Multiscale Binarization of Gene Expression Data for Reconstructing Boolean Networks

Martin Hopfensitz, Christoph Müssel, Christian Wawra, Markus Maucher, Michael Kühl, Heiko Neumann, and Hans A. Kestler

Abstract—Network inference algorithms can assist life scientists in unraveling gene-regulatory systems on a molecular level. In recent years, great attention has been drawn to the reconstruction of Boolean networks from time series. These need to be binarized, as such networks model genes as binary variables (either "expressed" or "not expressed"). Common binarization methods often cluster measurements or separate them according to statistical or information theoretic characteristics and may require many data points to determine a robust threshold. Yet, time series measurements frequently comprise only a small number of samples. To overcome this limitation, we propose a binarization that incorporates measurements at multiple resolutions. We introduce two such binarization approaches which determine thresholds based on limited numbers of samples and additionally provide a measure of threshold validity. Thus, network reconstruction and further analysis can be restricted to genes with meaningful thresholds. This reduces the complexity of network inference. The performance of our binarization algorithms was evaluated in network reconstruction experiments using artificial data as well as real-world yeast expression time series. The new approaches yield considerably improved correct network identification rates compared to other binarization techniques by effectively reducing the amount of candidate networks.

Index Terms—Binarization, gene-regulatory networks, Boolean networks, reconstruction.

1 Introduction

On a molecular level, homeostasis of adult organisms and organs as well as embryonic development of multicellular organisms is often poorly understood. Modeling and simulation contribute to a deeper understanding of these biological systems, particularly if dynamic aspects are essential [1]. Handcrafted models, often based on series of wet-lab experiments and exhaustive literature research, have been successfully applied to the study of complex biological systems [2], [3].

The increasing use of real-time PCR or high-throughput microarray techniques facilitates time series measurements of thousands of genes in parallel. To infer models from such large-scale data, an automation of the model construction process is required. Static methods, including Bayesian networks [4] and Boolean logic [5], [6], [7], have been applied to express general relations of a system [8]. In case of time-critical events like embryonic development or cell cycle processes, however, a dynamic system description is often indispensable to obtain a comprehensive understanding of biological systems. Such dynamic processes are frequently described by systems of ordinary differential equations [9], [10]. Another possibility to describe dependencies between consecutive gene expression measurements is dynamic Bayesian networks [11], [12]. They account for the stochastic nature that is inherent to biological systems. In addition to these approaches, special attention has been drawn to Boolean networks. These were pioneered by the work of Kauffman [13], [14] on gene-regulatory systems in the context of evolutionary issues. Boolean networks require the assignment of a label "expressed" or "not expressed" to an individual gene. Just as Boolean networks, the intricate internal dynamics of cellular circuits can exhibit a simple "on/off" behavior [15]. Many cellular systems have been successfully modeled with Boolean logic, such as the segment polarity gene network of *Drosophila melanogaster* [2], or the cell cycle control in yeast [16]. Saez-Rodriguez et al. describe the behavior of T-cell receptor signaling with a quasi-dynamic Boolean model that differentiates between an early and a late phase in the description of the dynamics [17].

In recent years, inference methods for Boolean networks became popular due to their simplicity and the fact that qualitative predictions of large complicated networks are more manageable. Liang et al. [18] developed the REVEAL algorithm, which uses the mutual information between input and output states (e.g., two subsequent measurements of a time series) to infer Boolean dependencies between them. Akutsu et al. [19] presented a simple algorithm that can deal with noisy data to infer such networks. Lähdesmäki et al. [20] and Nam et al. [21] devised a method that reduces the time complexity of this search (by a factor of 2^{2^k} compared to Akutsu et al. [19], where k is the number of input genes of a single Boolean function). Kim et al. [22] proposed an inference algorithm that determines associated genes based

- *M. Hopfensitz, M. Maucher, C. Müssel, C. Wawra, and H.A. Kestler are with the Research Group of Bioinformatics and Systems Biology, Ulm University, Albert-Einstein-Allee 11, D-89081 Ulm, Germany. E-mail: {martin.hopfensitz, markus.maucher, christoph.muessel, hans.kestler}@uni-ulm.de, christian.wawra@googlemail.com.*
- *H. Neumann is with the Institute of Neural Information Processing, Ulm University, D-89069 Ulm, Germany. E-mail: heiko.neumann@uni-ulm.de.*
- *M. Kühl is with the Institute of Biochemistry and Molecular Biology, Ulm University, Albert-Einstein-Allee 11, D-89081 Ulm, Germany. E-mail: michael.kuehl@uni-ulm.de.*

Manuscript received 9 July 2010; revised 17 Dec. 2010; accepted 9 Mar. 2011; published online 22 Mar. 2011.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-2010-07-0164. Digital Object Identifier no. 10.1109/TCBB.2011.62.

on the chi-square test. Compared to the previously mentioned exhaustive reconstruction algorithms, this heuristic method considerably reduces runtime.

A general problem in reconstructing networks from time series data is the large number of gene expression values compared to a relatively low number of temporal measurement points. This is particularly true for microarray data and quantitative real-time PCR data. Serial experiments are time consuming and expensive, and hence, only few measurements from limited numbers of time points are usually available. As a result, the available data are often consistent with multiple network configurations [23], [24]. To address this problem, Martin et al. [23] developed a method that combines similarly regulated genes, infers possible network candidates, and then evaluates their dynamic behavior to draw further conclusions on the gene-regulatory system.

In addition to a lack of measurements, the redundancy inherent in many biological systems often makes it hard to identify the underlying network: when multiple genes are coexpressed, it is difficult to determine which of these genes are truly involved in the network [23].

A further challenge must be met when reconstructing Boolean networks based on real data: the noisiness of gene expression data and the low number of temporal measurement points often yield several plausible binarizations. This makes it hard to determine a "true" binarization threshold. Differences in the binarization results can have strong effects on the architecture of the resulting Boolean networks because a state change for a single gene can lead to many differences in "downstream" functions and gene dependencies. In this paper, we propose a new binarization algorithm which can overcome these problems, particularly if the expression measurements are noisy and the number of candidate functions is high, while at the same time allowing to assess the reliability of the binarization and thus the correctness of the reconstructed functions.

The methods presented here were specifically developed to binarize gene expression data. These binarization algorithms incorporate the data at different scales to produce suitable and robust thresholds even for small numbers of data points. They additionally provide a measure of validity for the found thresholds. Incorporating this knowledge into a network reconstruction method allows for restricting the input of reconstruction algorithms to genes with meaningful thresholds. This is important since the computational cost associated with processing high numbers of genes is a limiting factor in many analyses [22]. Furthermore, constructing networks of fewer, but more reliable candidate genes reduces the workload required for validation of the results, including wet-lab experiments to verify new hypotheses on gene-regulatory systems.

2 Materials and Methods

2.1 Binarization across Multiple Scales (*BASC*)

It is often useful to describe a signal at different resolutions simultaneously: at a fine scale, small details of structure are revealed, while at a coarse scale, slow variation can be made visible. However, at a level of representation that is too detailed, it might be difficult to identify relevant features for the task at hand. At a too coarse level, relevant features might be missing. Here, we utilize the binarizations obtained at different scales for a reliability assessment of the found threshold. This also means that we can reject measurement sequences if they do not yield a reliable binarization. A link between the different scales is determined by the number and location of discontinuities that are used for the approximation of the original step function. We developed two multiscale approaches (denoted by *BASC A* and *BASC B*), each based on three computational stages:

