as possible.

2.3 Classes of regulatory functions

Different classes of functions have been proposed to model regulatory functions. The authors do not attempt to interfere in this discussion. Merely, the approach taken here is to show that many of the proposed functions fall into two classes for which Fourier-based algorithms provide an advantage in running time over algorithms based on exhaustive search. A precise definition is given later. Two classes of functions that may be reasonable models of functions in genetic regulatory networks are presented. For both of these classes, it is assumed that the number of essential variables is less or equal to k. The first class, denoted by $\mathscr{C}_{\lceil \frac{2}{3}k \rceil}$, includes

- functions with average sensitivity less than $\frac{2}{3}k$, and
- unbalanced functions,

where it is assumed that for any function f any restriction f' on $k' > 1$ of its essential variables has an average sensitivity less or equal than $\frac{2}{3}k'$ or is an unbalanced functions (or both). Note that a restriction f' is obtained from f by setting some of its variables to fixed values. The second class $\mathscr{C}_1$ includes

- unate functions, which further include
  - nested canalizing functions, and
  - linear threshold functions.

The average sensitivity of a Boolean function f is defined as

$$\mathrm{as}(f) = \sum_i \mathrm{I}_i(f),$$

where $\mathrm{I}_i(f)$ is the *influence* of the variable i on f, [12], defined as

$$\mathrm{I}_i(f) = \Pr\{f(X_1, \ldots, X_i, \ldots, X_n) \neq f(X_1, \ldots, -X_i, \ldots, X_n)\}. \quad (1)$$

Basically, low average sensitivity is a prerequisite of non-chaotic behavior in random Boolean networks (RBNs), in particular, the expectation of the average sensitivity has to be less or equal to 1 [13]. This motivates to study the class $\mathscr{C}_{\lceil \frac{2}{3}k \rceil}$ as it is widely assumed that Boolean models of biological networks are tolerant to perturbations. Unbalanced functions[c] are of interest due to a similar reason; namely, it is well known that the average sensitivity of balanced functions is lower bounded by 1 [14]. Hence, a function that has average sensitivity less than 1 is necessarily unbalanced.

Unate functions were shown to be of interest in the biological context by Grefenstette et al. [15]. These functions arise as a consequence of a biochemical model. They can be defined in terms of monotone functions. A function f is called monotone if $f(\mathbf{x}) \leq f(\mathbf{y})$ holds for every $\mathbf{x} \leq \mathbf{y}$, where $\mathbf{x} \leq \mathbf{y} \Leftrightarrow x_i \leq y_i$. A function $f(\mathbf{x}) = f(x_1, x_2, \ldots, x_n)$ is said to be unate if there exists some fixed $\sigma \in \{-1, +1\}^n$ such that $f(x_1 \cdot \sigma_1, x_2 \cdot \sigma_2, \ldots, x_n \cdot \sigma_n)$ is a monotone function. Besides the results of Grefenstette et al., the class of unate functions is considered to be very promising because each variable of a unate function is correlated with its output. This property was conjectured to be important from the first days on [1]. Secondly, it contains the class of nested canalyzing functions and linear threshold functions which can often be found in Boolean models of regulatory networks. Kauffman et al. [16] discussed nested canalizing functions in the context of RBNs and found them to have a stabilizing effect on the networks. Notably, Samal et al. [17] reported that in the large-scale Boolean model of the regulatory network of the *E. coli* metabolism [9], the input functions of 579 out of 583 genes are, at least, canalyzing. Further investigations by the authors of the present paper revealed that all functions are unate. Linear threshold functions (LTFs) often appear in Boolean models of regulatory networks, for example, [18,19]. A Boolean function is a LTF if it can be represented by

$$f(x_1, x_2, \ldots, x_n) = \begin{cases} +1 & \text{if } w_0 + \sum_{i=1}^n w_i \cdot x_i \geq 0 \\ -1 & \text{otherwise} \end{cases},$$

where $w_i \in \mathbb{R}$. For $n < 4$, the classes of unate and linear threshold functions coincide [20].

3 Learning essential variables of regulatory functions

3.1 Fourier analysis and learning

Let $f: \{-1, 1\}^n \to \{-1, 1\}$ be a n-ary BF. Any function f can be represented by its *Fourier* expansion

$$f(\mathbf{x}) = \sum_{U \subseteq [n]} \hat{f}(U) \cdot \chi_U(\mathbf{x}), \tag{2}$$

where $[n] = \{1, 2, ..., n\}$ and

$$\chi_U(\mathbf{x}) = \prod_{i \in U} x_i$$

are the *parity functions* on variables in U. The *Fourier coefficients* $\hat{f}(U)$ appearing in Equation 2 are given by

$$\hat{f}(U) = 2^{-n} \sum_{\mathbf{x} \in \{-1,+1\}^n} f(\mathbf{x}) \cdot \chi_U(\mathbf{x}). \tag{3}$$

The number of Fourier coefficients is 2^n and each takes values in the interval [-1, 1] and is a multiple of 2^{-n+1}. *Parseval's* theorem can be stated as

$$\sum_{U \subseteq [n]} \hat{f}(U)^2 = 1. \tag{4}$$

A particular property that is used later is the following. If f does not depend on the variable i, then

$$\hat{f}(U) = 0 \quad \text{if } i \in U. \tag{5}$$

Using this fact, Parseval's theorem implies that for a constant function f,

$$|\hat{f}(\emptyset)| = 1 \quad \text{and} \quad \hat{f}(U) = 0 \quad \text{for all } U \neq \emptyset.$$