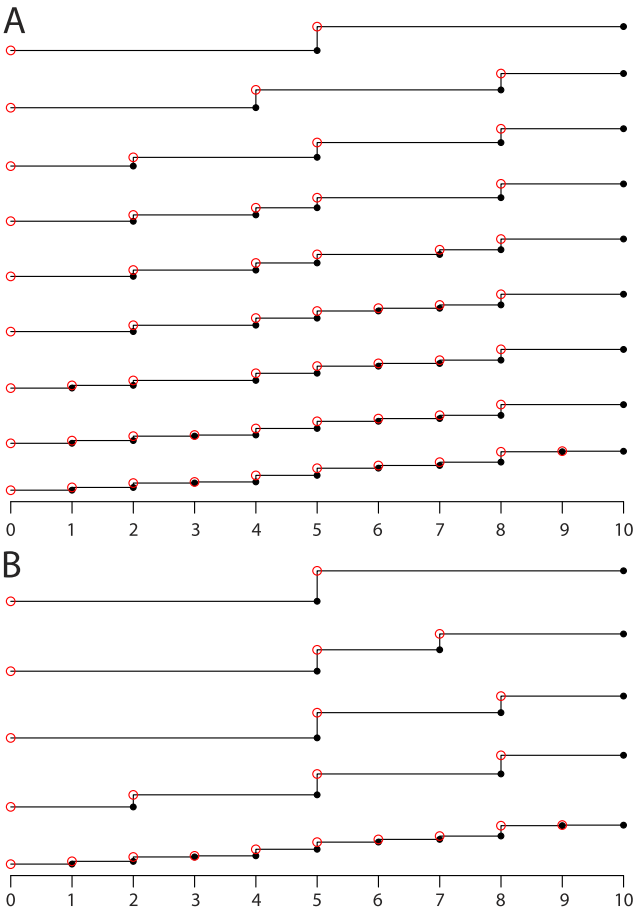

Fig. 1. A family of 1D time series obtained by approximating the original ordered time series (at the bottom of each plot) with step functions whose number of discontinuities decrease from bottom to top. Panel A shows an approximation minimizing the euclidean distance (*BASC A*); Panel B shows a scale space approximation (*BASC B*). The original time series is a sorted list of 10 measurements.

1. *Compute a series of step functions.* An initial step function is obtained by rearranging the original time series measurements in increasing order. Then, step functions with fewer discontinuities are calculated. *BASC A* calculates these step functions in such a way that each minimizes the euclidean distance to the initial step function. *BASC B* obtains step functions from smoothened versions of the input function in a scale-space manner (Fig. 1).
2. *Find strongest discontinuity in each step function.* A strong discontinuity is a high jump size (derivative) in combination with a low approximation error.

3. *Estimate location and variation of the strongest discontinuities.* Based on these estimates, time series measurements of gene expression values can be excluded from network reconstruction.

The input data of the algorithm are a vector of time series measurements of a single gene $\boldsymbol{u} = (u_1, \ldots, u_N) \in \mathbb{R}^N$. The elements of the vector $\boldsymbol{u}$ are sorted in ascending order. Based on this sorted vector, we define a discrete, monotonically increasing step function f with N steps, $N-1$ discontinuities $d_i \in \{1, \ldots, N-1\}$, and function values u_i with $i \in \{1, \ldots, N\}$

$$f(x) = \sum_{i=1}^{N} u_i I_{A_i}(x) \tag{1}$$

where

$$A_i = \begin{cases} (0, d_i], & \text{if } i = 1 \\ (d_{i-1}, N], & \text{if } i = N \\ (d_{i-1}, d_i], & \text{otherwise,} \end{cases} \tag{2}$$

are intervals, and I_A is the indicator function of A defined as follows:

$$I_{A_i}(x) = \begin{cases} 1, & \text{if } x \in A \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Note that in the following stages of the algorithms, a binarization of the data is not possible if all u_i in $\boldsymbol{u}$ are identical.

2.1.1 Compute a Series of Step Functions

A step function f can be approximated by another step function with fewer discontinuities at certain $d_i \in \{1, \ldots, N-1\}$. We here present different ways of approximating step functions, which constitute the main difference between our two *BASC* approaches.

BASC A. This approach calculates optimal step functions with fewer discontinuities n ($n < N-1$) by determining the step function with n discontinuities that minimizes the euclidean distance to the initial step function f. The euclidean distance of two step functions f_1 and f_2 is defined as

$$\|f_1 - f_2\| = \sqrt{\left(\sum_{x=1}^{N} (f_1(x) - f_2(x)\right)^2}. \tag{4}$$

Let S_v be the set of all step functions with at most v discontinuities at certain $d_i \in \{1, \ldots, N-1\}$. An optimal quantization of $f \in S_{N-1}$ with respect to S_n, $1 \le n < N-1$, is defined by

$$s_n^* = \underset{s_n \in S_n}{\operatorname{argmin}} \|f - s_n\| \quad \text{with } 1 \le n < N-1. \tag{5}$$

The distance $\|f - s_n\|$ is the approximation error of a step function s_n with n steps regarding the original function f. This concept of quantization tends to assign break points where the "variation" of f is large [25]. A large variation within one step of the function induces a high approximation error, which means that the strategy of assigning break points to regions of large variation minimizes the approximation error.

Finding the optimal step functions with n steps by enumerating all solutions requires calculating the distances of all possible $\binom{N}{n}$ step functions to the original step function f. Here, the overall complexity to determine the optimal step functions for all $n \in \{1, \ldots, N-2\}$ is $\sum_{n=1}^{N-2} \binom{N}{n} \cdot (N-2)$. A lower complexity can be achieved by a divide-and-conquer strategy, which is detailed in the following.

A step function with n steps can be created by inserting a new step into a step function with $n-1$ steps. In this way, we can determine an optimal step function recursively from partial solutions with fewer steps. Similar to Kämpke and Kober [25], this can be formulated as a Dynamic Programming problem, which allows for identifying all optimal step functions for $n = 1, \ldots, N-2$ simultaneously with a complexity of $O(N^3)$ (with N being the number of data points).

This algorithm iteratively calculates optimal partial solutions and stores the indices of their break points in a matrix Ind. Solutions are rated according to a cost function, which is stored in a second matrix C. Here, $C_i(j)$ contains the cost of a function having exactly j intermediate discontinuities between data points i and N. The cost of a function is the sum of the costs c_{ab} of all steps in the function

$$c_{ab} = \sum_{i=a}^{b} (f(i) - y(a,b))^2, \tag{6}$$

with

$$y(a,b) = \sum_{i=a}^{b} \frac{f(i)}{b-a+1}.$$

The value c_{ab} is the quadratic distance of the values of f between the start point a and the end point b ($a \le b$) of the step to the mean of these values, $y(a,b)$. Algorithm 1 shows the corresponding Dynamic Programming algorithm.

Algorithm 1. Calculate optimal step functions (dynamic programming)
Initialization:
$C_i(0) = c_{iN}, i = 1, \ldots, N$
Iteration:
for $j = 1$ to $N-2$ **do**
  for $i = 1$ to $N-j$ **do**
    $C_i(j) = \min_{d=i,\ldots,N-j}(c_{id} + C_{d+1}(j-1))$
    $Ind_i(j) = \operatorname{argmin}_{d=i,\ldots,N-j}(c_{id} + C_{d+1}(j-1))$

The initialization computes the cost c_{iN} from step i to step N with no intermediate break points (discontinuities). During each iteration, the minimal costs of additional intermediate break points are computed. $Ind_i(j)$ denotes the position of the first break point (out of j in total) with minimal costs from step i to N.

Based on Ind, we are able to reconstruct the break points of all optimal step functions s_n^* (Algorithm 2). The position of the ith break point for an optimal step function with j discontinuities is denoted by $P_i(j)$ with $i \le j$ and $P_i(j) \in \{1, \ldots, N-1\}$.

Algorithm 2. Compute the break points of all optimal step functions
for $j = 1$ to $N-2$ **do**
  $z = j$
  $P_1(j) = Ind_1(z)$
  if $j > 1$ **then**

$z = z - 1$
for $i = 2$ to j **do**
$P_i(j) = Ind_{P_{i-1}(j)+1}(z)$
$z = z - 1$

BASC B. The second algorithm uses discrete scale space representations of the first derivatives of the original step function $f(x)$ to determine a sequence of step functions. We consider the slopes $\Delta(x) = f'(x) \approx f(x+1) - f(x), x \in \mathbb{N}$, and successively apply discrete convolutions with a number of smoothing parameters $\sigma = \sigma_1, \ldots, \sigma_S$. For each of these smoothing parameters, the smoothed slopes are calculated as

$$\Delta_\sigma(x) = \sum_{k=-\infty}^{\infty} \Delta(x) \cdot T_\sigma(x-k),$$

with the kernel

$$T_\sigma(n) = e^{-2\sigma} I_n(2\sigma).$$

Here, I_n is the modified Bessel function of order n

$$I_n(x) = \left(\frac{1}{2}x\right)^n \sum_{k=0}^{\infty} \frac{1}{k!\,\Gamma(n+k+1)} \left(\frac{1}{4}x^2\right)^k,$$

which determines the discrete analogue of the Gaussian kernel [26], [27], [28], [29]. The complexity of one such convolution is $O(N^2)$, where N is the number of data points.

We now search for the local maxima

$$M_\sigma = \{x \mid \Delta_\sigma(x-1) < \Delta_\sigma(x) \wedge \Delta_\sigma(x+1) < \Delta_\sigma(x)\},$$

of each smoothed slope function Δ_σ. Each of these M_σ represents a candidate step function s_{M_σ}, where each maximum x in the set is the location of a discontinuity. We traverse the M_σ beginning at the smallest σ, taking the unique set of such candidate functions over the σ until a function with a single remaining discontinuity is found.

2.1.2 Find Strongest Discontinuity in Each Step Function

For each of the identified step functions, we now identify a single strongest discontinuity. These strongest discontinuities are later accumulated to determine a robust binarization threshold. In the following, we describe the procedure for *BASC A*.

We rate the strength of a discontinuity according to two criteria: first, we assume a large difference in height between the start point $P_i(j)$ and the end point $P_i(j)+1$ of the discontinuity. This means that the points to the left and to the right of the discontinuity form separate groups. The *jump size* (or *contrast*) of the ith discontinuity is defined as

$$h = \begin{cases} y(P_i(j)+1, P_{i+1}(j)) & \\ -y(1, P_i(j)), & i = 1 \wedge j > 1 \\ y(P_i(j)+1, N) & \\ -y(P_{i-1}(j)+1, P_i(j)), & i = j > 1 \\ y(P_i(j)+1, N) & \\ -y(1, P_i(j)), & i = j = 1 \\ y(P_i(j)+1, P_{i+1}(j)) & \\ -y(P_{i-1}(j)+1, P_i(j)), & \text{otherwise.} \end{cases} \quad (7)$$

That is, we consider the mean value y of the points between the $(i-1)$th discontinuity (or the first point) and the ith discontinuity, and the mean value of the points between the ith discontinuity and the $(i+1)$th discontinuity (or the last point). The jump size is the difference of these two mean values.

The second criterion is the *approximation error* of a threshold at the discontinuity with respect to the original function f. The approximation error is defined as

$$e = \sum_{d=1}^{N} (f(d) - z)^2, \quad (8)$$

with

$$z = \frac{f(P_i(j)) + f(P_i(j)+1)}{2}.$$

In other words, we calculate the sum of the quadratic distances of all data points to the threshold z defined by the ith discontinuity.

The two criteria in (7) and (8) are combined into a scoring function

$$q = \frac{h}{e}. \quad (9)$$

A maximal q is achieved by a high jump size in combination with a low approximation error. This definition of q ensures a robust binarization with respect to outliers, since a break point at an outlier would induce a high error and thus a small value of q.

The strongest discontinuities of the optimal step functions with $j = 1, \ldots, N-2$ steps are identified by calculating

$$v_j := \{P_i(j) : i = \text{argmax}_{1,\ldots,j} q\}.$$

BASC B employs the same procedure as *BASC A* with a slight modification: instead of calculating the quality of a step function based on the initial step function $f(x)$, the smoothened versions of the functions

$$f_\sigma(x) = u_1 + \sum_{n=1}^{\lceil x \rceil - 1} \Delta_\sigma(n),$$

for the corresponding σ are considered. This applies to the calculation of y, e, and z.

2.1.3 Estimate Location and Variation of the Strongest Discontinuities

In this last step of binarization, a single threshold is determined. Additionally, the variability of the found thresholds is assessed. Each entry in v is a candidate for the location of a possible binarization threshold.

This step is the same for both approaches *BASC A* and *BASC B*. Based on v, we define the single binarization threshold t as follows:

$$t = \frac{f(\lfloor \tilde{v} \rfloor + 1) + f(\lfloor \tilde{v} \rfloor)}{2},$$

where $\tilde{v}$ is the median of the values in v. A binarized vector $\boldsymbol{u}' = (u'_1, \ldots, u'_N) \in \mathbb{B}^N$ can now be calculated

$$u'_i = \begin{cases} 0 & u_i \le t \\ 1 & u_i > t \end{cases} \quad \text{for } i \in \{1, \ldots, N\}.$$

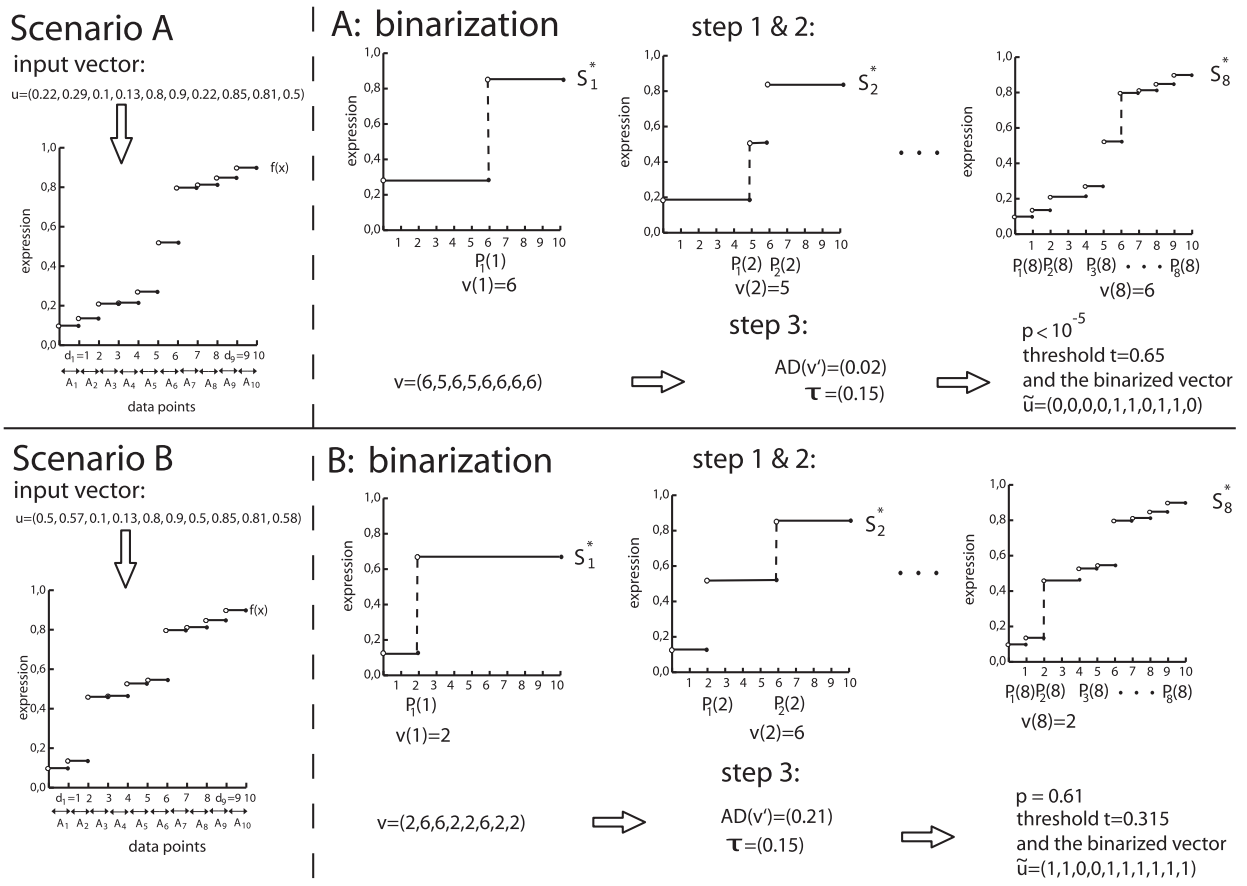

Fig. 2. Examples of the binarization of two sorted input sequences using *BASC A* (the corresponding figure for *BASC B* can be found in the supplementary material, Figure S10, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2011.62,). For input vector (A), the consecutive step functions suggest two adjacent input vector indices 5 and 6 for the threshold calculation. A meaningful threshold is determined. For vector (B), the threshold indices are far apart (2 and 6). This indicates an unreliable binarization.