Further, if f is a (n, k)-junta, all coefficients $f(U)$ with $|U| > k$ are zero, which reduces the maximal number of non-zero coefficients to 2^k. All coefficients are multiples of 2^{-k+1}, i.e., for some $c \in \mathbb{Z}$

$$\hat{f}(U) = c \cdot 2^{-k+1} \quad \text{with } |c| \leq 2^{k-1}. \tag{6}$$

Hence, for any non-zero $\hat{f}(U)$,

$$\min_{U \neq \emptyset} |\hat{f}(U)| \geq 2^{-k+1}. \tag{7}$$

Spectral learning techniques identify a function or its dependencies from randomly drawn samples by estimating the spectral coefficients. Given a set of samples $\mathcal{X}^m = \{(\mathbf{X}'_1, Y'_1), \ldots, (\mathbf{X}_m, Y_m')\}$, an estimator $\hat{h}(U)$ of the coefficient $\hat{f}(U)$ is given by

$$\hat{h}(U) = \frac{1}{m(1-2\varepsilon)^{|U|+1}} \sum_{i=1}^{m} Y'_j \cdot \chi_U(\mathbf{X}'_j). \tag{8}$$

A similar approach was first proposed in [21] for the noiseless case and can also be used in the presence of noise [22]. It can be shown that

$$\mathbb{E}\left\{\hat{h}(U)\right\} = \hat{f}(U), \tag{9}$$

see, for example, [22]. If the number of samples m grows, the estimator Equation 8 will converge to its expected value, namely $\hat{f}(U)$.

3.2 Spectral properties of specific regulatory functions

The Boolean functions mentioned in Section 2.3 be categorized according to their *lowness* [6].

Definition 1. *A Boolean function* $f: \{-1, +1\}^n \to \{-1, +1\}$ *is* τ *-low if for any* $i \in$ var*(f) there exists a set* $U \subseteq [n]$ *with* $0 < |U| \leq \tau$ *such that* $i \in U$ *and*

$$|\hat{f}(U)| > 0.$$

Clearly any function that is τ-low is also τ'-low if $\tau' > \tau$. The notation of lowness allows to define the following families of classes.

Definition 2. $\mathscr{C}_\tau$ *is the set of functions that are* τ*-low.*

In this paper, the focus is on $\lceil \frac{2}{3}k \rceil$-low and 1-low functions. First, the latter class is considered. All unate functions are 1-low. This follows as

$$|\hat{f}(\{i\})| = \mathrm{I}_i(f), \quad \text{if } f \text{ is unate}, \tag{10}$$

[23], and the fact that for any Boolean function, the influence of an essential variable is larger than zero. Hence, if the ith variable of a unate function f is essential, the Fourier coefficient $\hat{f}(\{i\})$ is non-zero.

Now the class $\mathscr{C}_{\lceil \frac{2}{3}k \rceil}$ is discussed, first the following definition is needed.

Definition 3. *A function* $f: \{-1, +1\}^n \to \{-1, +1\}$ *is mth-order correlation immune if for all* $U \subseteq [n]$ *with* $1 \leq |U| \leq m$

$$\hat{f}(U) = 0.$$

Correlation immune functions were considered by Siegenthaler [24] who used a different definition. The definition in terms of the Fourier coefficients as used here is due to Xiao and Massey [25]. These functions are of interest in cryptography, for example, to design combining functions of stream ciphers.

Unbalanced correlation immune functions cannot exist for too large m as the next theorem shows.

Theorem 1 (Mossel et al. [8]). *Let* $f: \{-1, +1\}^n \to \{-1, +1\}$ *be an unbalanced, mth order correlation immune function. Then* $m \leq \frac{2}{3} \cdot n$.

A similar proposition holds for functions with low average sensitivity.

Proposition 1. *Let* $f: \{-1, +1\}^n \to \{-1, +1\}$ *be a mth-order correlation immune function such that* $\mathrm{as}(f) \leq \frac{2}{3}n$, *where* $\mathbf{X} \in \{-1, +1\}^n$ *is uniformly distributed. Then* $m \leq \frac{2}{3} \cdot n$.

Proof. If f is unbalanced, the proposition is true. Suppose f is balanced. Assume for contradiction that

$$|\hat{f}(U)| = 0 \text{ for } 1 \leq |U| \leq m = \frac{2}{3}n. \tag{11}$$

From Parseval's theorem it follows that

$$\begin{aligned}\mathrm{as}(f) &= \sum_{U \subseteq [n]} |U| \hat{f}(U)^2 = \sum_{|U| > m} |U| \hat{f}(U)^2 \\ &> m \sum_{U \neq \emptyset} \hat{f}(U)^2 = m \cdot (1 - \hat{f}(\emptyset)^2) = \frac{2}{3}n\end{aligned}$$

which contradicts the assumption of the proposition. □

Proposition 2. *Let* f *be a function with* $k \geq 2$ *essential variables (out of n) such that any restriction* f' *on* k' *of its essential variables, where* $1 < k' \leq k$, *has an average sensitivity less or equal than* $\frac{2}{3}k'$ *or is an unbalanced functions (or both). Then* f *is* $\lceil \frac{2}{3}k \rceil$*-low.*