In a further step called *noisy gene elimination*, we assess the validity of the binarization by examining the variability of the strongest discontinuities v. The idea of this approach is to check if the proposed locations of possible binarization thresholds (optimal break points) for the chosen step functions with different j stay within a small range (in the best case, they are identical). A small range means that the values of v vary only little around the location that is chosen for calculating the binarization threshold. This characteristic is illustrated in Fig. 2 for *BASC A*: if, for instance, the vector of positions of the strongest discontinuities $v = (6, 5, 6, 5, 6, 6, 6, 6)$ (Scenario A), only the fifth and the sixth value in the input vector $\mathbf{u}$ are threshold candidates. Hence, only the sixth value of $\mathbf{u}$ is binarized ambiguously by the $N-2$ reduced step functions, which means that the binarization is comparatively stable. By contrast, if $v = (2, 6, 6, 2, 2, 6, 2, 2)$ (Scenario B), the two possible thresholds are determined by the second and the sixth value of $\mathbf{u}$, i.e., the thresholds are far apart. In this case, four values are binarized ambiguously. The corresponding example for *BASC B* is provided in Figure S10 of the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2011.62.

To make the variability of the thresholds comparable, the values of v are divided by $N-1$, resulting in the normalized vector v'. For the random variable V from which v' was sampled, we expect the mean deviation from its median $\tilde{V}$ to be smaller than a predefined value $\tau \in (0, 1]$. The hypotheses of the corresponding test are

$$H_0 : E(|V - \tilde{V}|) \geq \tau \text{ and } H_1 : E(|V - \tilde{V}|) < \tau.$$

For a sample x with N elements, the average deviation from the median $\tilde{x}$ can be calculated as follows:

$$AD(x) = \frac{1}{N} \sum_{j=1}^{N} |x_j - \tilde{x}|. \tag{10}$$

Since the data in v' are correlated, a moving-blocks Bootstrap test [30], [31] based on the distribution of the test statistic $t_0 = \tau - AD(v')$ under the null hypothesis is used. For the moving-blocks bootstrap test, blocks of length l are drawn with replacement from $v'_1, \ldots, v'_{N-2}$ and concatenated to form a bootstrap sample $v^*_1, \ldots, v^*_{N-2}$. The theoretically optimal block length l for one-sided tests is of the order $(N-2)^{1/4}$[31], [32]. Typically, the bootstrap-based one-sided distribution function is a consistent estimator for a wide range of values of the block length l [31], [32]. We use a block length l of $(N-2)^{1/4} + 1$, since $(N-2)^{1/4} + 1 \geq 2 \forall N \geq 3$. We call the number of blocks b, so that $(N-2) = b \cdot l$. If $(N-2)$ is not a multiple of l, the last selected block is shorter to obtain a bootstrap sample of size $(N-2)$. For each bootstrap sample, the test statistic

$$t(v^*) = AD(v') - AD(v^*) \tag{11}$$

is calculated. The $t(v^*)$ calculated from B samples then represent the distribution of the test statistic, and a p-value is calculated as follows:

$$p = \#\{t(v^*) \geq t_0\}/B. \tag{12}$$

This p-value is an estimate of the probability that the average deviation of the thresholds from their respective median is greater or equal to τ.

$\tau \in (0, 1]$ is set to balance experimental specificity and sensitivity. Decreasing τ can reduce the number of false binarization thresholds accepted as significant, but may also reduce the number of true binarization thresholds that are significant. Since v' is independent of N, the same τ can be used for experiments with different N and results in the same experimental specificity and sensitivity. In our experiments, we set the significance level α to 0.05. In the subsequent network reconstruction, this makes a reduction of candidate genes possible.

3 Experiments

To assess the performance of our binarization algorithms, we designed three experimental setups. In two settings, we evaluated the influence of the binarization on the reconstruction of Boolean networks

- In the first setting, artificial real-valued time series were obtained from randomly created Boolean networks. After applying our binarization algorithms (*BASC*), we measured the performance of a subsequent network reconstruction. Furthermore, we compare the performance to other binarization approaches (see Section 3.1).
- In the second setting, we binarized real time series from the yeast cell cycle [33], [34] using *BASC A*, *BASC B*, and several other binarization approaches. We then compared the reconstructed functions to known biological dependencies (see Section 3.2).

Moreover, we evaluated the robustness of our algorithm to noise. We generated artificial microarray data on the basis of a noise model and measured the difference between binarizations of the data before and after the addition of noise. Again, the results were compared to other binarization techniques. The experimental setup and the results of these experiments are described in the supplementary material, which can be found on the Computer Society Digital Library.

3.1 Reconstruction of Random Boolean Networks

To test the utility of our binarization algorithm for the reconstruction of regulatory networks from time series measurements, we generated artificial time series data based on random Boolean networks [13], [14], reconstructed possible Boolean functions after adding noise, and analyzed the reconstruction process. A Boolean network consists of a set of genes represented by Boolean variables $X = \{X_1, \ldots, X_n\}$ and a set of transition functions $F = \{f_1, \ldots, f_n\}$, one for each variable. These transition functions map an input of the Boolean variables in X to a Boolean value. They usually only depend on a subset of k variables in X. A transition function with k inputs can be represented by a truth table with 2^k entries. We call a Boolean vector $\mathbf{x}(t) = (x_1(t), \ldots, x_n(t))$ the *state* of the network at time t. Then, the next state of the network $\mathbf{x}(t+1)$ is calculated by applying all transition functions $f_i(\mathbf{x}(t))$ synchronously. When reconstructing networks from time series, successive states of the reconstructed network should match successive time series measurements.

According to Kauffman [13], [14], random Boolean networks with n genes were generated. A time series was created by first selecting an initial network state, i.e., an n-dimensional vector with the entries chosen randomly and uniformly from $\{0, 1\}$. Starting with the initial state, the successor states were determined by synchronously updating the values of the genes using the former states as inputs. With this procedure, we generated a time series of length m. We simulated noisy data by adding a noise term with normal distribution and different standard deviations: with a probability q, we added noise with a standard deviation of 0.8 (high noise) to all time values of a certain gene. Otherwise, we used a standard deviation of 0.1. To reconstruct a Boolean network from the time series data, we used the best fit extension algorithm [20]. This approach finds the best predictor for each gene by computing the least possible error w (defined in [20]) that can be achieved by a Boolean function on the input data for each possible combination of k input variables in comparison to other approaches that always require zero error.

3.1.1 Experimental Setup

For the time series data without noise, we determined all consistent functions F_i for each gene i by using a simple consistency algorithm [20]. A function f is *consistent* with the time series for the ith gene if f could have produced these measurements (i.e., it predicts the time series with $w = 0$). F_i is called the set of *true solutions* and contains all consistent functions for the ith gene. The time series do not necessarily determine the underlying original function unambiguously, and thus more than one function per gene can be consistent with the given data. However, since time series were generated from a Boolean network, F_i contains at least one true solution representing the original function of the gene in this network. In the case of noisy time series data, these real-valued measurements need to be binarized before reconstructing the Boolean network. For each gene, we binarized the given m noisy values using our algorithms that additionally predict the quality of the binarization (expressed by the p-value). We denote the p-value for a gene i by p_i. If $p_i \leq 0.05$, the binarization is recommended. For the binarized noisy time series, we determined the set $\tilde{F}_i$ of all consistent functions for a gene i by applying the best fit extension procedure. Clearly, $\tilde{F}_i$ can be different from F_i due to a wrong binarization. We distinguished the reconstructed Boolean functions $\tilde{f}_i \in \tilde{F}_i$ as follows:

$$\tilde{f}_i \in \begin{cases} \tilde{F}_i^t & \text{if } p_i \leq 0.05 \wedge p_j \leq 0.05 \\ & \forall j : j \text{is an input gene of } \tilde{f}_i \\ & \textit{recommended by our algorithm} \\ \tilde{F}_i^f & \text{otherwise} \\ & \textit{rejected by our algorithm.} \end{cases} \tag{13}$$

That is, a function is recommended if the corresponding gene as well as all its input genes pass the quality criteria of the binarization algorithm. Hence, functions in $\tilde{F}_i^t$ are more likely to be reconstructed accurately. To finally analyze the benefit of our binarization algorithms, we performed two experiments **A** and **B**.

In the first experiment **A**, we performed multiple simulations with randomly created networks and determined the number of reconstructed functions $\tilde{f}_i$ that were recommended by the algorithms ($\tilde{f}_i \in \tilde{F}_i^t$) and not recommended by the algorithms ($\tilde{f}_i \in \tilde{F}_i^f$), respectively. Both possibilities were further distinguished depending on whether the functions were included in the set of true solutions before the addition of noise or not, that is, if $\tilde{f}_i \in F_i$ or $\tilde{f}_i \notin F_i$. Using the above criteria, we can divide the reconstructed functions into four categories:

- *True positives (TP)*. Recommended by *BASC* ($p \leq 0.05$) and in the set of true solutions ($\tilde{f}_i \in F_i$).
- *False negatives (FN)*. Not recommended by *BASC* ($p > 0.05$), but in the set of true solutions ($\tilde{f}_i \in F_i$).
- *False positives (FP)*. Recommended by *BASC* ($p \leq 0.05$), but not in the set of true solutions ($\tilde{f}_i \notin F_i$).
- *True negatives (TN)*. Not recommended by *BASC* ($p > 0.05$) and not in the set of true solutions ($\tilde{f}_i \notin F_i$).

To determine whether a function is in the set of true solutions, we define two Boolean functions to be equal if they have the same input genes and if the truth tables are not inconsistent. Two truth tables are inconsistent if for any input setting, one table has a 0 entry and the other one has a 1 entry, or vice versa. Note that the reconstructed Boolean functions are not necessarily fully defined, but can have undefined entries (*don't cares*). Thus, we do not claim a complete concordance of the truth tables.

For the experiment, we generated 1,000 time series from random networks with $n = 10$ genes using a fraction of genes with high noise of $q = 0.6$, $k = 3$ input genes per function, and a time series with $m = 20$ states. We varied τ in the range of $[0.001, 1]$ and counted true positives, false negatives, false positives, and true negatives for each value of τ. From these values, sensitivity and specificity can be calculated as $Sens = TP/(TP + FN)$ and $Spec = TN/(TN + FP)$. By comparing sensitivity and specificity at different values of τ, we can analyze how τ controls the trade-off of the two rates.

In experiment **B**, we additionally compared our novel binarization algorithms to other binarization methods. The StepMiner algorithm [35] fits one-step and two-step functions to the real-valued expression curve of the time series, which amounts to considering $O(N^2)$ step functions. It then assesses how well these functions fit the data in a statistical test. Details on the approach can be found in the discussion of existing binarization approaches below. The significance level for StepMiner's testing procedure was also set to $\alpha = 0.05$. We also employ the mean of the time series data as the binarization threshold, which can be calculated in $O(N)$. A further binarization approach is the k-means cluster algorithm [36] with $k = 2$. k-means has a complexity of $O(I \cdot N)$, where I is the number of iterations. In addition, we included two baseline methods: the random threshold method chooses a random data point as a threshold and binarizes all values less than or equal to this value to 0 and all greater values to 1. The random prototype method is a k-means algorithm with only one iteration: two data points are chosen randomly, and the other data points are assigned to the nearest of these points. The group with the smaller values is binarized to 0, and the group with the higher values is binarized to 1.

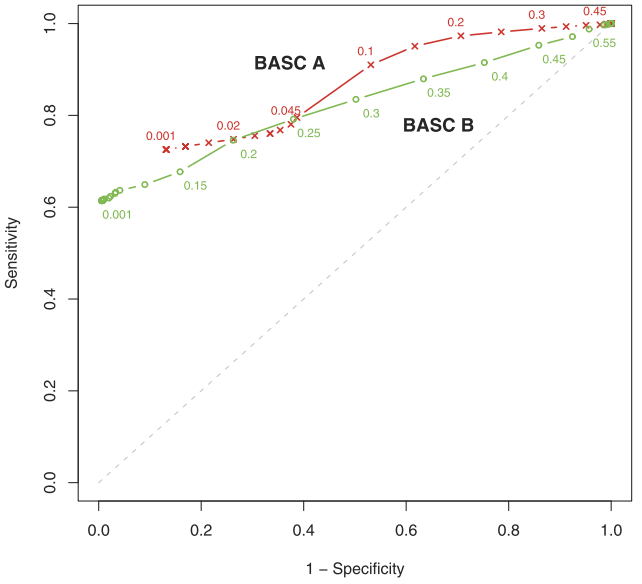

Fig. 3. Sensitivity and specificity of *BASC A* (red cross) and *BASC B* (green circle) for different levels of τ (printed next to the points).

We also used two variants of our algorithms: the first variant eliminates noisy genes according to condition (15), the second one does not.

The calculation of sensitivity and specificity as in experiment **A** requires all algorithms to include a testing procedure which rejects irrelevant genes. As experiment **B** involves several methods without such a testing procedure, we compared them by calculating the positive predictive value $PPV = TP/(TP + FP)$, i.e., the fraction of reconstructed functions that were found in the set of true solutions F_i over all genes i.

For each fraction of genes with high noise $q \in \{0.0, 0.1, 0.2, \ldots, 0.8, 0.9\}$, we performed 1,000 tests with $n = 10$ genes, $k = 3$ input genes per function, and a time series with $m = 20$ states. We set $\tau = 0.001$, since we wanted to reject as much noisy time series data (characterized by ambiguous binarization thresholds) as possible. The procedure is illustrated in Figure S1 in the supplementary material, which can be found on the Computer Society Digital Library.

3.1.2 Results

The results of experiment **A** can be visualized in a ROC curve, plotting $1 - Sens$ against $Spec$ for the different levels of τ (see Fig. 3). Here, the diagonal denotes the baseline for randomly recommending or rejecting functions.

Both approaches stay significantly above the random baseline. Varying the τ parameter is an effective way of setting the trade-off of sensitivity and specificity: for the lowest level of $\tau = 0.001$, the algorithms achieve a sensitivity of 73 percent (*BASC A*)/61 percent (*BASC B*) in combination with a specificity of 87 percent (*BASC A*)/ 99 percent (*BASC B*). Above a certain level of τ (around

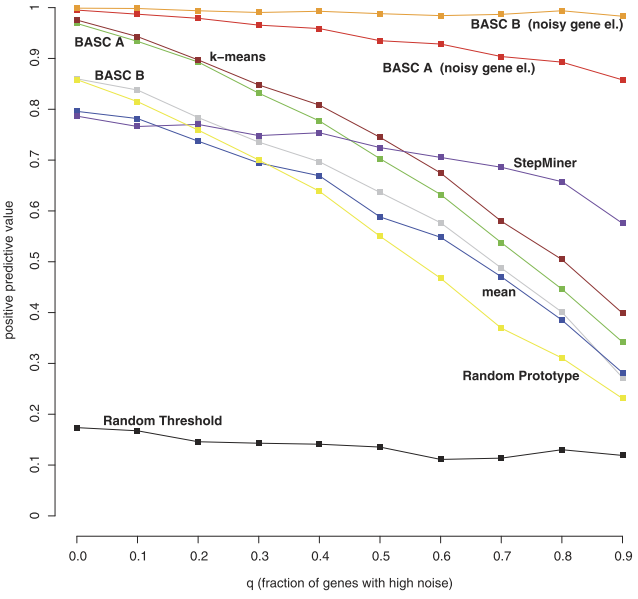

Fig. 4. Average positive predictive value in 1,000 runs for various fractions of genes with high noise q. The reconstruction was performed on the basis of our two multiscale binarization approaches with and without noisy gene elimination, the mean-based binarization, 2-means-based binarization, StepMiner, and two random baselines. Details on the positive predictive values and their standard deviations are supplied in Table S1 of the supplementary material, which can be found on the Computer Society Digital Library.

0.5), the sensitivity approaches 100 percent, and the specificity approaches 0 percent, which means that nearly all solutions are recommended by the algorithms. A sensitivity of 100 percent is finally reached for $\tau = 0.6$ (*BASC A*) and $\tau = 0.8$ (*BASC B*).

Fig. 4 depicts the results of experiment **B**. These results again demonstrate the great improvement of the positive predictive value which can be achieved using noisy gene elimination: when the fraction of genes with high noise is increased, a reconstruction on the basis of k-means, the mean-based binarization, and our binarization algorithms without noisy gene elimination mostly retrieves irrelevant functions (e.g., less than 40 percent of the reconstructed functions are in the set of true solutions when 90 percent of the genes were highly noisy). *BASC A* performs better than *BASC B* without elimination of noisy genes. StepMiner performs worse than the other algorithms for lower levels of noise, but outperforms the algorithms without noisy gene elimination for higher levels of noise. This is due to the statistical testing procedure employed by StepMiner which is similar to our noisy gene elimination. The corresponding *BASC* approaches with noisy gene elimination outperform all other approaches—including StepMiner—by rejecting most of the irrelevant solutions. For *BASC A*, the fraction of relevant functions always stays above 85 percent. The scale space approach (*BASC B*) even achieves an accuracy of at least 98 percent. However, this is achieved at the cost of rejecting many relevant solutions: in comparison to *BASC A*, the fraction of rejected solutions is higher. As seen in experiment **A**, the strictness of *BASC* can be configured appropriately using the parameter τ. In experiment **B**, we use a very strict setting of $\tau = 0.001$.