Proof. First note that if $k = 2$ the proposition is true. Now consider a function with $k > 2$. By assumption there is a variable $i \in \mathrm{var}(f)$ with a "low" coefficient,

1 Input: $\mathcal{X}$, n, d
2 Output: $\tilde{R}$ the essential variables
3 Global Parameters: τ, ϵ
4 begin
5 $\tilde{R} = \emptyset$;
6 foreach $U \subseteq [n]$ *and* $1 \leq |U| \leq \tau$ **do**
7 $\hat{h}(U) \leftarrow (1 - 2\varepsilon)^{-|U|-1} \cdot m^{-1} \sum_{(\mathbf{x},y) \in \chi} y \cdot \chi_U(\mathbf{x})$;
8 if $|\hat{h}(U)| \geq 2^{-d}$ **then**
9 $\tilde{R} \leftarrow \tilde{R} \cup U$;
10 end
11 end
12 end

Algorithm 1: τ-NOISY-FOURIER$_d$

that is $U \ni i$ and $|U| \leq \frac{2}{3}k$. Consider the restrictions of f to the variable i denoted with f_{-1} and f_{+1}. It is straightforward to show that

$$\hat{f}(U) = \frac{1}{2}\left(\hat{f}_{+1}(U \backslash \{i\}) + (-1)^{|\{i\} \cap U|} \hat{f}_{-1}(U \backslash \{i\})\right). \tag{12}$$

For variable $j \neq i$ there is a set $V \ni j$ and $i \notin V$ with $|V| \leq \frac{2}{3}(k-1)$ such that either $\hat{f}_{+1}(V) \neq 0$ or $\hat{f}_{-1}(V) \neq 0$ Eq. (12) implies that either $\hat{f}(V)$ or $\hat{f}(V \cup \{i\})$ not equal to zero. In the worst case one has to consider the coefficient $\hat{f}(V \cup \{i\})$. Now note that as $|V \cup \{i\}|$ is an integer number

$$|V \cup \{i\}| \leq \left\lfloor \frac{2}{3}(k-1) \right\rfloor + 1 \leq \left\lceil \frac{2}{3}k \right\rceil.$$

This argument can now be repeated recursively (applying Eq. (12) to f_{-1} and f_{+1}) showing the proposition. □

3.3 The τ-NOISY-FOURIER$_d$ algorithm

A simple algorithm to find the essential variables of τ-low (n, k)-juntas directly follows from Equations 6 and 7. First, all Fourier coefficients up to weight τ are estimated. The absolute value of each estimated coefficient $\hat{h}(U)$ is compared with a threshold. If a coefficient $\hat{f}(U)$ is non-zero, its absolute value cannot be smaller then 2^{-k+1}, see Equation 7. Hence, if $|\hat{h}(U)|$ is larger than 2^{-k}, the variables corresponding to U are classified as essential. The algorithm was given by [6], but they used 2^{-d-1} as threshold (see Line 8).

The following theorem appeared first in [6] but with a different bound.

Theorem 2. *Let* f *be a* τ*-low* (n, k)*-junta and*

$$m \geq 2 \cdot 2^{2k} \cdot (1 - 2\varepsilon)^{-2\tau-2} \ln \frac{2n^\tau}{\delta}. \tag{13}$$

Then Algorithm 1 *identifies all essential variables with probability* $1 - \delta$.

The bound is even true if ϵ is only an upper bound on the noise rate. The theorem follows from applying standard Hoeffding bounds. Note that the bound above is different to [6]. If $\tau = 1$, the number of samples required to reach a predefined probability of error is smaller by a factor 4. This directly follows from the different threshold used here. If $\tau > 1$, it was claimed in [6] that n^τ can be replaced by n. But simulation results of the authors (not shown) contradict this result; hence, we rely here on the weaker result shown in Theorem 2. This issue will be discussed in future work.

3.4 Improved algorithms

In the following section, two algorithms are discussed that lead to better numerical results as Algorithm 1 especially for low k. The first algorithm is a straight forward modification of the τ-NOISY-FOURIER algorithm and is discussed in Section 3.4.1. The second algorithm requires a further assumption on the functions to which it is applied; namely, suppose that f is τ-low. If a variable of the function f is set to a particular fixed value, i.e., -1 or +1, the *restricted* version of f is obtained (this will be discussed in more detail later on). Now it has to be

assumed that the restricted function is still τ-low, i.e., they have to be *recursive* τ-low. While it is possible to define such classes, only unate functions are considered. On the one hand, they naturally fulfill the constraint defined above, as any restriction of a unate function is again a unate function. On the other hand, they seem to be the most important class of functions as discussed earlier. Nevertheless, the following algorithms will be formulated in a way such that it is clear how to apply them for recursive τ-low functions.

3.4.1 A modification of the τ-NOISY-FOURIER$_d$

Algorithm 1 suffers from a high number of so-called type-2-errors, i.e., it classifies non-essential variables as essential, especially for a small number of samples m. Hence, a simple modification is to return only a limited number of essential variables by taking only the variables that correspond to the coefficients with largest absolute value. The algorithm is denoted by τ -NOISY-FOURIER-$^{\text{MOD}}$ and is shown below. The computational complexity of the algorithm increases compared with Algorithm 1. In line 8 $\binom{n}{\tau}$, many spectral coefficients have to be sorted which can be done in roughly $n^{2\tau}$ in the worst case [26].[d] In Figure 1 on page 19, the effect of the modification on the detection error is numerically studied.