The baseline method Random Threshold yields the worst performance with no more than 18 percent of true solutions. The other baseline approach, Random Prototype, sometimes outperforms the mean-based binarization and StepMiner, in particular for low levels of noise. The k-means approach always stays above this baseline which is equivalent to the initialization of k-means.

The number of overall reconstructed functions decreases with increasing noise for all algorithms, which means that only few functions with $k = 3$ inputs that match the highly noisy time series can be found (see Table S1 in the supplementary material, which can be found on the Computer Society Digital Library, for details). This is independent of the question whether the solutions are true solutions or not. For all approaches, the number of reconstructed functions varies strongly in the 1,000 runs. This is due to the high amount of randomness in the process: first, the structure and dynamics of the original network is generated at random. Second, the start point of the time series—which determines the amount of information in the series—is chosen randomly. Third, random noise is added to the data. However, the variation of the positive predictive value across these different networks is reduced considerably by noisy gene elimination: the *BASC* approaches with noisy gene elimination yield the smallest variance in the fraction of true solutions.

3.2 Reconstruction from Yeast Gene Expression Data

To test the performance of our binarization algorithm on real data, we used the yeast gene expression data of Cho et al. [34], which is included in the Spellman et al. data [33] available at http://cellcycle-www.stanford.edu. This data were derived from microarray analysis of yeast cultures and synchronized in late G1 (cdc28 cells). It contains 17 measurements. In this data set, the function and identity of many genes as well as their transcriptional regulators are known (see [37]). The measurements were taken at fixed intervals and cover two complete cell cycles. These properties ensure that there is some redundancy in the data, which can help to stabilize a network reconstruction process. The data set was preprocessed as described in the supplementary material, which can be found on the Computer Society Digital Library.

We considered a well-studied subset of cell cycle genes described by Simon et al. [37]. This list comprises eight transcriptional regulators and 92 genes. Two of the regulators are complexes of multiple genes: MBF (a complex of Mbp1 and Swi6) and SBF (a complex of Swi4 and Swi6). As the time series only contains single genes, we replaced them by Mbp1 and Swi4, respectively. Six of the listed 92 genes showed more than 20 percent missing values and were excluded. Hence, we employed the remaining 86 genes in the experiment.

3.2.1 Experimental Setup

We binarized the data using our algorithm with and without noisy gene elimination according to condition (13). Furthermore, we performed binarizations using the mean value as a threshold, on the basis of the k-means algorithm with $k = 2$, with the StepMiner algorithm and with the two random baseline algorithms, Random Prototype and Random Threshold. Based on the resulting binary

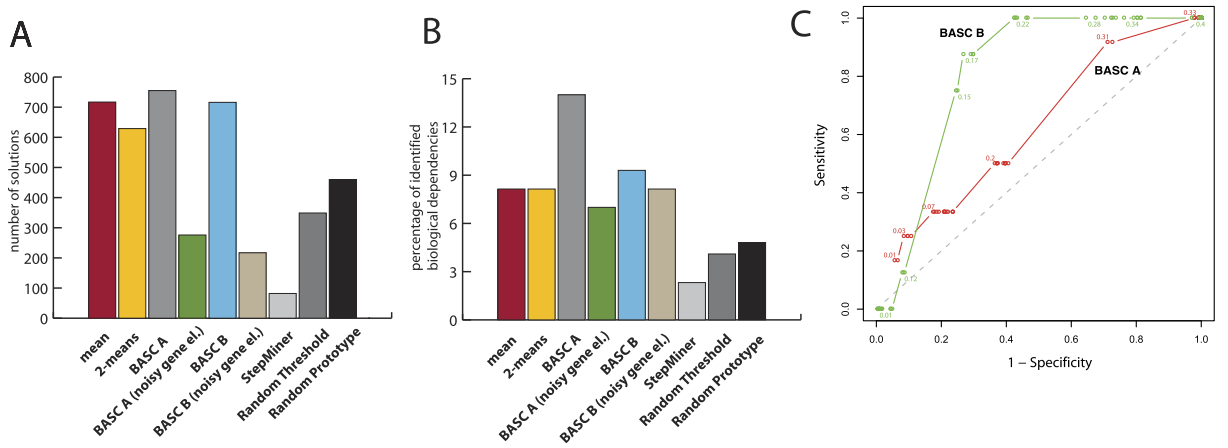

Fig. 5. Panel A: total number of solutions returned by mean-based binarization, 2-means binarization, StepMiner, the multiscale binarization approaches with and without noisy gene elimination, and two random baselines on the yeast time series. Panel B: percentage of known biological functions identified by the algorithms. The total number of known biological dependencies is 86. Panel C: ROC curve for *BASC A* (red) and *BASC B* (green) for various levels of τ.

data, we reconstructed possible Boolean functions using the best fit extension algorithm [20]. For the binarization of the data with *BASC*, we set $\tau = 0.2$. This setting allows a slight variance of the threshold locations.

To verify the biological relevance of the reconstructed Boolean functions, we compared them to known interactions between the eight transcription factors and the 86 cell cycle genes. We extracted these dependencies from the YEASTRACT database, which comprises regulatory associations of budding yeast based on more than 1,200 publications [38], and from the TRANSFAC database [39]. Our gold standard consists of all documented regulations between the transcription factors and the genes in these databases.

For StepMiner and *BASC*, only functions which exhibited a p-value smaller than or equal to 0.05 for both the corresponding gene itself and the input genes of the function were recommended (see condition (13)).

For each method, we counted the number of consistencies between the reconstructed Boolean functions and the known biological functions (i.e., the instances where all input genes of a gene from a Boolean solution were identical to the transcriptional regulators extracted from the databases).

In a second setting, we measured sensitivity and specificity of *BASC* for various levels of $\tau \in [0.01, 1]$.

3.2.2 Results

The results of these experiments are illustrated in Fig. 5. Panel A shows that our algorithms are able to reduce the set of candidate solutions returned by the reconstruction algorithm significantly. At the same time, the number of solutions among the candidate solutions that match biologically known functions exactly is comparable to the other algorithms (Panel B). *BASC* algorithms successfully eliminate false solutions, while being able to retain most of the biologically meaningful ones. Without noisy gene elimination, *BASC A* retrieves much more true solutions than any of the other algorithm, but also yields a greater number of false positives. StepMiner yields only few overall solutions, but also few true solutions. One reason for this is that the solutions returned by the reconstruction algorithm on a time series binarized by StepMiner are mostly constant or have only one input gene, which means that there are drastically fewer alternatives than for functions with more inputs (data not shown).

Panel C shows the ROC curve of *BASC* on the yeast data for various levels of τ. For higher levels of τ, *BASC B* quickly achieves a sensitivity of close to 100 percent, with a specificity of up to 60 percent. For very strict settings of τ, *BASC B* rejects all solutions, yielding a sensitivity of 0 percent and a specificity of 100 percent. *BASC A* mostly has a lower sensitivity than *BASC B* in this setting. This corresponds to the observation in Panel B that *BASC A* retrieves a higher number of true solutions than the other approaches, but rejects about half of them in the process of noisy gene elimination.

4 Comparison of *BASC* to Other Binarization Methods

BASC is a general binarization technique that utilizes a multiscale view of the complete data to determine robust thresholds. Analyzing the data at multiple scales also allows for assessing the reliability of the binarization in a statistical testing procedure. It is nonparametric in the sense that it does not assume specific distributions in the data.