3.4.2 The KJUNTA algorithm

The second algorithm is based on the original idea of Mossel et al. [8] who recursively applied their algorithm to restricted functions of the original. While they did for other reasons, a slight modification of their approach can be used to reduce the number of samples needed. The running time of the algorithm is increased by an exponential dependency on k.

1 Input: $\mathcal{X}$, n, d
2 Output: $\tilde{R}$ the essential variables
3 Global Parameters: τ, ϵ
4 begin
5 $\tilde{R} \leftarrow \emptyset$;
6 **foreach** $U \subseteq [n]$ *and* $|U| \leq \tau$ **do**
7
$\hat{h}(U) \leftarrow (1-2\varepsilon)^{-|U|-1} \cdot m^{-1} \cdot \sum_{(\mathbf{x},y)\in\mathcal{X}} y \cdot \chi_U(\mathbf{x})$;
8 **end**
9 $U_i : |\hat{h}(U_1)| \geq |\hat{h}(U_2)| \geq \cdots \geq |\hat{h}(U_l)|$ `// mod: sorted index;`
10 **for** $i = 1$ *to* l **do**
11 **if** $|\tilde{R}| < d$ **then** `// mod: limiting condition`
12 **if** $|\hat{h}(U_i)| \geq 2^{-d}$ **then** $\tilde{R} \leftarrow \tilde{R} \cup U_i$;
13 **end**
14 **end**
15 end

Algorithm 2: τ -NOISY-FOURIER$^{\text{MOD}}$

To describe the algorithm, some additional definitions are needed. Define a (n, d) restriction $\rho = (\rho_1, \rho_2, ..., \rho_n)$ as a vector of length n which consists of symbols in {+1, -1, *}, where the symbol * occurs exactly d times. The *restricted* function $f|_\rho$ can be obtained from the function f by fixing d arguments x_i in the following way. If $\rho_i \neq *$ then $x_i = \rho_i$. All x_i for i such that $\rho_i = *$ are the arguments of $f|_\rho$; hence, it depends on at most d arguments. A vector $\mathbf{x}$ of length n matches if for all $\rho_i \neq *$ it holds that $x_i = \rho_i$. The restricted samples set $\mathcal{X}_\rho$ is defined as a subset of $\mathcal{X}$ that contains all samples $(\mathbf{x}, y)$ such that $\mathbf{x}$ matches the restriction ρ, i.e.,

$$\mathcal{X}_\rho = \langle (\mathbf{x},\ y) \in \mathcal{X} | \mathbf{x} \text{ matches } \rho \rangle.$$

The algorithm is now described as follows. Suppose there exists a procedure IDENTIFY that can identify at least one essential variable of a function f given a number of samples. If no essential variables exist, i.e., if f is constant, the procedure returns the empty set $\emptyset$.

Given a (n, k)-junta f, with $k > 0$, and a set $I \subseteq R = \text{var}(f)$ that contains some essential variables that are already known. Further, assume that there is a restriction ρ that fixes exactly the variables in I. The function $f|_\rho$ can be either the constant function or depend on some of the variables that are not fixed yet. For the latter case suppose that at least one new variable can be identified, using procedure IDENTIFY. Denote the set of newly identified variables with I. Then the procedure is continued with all of the $2^{|I|}$ new restrictions that fix the

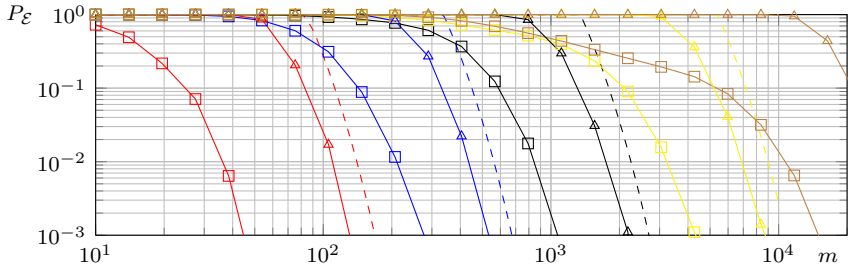

Figure 1 The average detection error in 10000 trials: Theoretical bound (dashed), original (triangle), and modified (box) τ-NOISY-FOURIER$_d$, for unate functions with $n = 500$, $\epsilon = 0.05$, $d = k = 1$ (red), 2 (blue), 3 (black), 4 (yellow), 5 (brown).

variables in I until all these sub-restrictions will be constant. The resulting algorithm in a recursive form is given as Algorithm 3. Initially, the algorithm is started with KJUNTA$(\mathcal{X}, n, d)$, where the global parameters (τ = 1, ϵ) are fixed.

Most of the algorithm has been explained already. First note that passing n as an argument is not necessary, because it is an implicit parameter of the

1 Input: $\mathcal{X}$, n, d
2 Output: $\tilde{R}$ the essential variables
3 Global Parameters: τ, ϵ
4 begin
5 $\tilde{R} \leftarrow \emptyset$;
6 $I \leftarrow$ IDENTIFY$(\mathcal{X}, d)$;
7 **if** $(d > |I| > 0)$ **then**
8 $\tilde{R}' \leftarrow \emptyset$;
9 **foreach** *restriction* ρ **do**
10 $\tilde{R}' \leftarrow \tilde{R}' \cup$ KJUNTA $(\mathcal{X}_\rho,\ n - |I|,\ d - |I|)$;
11 **end**
12 $\tilde{R} \leftarrow$ COMBINE $\left(\tilde{R}, \tilde{R}', \rho\right)$;
13 **end**
14 end

Algorithm 3: KJUNTA

1 Input: $\mathcal{X}$, n, d
2 Output: I variables found
3 Global Parameters: τ, ϵ
4 begin
5 $I \leftarrow \varnothing$;
6 **foreach** $U \subseteq [n]$ *and* $|\mathrm{U}| \leq \tau$ **do**
7 $\hat{h}(U) \leftarrow (1-2\varepsilon)^{-|U|-1} \cdot m^{-1} \cdot \sum_{(\mathbf{x},y)\in\mathcal{X}} y \cdot \chi_U(\mathbf{x})$;
8 **end**
9 $M \leftarrow \arg\max_{U:0<|U|\leq\tau} |\hat{h}(U)|$;
10 **if** (CONST $(\hat{h}(M),\ \hat{h}(\emptyset),\ d)$ = **true**)**then** $I \leftarrow M$;
11 end

Algorithm 4: IDENTIFY

samples. Further comments should be given to the line 9. The *foreach loop* is executed for each of the $2^{|I|}$ possible restrictions of the variables contained in I. For each restriction, the corresponding restricted sample set is calculated and passed in a new call to KJUNTA. Each of these calls runs on smaller problems, namely finding variables of a (n - $|I|$, d - $|I|$)-junta. Notably, each of these runs is independent of the others. The variables found are then combined with $\tilde{R}$ in line 11 using the procedure COMBINE. This is not just a union of sets since one has to take care about the labeling of the variables. For example, if $\tilde{R} = \{1\}$, and a subsequent call of KJUNTA returns variables joined to $\tilde{R}' = \{1, 3\}$, combining both leads to $\tilde{R} = \{1, 2, 4\}$.

The IDENTIFY **procedure** The question remains how to identify some of the essential variables or how to decide whether the function is constant. For τ-low functions, it is sufficient to estimate all coefficients $\hat{f}(U)$ with $|U| \leq \tau$. In [7], it was proposed to search for the first coefficient that is above a certain threshold. The approach here is different. In particular, all coefficients with weight less or equal τ are computed. The coefficient with the maximum absolute value is compared with the zero coefficient to distinguish between a constant and a non-constant function. How this can be done is discussed below. The resulting algorithm is formulated in terms of Algorithm 4 on page 12. In line 8, the procedure CONST is called which tries to distinguish between a constant function and a non-constant function. If a non-constant function is found, the variables in M are returned, otherwise the empty set.

The CONST **procedure** In the following it is discussed how a constant function can be distinguished from a non-constant function, given that the function depends on not more than k variables. This is done based on the zero coefficient $\hat{f}(\emptyset)$ and the coefficient with the largest absolute value, denoted by $\hat{f}(M)$. Note that if and only if f is constant, $|\hat{f}(\emptyset)| = 1$ and $\hat{f}(U) = 0$ for any set $U \neq \emptyset$ by Parseval's theorem. If f is non-constant, $|\hat{f}(\emptyset)| < 1$ and there exists at least one coefficient with $|\hat{f}(U)| > 0$ for some U; hence, it follows that $|\hat{f}(M)| > 0$.

To distinguish between a constant and a non-constant function different procedures exist. The most simple one was proposed by Mossel et al. which will be denoted by CONST1. There, if $|\hat{h}(\emptyset)| > 1 - 2^{-d}$ or $|\hat{h}(M)| < 2^{-d}$, the function is declared as constant.

For small d, a better procedure, that requires less samples, exists. It is denoted by CONST2. Given the 2-tuple $(\hat{h}(\emptyset), \hat{h}(M))$ compute the–in Euclidean distance– closest tuple (α, β) such that $\alpha < 1$, $\beta > 0$ are multiples of 2^{-d+1}. Hence, the function is declared as constant if

$$\mathrm{dist}\left((\hat{h}(\emptyset),\ \hat{h}(M)),\ (1,\ 0)\right) < \mathrm{dist}\left((\hat{h}(\emptyset),\ \hat{h}(M)),\ (\alpha,\ \beta)\right),$$

where dist $(\cdot,\cdot)$ denotes the Euclidean distance.

A note on the computational complexity As mentioned, Algorithm 3 has an increased complexity compared with Algorithm 1. In the worst case, the algorithm is called 2^k times, but clearly each time on a smaller problem. If it is assumed that $\hat{h}(U)$ can be computed in time $\mathcal{O}(n \cdot m)$, the algorithm runs in $\mathcal{O}(2^k \cdot n^2 \cdot m)$ for 1-low functions. Obviously for constant k, this reduces to $\mathcal{O}(n^2 \cdot m)$.

3.5 Simulation results for unate networks

To compare the performance of the different algorithms, the following procedure is used. Suppose a BF f is chosen uniformly at random from a class $\mathcal{F} \subseteq \mathcal{F}^n$ of n-ary

τ-low functions, where τ and n are known. For the functions f, a set of m noisy state-transitions $\mathcal{X}^m = \{(\mathbf{X}'_l, Y'_l)|l = 1..m\}$ is generated as described in Section 2.2. The noise rate is fixed to $\epsilon = 0.05$.