Another approach that provides a measure of reliability for the binarized results is the StepMiner method already introduced in the experimental section [35]. The main focus of this method is to find the point of time at which the expression level changes in a time series of measurements. It is, therefore, dependent on the temporal order of the measurements. In contrast to most common binarization approaches, StepMiner does not use a global binarization threshold. Instead, it tries to approximate the time course of expression levels by binary one-step or two-step functions. That is, StepMiner expects the expression level not to change more than twice in the time course. In a statistical test, it then assesses whether these models are suitable to explain the data. The test hypotheses do not make a statement on how well the data can be binarized in general. Although *BASC* fits step functions to the data as well, the two approaches are different: The original StepMiner approach strongly focuses on fitting step functions to temporally ordered measurements. In another context,

StepMiner was also applied to sorted input data to produce a ternary quantization, but this means that the testing procedure is no longer applicable [7]. *BASC* aims at separating groups of similar values independent of the original order of the measurements. In the context of the original order of the data, such a grouping can lead to an arbitrary number of discontinuities and is not restricted to one or two steps. By analyzing the data at multiple scales, *BASC* assesses whether it is possible to divide the data into two stable groups.

Some other binarization and quantization approaches have also been specifically designed for biological data. Dimitrova et al. [40] determine the optimal number of discretization states using a modified single-linkage clustering approach. They state that this method is particularly suitable for short time series.

Many techniques for biological data make use of prior knowledge. Hakamada et al. [41] and Hirose et al. [42] use known gene interactions to determine a single binarization threshold for all genes. Pe'er et al. [43] make use of additional biological data from repeated wild-type experiments to estimate the distribution of expression levels. Similarly, Camillo et al. [44] employ experimental replicates to determine deviations of the expression level from a baseline distribution. The integration of additional knowledge and data possibly yields more accurate results than more general approaches, but restricts the use of the algorithms to specific applications. Furthermore, this requires the availability of such data.

Taking the mean value as a threshold is probably one of the simplest binarization methods. Although this approach is highly sensitive to data with unbalanced distributions of high and low values, it has also been employed as a preprocessing step for reverse engineering of Boolean networks [22].

Zhou et al. [5] suggest a mixture model for binarization. This approach fits two overlaid log-normal distributions to gene expression measurements to determine a threshold. The edge detection binarization approach by Shmulevich and Zhang [6] chooses the binarization threshold according to the first location in the sorted input values that exhibits a difference between two successive values greater than a predefined value. This threshold criterion is similar to the way the strongest discontinuities are chosen in *BASC*. However, *BASC* additionally takes into account the approximation error of the threshold.

Binarization is also an issue in many other research areas. In the context of image segmentation, histogram-based and entropy-based methods are commonly employed. Such thresholding techniques are often associated with a significant loss of information [45]. Other approaches, in particular algorithms used for clustering of gene expression data, are nondeterministic and can produce different results with different initialization settings [46].

5 Discussion and Conclusion

A general problem of reconstruction algorithms is the large number of genes that can be measured in parallel compared to the relative low number of temporal measurement points [24]. Contemporary microarray technologies allow for routinely analyzing expression levels of hundreds of thousands of genes simultaneously [47], [48], [49]. However, these experiments are laborious and costly, so that time series experiments will currently only cover a very limited number of typically no more than 10 to 20 time points.

Together with the inherent noisiness of gene expression data [50], this often results in ambiguous thresholds when binarization of the data is performed. This poses serious problems for the reconstruction of Boolean networks, as differences in the binarization results can have strong effects on the resulting Boolean models. A state change from 0 to 1, or vice versa, for a single gene can cause different functions and gene dependencies in many "downstream" elements of the network. Therefore, binarization has to be considered a crucial step in the network construction process.

For this reason, we propose a binarization across multiple scales to yield adequate and robust thresholds even for data sets with small numbers of data points. The two approaches *BASC A* and *BASC B* differ in the way of scaling: *BASC B* is a true scale space approach and defines coarse and fine scales by the amount of smoothing that is applied to the function. This means that the "level of detail" is decreasing from a fine scale to a coarse scale. By contrast, *BASC A* ensures the minimal quantization error for each scale. Hence, traversing the scales from a fine scale to a coarse scale implies a monotonically increasing quantization error in this case. A more detailed discussion on this can be found in the supplementary material, which can be found on the Computer Society Digital Library.

By applying an additional validity measure to the binarization results, our method further allows to filter out the most suitable solutions for a given reconstruction problem. As a result, the algorithms produce drastically fewer and at the same time more reliable network models, which allows researchers to formulate new hypotheses on gene-regulatory networks with a much higher confidence. Although our method cannot completely eliminate the problem of low temporal resolution and more exhaustive measurements will always be desirable, it can serve to considerably reduce the effort (including "wet lab" experiments) required to validate newly developed hypotheses.

The *BASC* method is, thus, ideally suited for analyses involving Boolean network reconstruction, especially if conventional methods would result in a multiplicity of solutions. The algorithms can also be helpful in the reconstruction of probabilistic Boolean networks when reconstruction is based on binarized data [51].

Moreover, other areas of data analysis may benefit from our approach as well. In microarray gene expression data analysis, for example, promising results have recently been achieved with clustering and classification methods working entirely in the binary domain [5], [6], [52]. Although we demonstrated the use of the *BASC* algorithms in a biological context, *BASC* can be applied to binarize any type of real-valued data. In particular, it is not restricted to time series. Furthermore, it does not depend on external knowledge or data. *BASC* can be applied to spatial samplings even in multiple dimensions, since it derives its decisions from the signal data and its statistics in a parameter-free fashion.

Acknowledgments

This work is supported by the German Science Foundation (SFB 518, Project C5), the Stifterverband für die Deutsche Wissenschaft (HAK), the Graduate School of Mathematical

Analysis of Evolution, Information and Complexity at the University of Ulm (CM, HN, HAK), and the International Graduate School (GSC 270) in Molecular Medicine Ulm (CW, MK, HAK). Martin Hopfensitz, Christoph Müssel, and Christian Wawra contributed equally. Hans A. Kestler is the corresponding author.

REFERENCES

[1] H. de Jong, "Modeling and Simulation of Genetic Regulatory Systems: A Literature Review," *J. Computational Biology,* vol. 9, no. 1, pp. 67-103, 2002.

[2] R. Albert and H. Othmer, "The Topology of the Regulatory Interactions Predicts the Expression Pattern of the Segment Polarity Genes in Drosophila Melanogaster," *J. Theoretical Biology,* vol. 223, no. 1, pp. 1-18, 2003.

[3] T. Helikar, J. Konvalina, J. Heidel, and J.A. Rogers, "Emergent Decision-Making in Biological Signal Transduction Networks," *Proc. Nat'l Academy of Sciences USA,* vol. 105, no. 6, pp. 1913-1918, 2008.

[4] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, "Using Bayesian Networks to Analyze Expression Data," *J. Computational Biology,* vol. 7, nos. 3/4, pp. 601-620, 2000.

[5] X. Zhou, X. Wang, and E.R. Dougherty, "Binarization of Microarray Data on the Basis of a Mixture Model," *Molecular Cancer Therapeutics,* vol. 2, no. 7, pp. 679-684, 2003.

[6] I. Shmulevich and W. Zhang, "Binary Analysis and Optimization-Based Normalization of Gene Expression Data," *Bioinformatics,* vol. 18, no. 4, pp. 555-565, 2002.

[7] D. Sahoo, D.L. Dill, A.J. Gentles, R. Tibshirani, and S.K. Plevritis, "Boolean Implication Networks Derived from Large Scale, Whole Genome Microarray Datasets," *Genome Biology,* vol. 9, no. 10, p. R157, Jan. 2008.

[8] F. Markowetz and R. Spang, "Inferring Cellular Networks—A Review," *BMC Bioinformatics,* vol. 8(Suppl 6):S5, 2007.

[9] E. Lee, A. Salic, R. Krüger, R. Heinrich, and M.W. Kirschner, "The Roles of APC and Axin Derived from Experimental and Theoretical Analysis of the Wnt Pathway," *PLoS Biology,* vol. 1, no. 1, pp. 116-132, 2003.

[10] C. Wawra, M. Kühl, and H.A. Kestler, "Extended Analyses of the Wnt/β-Catenin Pathway: Robustness and Oscillatory Behaviour," *FEBS Letters,* vol. 581, no. 21, pp. 4043-4048, 2007.

[11] N. Dojer, A. Gambin, A. Mizera, B. Wilczyski, and J. Tiuryn, "Applying Dynamic Bayesian Networks to Perturbed Gene Expression Data," *BMC Bioinformatics,* vol. 7, article 249, 2006.