The most important indicator is the probability of a detection error. Define $\mathcal{E}$ as the event $\{\tilde{R} \neq \mathrm{var}(f)\}$ where $\tilde{R}$ is the *detected variable set*. The detection error probability

$$P_{\mathcal{E}} = \Pr\left\{\tilde{R} \neq \mathrm{var}(f)\right\}$$

is a prior indicator on the algorithm's performance. It should be mentioned that if there exists a function f such that var(f) >d, the detection error probability $P_{\mathcal{E}}$ does not vanish, even for large m.

Further evaluation criteria that are used in Section 3.5.3 are the *precision rate* ρ and the *false-negative rate* β. In the present context, the precision rate is defined as the conditional probability that a detected variable is indeed an essential variable, i.e.,

$$\rho = \Pr\left\{i \in \mathrm{var}(f)|i \in \tilde{R}\right\}.$$

An equivalent way of stating that matter is that a predicted edge e is in E, where $G(V, E)$ is the associated graph of the network. The false-negative rate is defined as the conditional probability that an essential variable is not detected as being essential,

$$\beta = \Pr\left\{i \notin \tilde{R}|i \in \mathrm{var}(f)\right\}.$$

In a network, this can be interpreted as the fraction of edges that have not been detected. The definitions above are consistent with Zhao et al. [27] who defined the type-1-error as the event that a node i is classified as a controlling node of some node j although this is not the case. Consequently the type-2-error is defined as the event $\{i \notin \tilde{R}|i \in \mathrm{var}(f)\}$.

3.5.1 τ-NOISY-FOURIER$_d$ versus $\tau - \mathrm{NOISY} - \mathrm{FOURIER}_d^{\mathrm{mod}}$

First, the modified version of the τ-NOISY-FOURIER$_d$ algorithm is compared with the original algorithm. In 10,000 independent experiments, unate functions with exactly k essential variables are randomly drawn. The parameter d is always set to k. The results are presented in Figure 2, further the upper bounds on the detection error probability (Theorem 2) are shown. As promised $\tau - \mathrm{NOISY} - \mathrm{FOURIER}_d^{\mathrm{mod}}$ outperforms the original algorithm.

3.5.2 $\tau - \mathrm{NOISY} - \mathrm{FOURIER}_d^{\mathrm{mod}}$ versus KJUNTA

Again a subset of unate functions with exactly k essential variables is used to compare the $\tau - \mathrm{NOISY} - \mathrm{FOURIER}_d^{\mathrm{mod}}$ algorithm with the KJUNTA algorithm. The parameter d is always set to k. The results are shown in Figure 2. For functions with a low number of essential variables, the procedure CONST1 outperforms the τ-NOISY-FOURIER$_d$ algorithm. But the better performance vanishes with an increasing number of variables.

3.5.3 τ-NOISY-FOURIER$_d$ versus KJUNTA on an E. coli network

In this simulation, the functions are chosen from the regulatory functions of the control network of the *E. coli* metabolism [9]. This set includes functions with a different number of essential variables. Further, also some constant functions are included and some functions occur several times. Each function f has 583 possible arguments but depends on not more than eight variables. The functions distribution on essential variables is given in Table 1 and is equivalent to the in-degree distribution of the corresponding network.[e] The results in Figure 3 are obtained by applying the algorithms to each function in the set, this experiment is performed 100 times.

Remarkable results: In the previous simulations, the parameter d is always set to k. Further only functions with *exactly* k essential variables are chosen. Here, the parameter d is usually smaller than k, which implies that not all variables can be found. Only variables with influence large or equal 2^{-d} can be detected. This is implied by Equations 10 and 7. On the other hand, even if $d < k$ for some function f, the algorithm can possibly detect some of the essential variables of f.

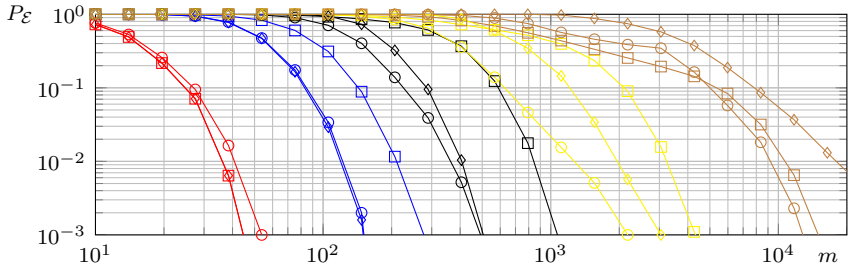

Figure 2 The average detection error in 10,000 trials: $\tau - \mathrm{NOISY} - \mathrm{FOURIER}_d^{\mathrm{mod}}$ (box) and KJUNTA with CONST1 (circle) and CONST2 (diamond) procedure, unate functions ($n = 500$, $\epsilon = 0.05$, $d = k = 1$ (red), 2 (blue), 3 (black), 4 (yellow), 5 (brown).

Table 1 In-degree distribution of the Boolean network (see text).

|var(f)|	0	1	2	3	4	5	6	7	8
#	12	293	159	66	38	9	4	0	2

4 Conclusion

In this paper, the problem to detect controlling nodes in Boolean networks is discussed. Boolean functions that are relevant for modeling genetic networks seem to belong to classes of functions for which spectral-based algorithms provide an efficient solution–both, in computational complexity and data needed. Especially the algorithms for unate functions are highly efficient in both running time and the number of samples needed to identify controlling nodes. Further analytical bounds on the probability of a detection error can be stated.

If the samples are chosen according to a uniform distribution, the results are promising. Applying the methods to the *E. coli* control network, with 583 nodes, shows that using approximately 200 samples, it is possible to find nearly 40% of all edges in the network with a precision rate close to one. On the other hand, a wrong selection of the parameter d can have a dramatic effect on the precision. For example, if under the same conditions $d = 4$ is chosen, the precision will drop below 0.5. Fortunately, the choice of the parameter can be guided by the available analytical bounds of the detection error probability. The latter is dominated by the probability that the estimator $\hat{h}(\{i\})$ will deviate from $\hat{f}(\{i\})$ by more than +/- 2^{-d}. But this also determines the precision of the algorithm. Suppose that 200 samples are obtained from the *E. coli* network. The analytical bounds shown in Figure 1 suggest to choose $d = 1$ which indeed leads to a high precision (see Figure 3).

Clearly, our assumption of uniformly distributed samples is too optimistic. Fortunately, known results from PAC learning [6] show that it is possible to use similar algorithms for product distributed samples, i.e., in a random vector $\mathbf{X}$ each X_i is chosen independently of the others with a certain probability such that $-1 < \mathbb{E}\{X_i\} = \mu_i < 1$. But there is a major problem: If $\mu_{\max} = \max_{1 \le i \le n} |\mu_i|$ gets closer to 1, the number of samples needed will increase with roughly $(1 - \mu_{\max})^{-2k}$. In unate networks, this coincides with the fact that the influences of the variables can become very small. Hence, further investigations in this direction are necessary. This would be a major step toward the application of spectral algorithms in a real-world scenario.

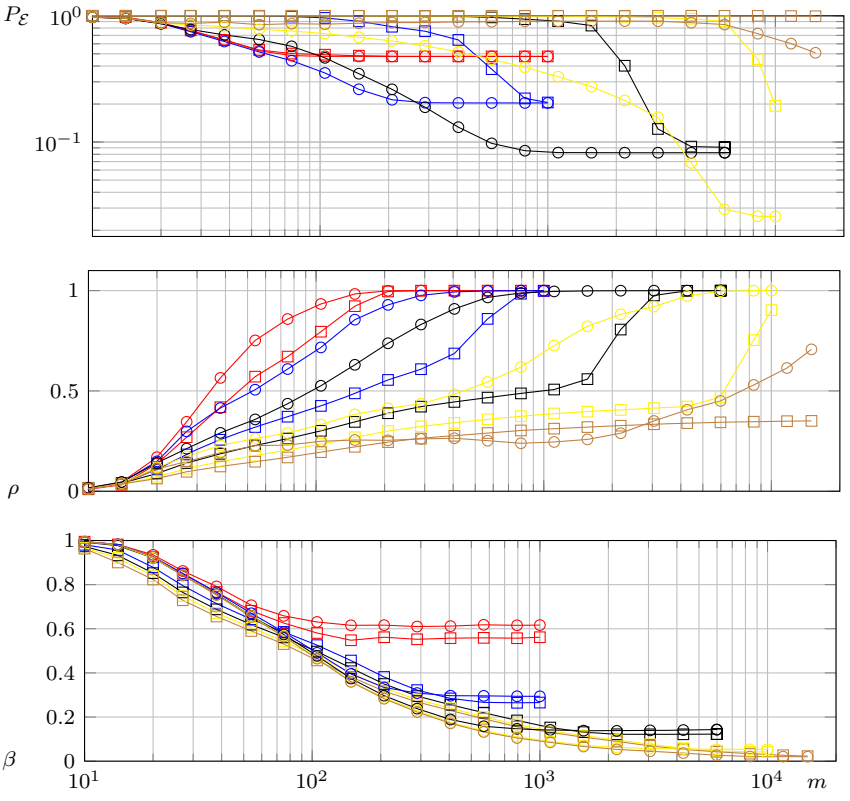

Figure 3 Simulation results for the modified τ-NOISY-FOURIER$_d^{mod}$ (box) and KJUNTA with the CONST1 (circle) procedure applied on the regulatory functions of a network of *E. coli*, see text. ($n = 583$, $\epsilon = 0.05$, $d = k = 1$ (red), 2 (blue), 3 (black), 4 (yellow), 5 (brown).

5 Competing interests

The authors declare that they have no competing interests.

Endnotes

[a]The theoretical analysis requires the noise level to be bounded below a small value. [b]This will be defined more precisely later. [c]A function is unbalanced if the number of +1 and -1 in the truth table is different. [d]Using a better implementation as Algorithm 2, this can be reduced to $2\tau \log N$. [e]The detailed table of the used functions can be found in the supplementary material.

Author details
[1]Institute of Telecommunications and Applied Information Theory, Ulm University, Ulm, Germany [2]The Communication Technology Laboratory, ETH Zürich, Switzerland

Received: 1 November 2010 Accepted: 11 October 2011
Published: 11 October 2011

References

1. S Liang, S Fuhrman, R Somogyi Reveal, A general reverse engineering algorithm for inference of genetic network architectures, in *Proceedings of the Pacific Symposium on Biocomputing*, 18–29 (1998)
2. T Akutsu, S Miyano, S Kuhara, Identification of genetic networks from a small number of gene expression patterns under the boolean network model, in *Proceedings of the Pacific Symposium on Biocomputing*, 17–28 (1999)
3. T Akutsu, S Miyano, S Kuhara, Inferring qualitative relations in genetic networks and metabolic pathways. Bioinformatics **16**(8), 727–734 (August 2000). doi:10.1093/bioinformatics/16.8.727
4. H Lähdesmäki, I Shmulevich, O Yli-Harja, On learning gene regulatory networks under the boolean network model. Mach Learn. **52**(1-2), 147–167 (2003)
5. LG Valiant, A theory of the learnable. Commun ACM. **27**(11), 1134–1142 (1984). doi:10.1145/1968.1972
6. J Arpe, R Reischuk, Learning juntas in the presence of noise. Theor Comput Sci. **384**(1), 2–21 (2007). doi:10.1016/j.tcs.2007.05.014
7. E Mossel, R O'Donnell, RP Servedio, Learning juntas, in *Proceedings of the ACM Symposium on Theory of Computing* (ACM, San Diego, CA, USA, 2003), pp. 206–212
8. E Mossel, R O'Donnell, RA Servedio, Learning functions of k relevant variables. J Comput Syst Sci. **69**(3), 421–434 (2004). doi:10.1016/j.jcss.2004.04.002
9. MW Covert, EM Knight, JL Reed, MJ Herrgard, BO Palsson, Integrating high-throughput and computational data elucidates bacterial networks. Nature **429**(6987), 92–96 (2004). doi:10.1038/nature02456
10. S Schober, K Mir, M Bossert, Reconstruction of boolean genetic regulatory networks consisting of canalyzing or low sensitivity functions, in *Proceedings of International ITG Conference on Source and Channel Coding (SCC'10)* (2010)
11. S Schober, R Heckel, D Kracht, Spectral properties of a boolean model of the *E.Coli* genetic network and their implications of network inference, in *Proceedings of International Workshop on Computational Systems Biology*, (Luxembourg, June 2010)
12. M Ben-Or, N Linial, Collective coin flipping, robust voting schemes and minima of banzhaf values, in *Proceedings of IEEE Symposium on Foundations of Computer Science*, 408–416 (1985)
13. JF Lynch, Dynamics of Random Boolean Networks, in *Current Developments in Mathematics Biology: Proceedings of Conference on Mathematical Biology and Dynamical Systems*, ed. by Culshaw R, Mahdavi K, Boucher J. (World Scientific Publishing Co, 2007), pp. 15–38
14. J Kahn, G Kalai, N Linial, The influence of variables on boolean functions, in *IEEE Proceedings of Symposium on Foundations of Computer Science*, 68–80 (1988)
15. J Grefenstette, So Kim, S Kauffman, An analysis of the class of gene regulatory functions implied by a biochemical model. Biosystems **84**(2), 81–90 (2006). doi:10.1016/j.biosystems.2005.09.009
16. SA Kauffman, C Peterson, B Samuelsson, C Troein, Genetic networks with canalyzing boolean rules are always stable. PNAS **101**(49), 17102–17107 (2004). doi:10.1073/pnas.0407783101
17. A Samal, S Jain, The regulatory network of e. coli metabolism as a boolean dynamical system exhibits both homeostasis and flexibility of response. BMC Syst Biol. **2**(1), 21 (2008). doi:10.1186/1752-0509-2-21
18. F Li, T Long, Y Lu, Q Ouyang, C Tang, The yeast cell-cycle network is robustly designed. PNAS **101**(14), 4781–4786 (2004). doi:10.1073/pnas.0305937101
19. MI Davidich, S Bornholdt, Boolean network model predicts cell cycle sequence of fission yeast. PLoS ONE **3**(2), e1672 (2008). doi:10.1371/journal.pone.0001672
20. R McNaughton, Unate truth functions. IRE Trans Electron Comput. **10**, 1–6 (1961)
21. N Linial, Y Mansour, N Nisan, Constant depth circuits, Fourier transform, and learnability. Journal ACM **40**(3), 607–620 (1993). doi:10.1145/174130.174138
22. NH Bshouty, JC Jackson, C Tamon, Uniform-distribution attribute noise learnability. Inf Comput. **187**(2), 277–290 (2003). doi:10.1016/S0890-5401(03)00135-4
23. C Gotsman, N Linial, Spectral properties of threshold functions. Combinatorica **14**(1), 35–50 (1994). doi:10.1007/BF01305949
24. T Siegenthaler, Correlation-immunity of nonlinear combining functions for cryptographic applications. IEEE Trans Inf Theory **30**(5), 776–780 (1984). doi:10.1109/TIT.1984.1056949
25. G-Z Xiao, JL Massey, A spectral characterization of Correlation-Immune combining functions. IEEE Trans Inf Theory **34**(3), 569–571 (1988). doi:10.1109/18.6037
26. DE Knuth, *Art of Computer Programming, Volume 3: Sorting and Searching*, 2nd edn. (Addison-Wesley Professional, Reading, MA, 1998)
27. W Zhao, E Serpedin, ER Dougherty, Inferring connectivity of genetic regulatory networks using information-theoretic criteria. IEEE/ACM Trans Comput Biol Bioinf. **5**(2), 262–274 (2008)

doi:10.1186/1687-4153-2011-6
Cite this article as: Schober *et al.*: **Detecting controlling nodes of boolean regulatory networks.** *EURASIP Journal on Bioinformatics and Systems Biology* 2011 **2011**:6.