[12] K. Murphy and S. Mian, "Modelling Gene Expression Data Using Dynamic Bayesian Networks," technical report, Computer Science Division, Univ. of California, Life Sciences Division, Lawrence Berkely Nat'l Laboratory, 1999.

[13] S.A. Kauffman, "Metabolic Stability and Epigensis in Randomly Constructed Genetic Nets," *J. Theoretical Biology,* vol. 22, no. 3, pp. 437-467, 1969.

[14] S.A. Kauffman, *The Origins of Order: Self-Organization and Selection in Evolution.* Oxford Univ. Press, 1993.

[15] O. Brandman, J.E. Ferrell, L. Rong, and T. Meyer, "Interlinked Fast and Slow Positive Feedback Loops Drive Reliable Cell Decisions," *Science,* vol. 310, no. 5747, pp. 496-498, 2005.

[16] F. Li, T. Long, Y. Lu, Q. Ouyang, and C. Tang, "The Yeast Cell-Cycle Network is Robustly Designed," *Proc. Nat'l Academy of Sciences USA,* vol. 101, no. 14, pp. 4781-4786, 2004.

[17] J. Saez-Rodriguez, L. Simeoni, J. Lindquist, R. Hemenway, U. Bommhardt, B. Arndt, U. Haus, R. Weismantel, E. Gilles, S. Klamt, and B. Schraven, "A Logical Model Provides Insights into T Cell Receptor Signaling," *PLoS Computational Biology,* vol. 3, no. 8, p. e163, 2007.

[18] S. Liang, S. Fuhrman, and R. Somogyi, "REVEAL, A General Reverse Engineering Algorithm for Inference of Genetic Network Architectures," *Proc. Pacific Symp. Biocomputing,* R.B. Altman, A.K. Dunker, L. Hunter, and T.E.D. Klein, eds., vol. 3, pp. 18-29, 1998.

[19] T. Akutsu, S. Miyano, and S. Kuhara, "Inferring Qualitative Relations in Genetic Networks and Metabolic Pathways," *Bioinformatics,* vol. 16, no. 8, pp. 727-734, 2000.

[20] H. Lähdesmäki, I. Shmulevich, and O. Yli-Harja, "On Learning Gene Regulatory Networks under the Boolean Network Model," *Machine Learning,* vol. 52, nos. 1/2, pp. 147-167, 2003.

[21] D. Nam, S. Seo, and S. Kim, "An Efficient Top-Down Search Algorithm for Learning Boolean Networks of Gene Expression," *Machine Learning,* vol. 65, no. 1, pp. 229-245, 2006.

[22] H. Kim, J.K. Lee, and T. Park, "Boolean Networks Using the Chi-Square Test for Inferring Large-Scale Gene Regulatory Networks," *BMC Bioinformatics,* vol. 8, article 37, 2007.

[23] S. Martin, Z. Zhang, A. Martino, and J.-L. Faulon, "Boolean Dynamics of Genetic Regulatory Networks Inferred from Microarray Time Series Data," *Bioinformatics,* vol. 23, no. 7, pp. 866-874, 2007.

[24] D. Nam, S. Yoon, and J. Kim, "Ensemble Learning of Genetic Networks from Time-Series Expression Data," *Bioinformatics,* vol. 23, no. 23, pp. 3225-3231, 2007.

[25] T. Kämpke and R. Kober, "Discrete Signal Quantization," *Pattern Recognition,* vol. 32, no. 4, pp. 619-634, 1999.

[26] A. Witkin, "Scale Space Filtering," *Proc. Int'l Joint Conf. Artificial Intelligence,* pp. 1019-1022, 1983.

[27] J.J. Koenderink, "The Structure of Images," *Biological Cybernetics,* vol. 50, no. 5, pp. 363-370, 1984.

[28] T. Lindeberg, "Scale-Space for Discrete Signals," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 12, no. 3, pp. 234-254, Mar. 1990.

[29] A. Cunha, R. Teixeira, and L. Velho, "Discrete Scale Spaces via Heat Equation," *Proc. 14th Brazilian Symp. Computer Graphics and Image Processing,* pp. 68-75, 2001.

[30] B. Efron and R.J. Tibshirani, *An Introduction to the Bootstrap.* Chapman & Hall/CRC, 1993.

[31] S.N. Lahiri, *Resampling Methods for Dependent Data.* Springer, 2003.

[32] P. Hall, J.L. Horowitz, and B.-Y. Jing, "On Blocking Rules for the Bootstrap with Dependent Data," *Biometrika,* vol. 82, no. 3, pp. 561-574, 1995.

[33] P. Spellman, G. Sherlock, M. Zhang, V. Iyer, K. Anders, M. Eisen, P. Brown, D. Botstein, and B. Futcher, "Comprehensive Identification of Cell Cycle-Regulated Genes of the Yeast Saccharomyces Cerevisiae by Microarray Hybridization," *Molecular Biology of the Cell,* vol. 9, no. 12, pp. 3273-3297, 1998.

[34] R. Cho, M. Campbell, E. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T.G. Wolfsberg, A.E. Gabrielian, D. Landsman, D.J. Lockghart, and R.W. Davis, "A Genome-Wide Transcriptional Analysis of the Mitotic Cell Cycle," *Molecular Cell,* vol. 2, no. 1, pp. 65-73, 1998.

[35] D. Sahoo, D.L. Dill, R. Tibshirani, and S.K. Plevritis, "Extracting Binary Signals from Microarray Time-Course Data," *Nucleic Acids Research,* vol. 35, no. 11, pp. 3705-3712, Jan. 2007.

[36] J.A. Hartigan and M.A. Wong, "A K-Means Clustering Algorithm," *Applied Statistics,* vol. 28, pp. 100-108, 1979.

[37] I. Simon, J. Barnett, N. Hannett, C. Harbison, N. Rinaldi, T. Volkert, J. Wyrick, J. Zeitlinger, D. Gifford, T. Jaakkola, and R. Young, "Serial Regulation of Transcriptional Regulators in the Yeast Cell Cycle," *Cell,* vol. 106, no. 6, pp. 697-708, 2001.

[38] M. Teixeira, P. Monteiro, P. Jain, S. Teneiro, A.R. Fernandes, N.P. Mira, M. Alenquer, A.T. Freitas, A.L. Oliviera, and I. Sá-Correia, "The YEASTRACT Database: A Tool for the Analysis of Transcription Regulatory Associations in Saccharomyces Cerevisiae," *Nucleic Acids Research,* vol. 34, no. Suppl. 1, pp. D446-D451, 2006.

[39] V. Matys, E. Fricke, R. Geffers, E. Gössling, M. Haubrock, R. Hehl, K. Hornischer, D. Karas, A.E. Kel, O.V. Kel-Margoulis, D.U. Kloos, S. Land, B. Lewicki-Potapov, H. Michael, R. Münch, I. Reuter, S. Rotert, H. Saxel, M. Scheer, S. Thiele, and E. Wingender, "TRANSFAC: Transcriptional Regulation, from Patterns to Profiles," *Nucleic Acids Research,* vol. 31, no. 1, pp. 374-378, 2003.

[40] E.S. Dimitrova, M.P.V. Licona, J. McGee, and R. Laubenbacher, "Discretization of Time Series Data," *J. Computational Biology,* vol. 17, no. 6, pp. 853-868, Jan. 2010.

[41] K. Hakamada, T. Hanai, H. Honda, and T. Kobayashi, "A Preprocessing Method for Inferring Genetic Interaction from Gene Expression Data Using Boolean Algorithm," *J. Bioscience Bioeng.,* vol. 98, no. 6, pp. 457-63, Jan. 2004.

[42] O. Hirose, N. Nariai, Y. Tamada, and H. Bannai, "Estimating Gene Networks from Expression Data and Binding Location Data via Boolean Networks," *Proc. First Int'l Workshop Data Mining and Bioinformatics,* pp. 349-356, 2005.

[43] D. Pe'er, A. Regev, G. Elidan, and N. Friedman, "Inferring Subnetworks from Perturbed Expression Profiles," *Bioinformatics,* vol. 17, no. Suppl. 1, pp. S215-S224, Jan. 2001.

[44] B.D. Camillo, F. Sanchez-Cabo, G. Toffolo, S.K. Nair, Z. Trajanoski, and C. Cobelli, "A Quantization Method Based on Threshold Optimization for Microarray Short Time Series," *BMC Bioinformatics,* vol. 6(Suppl. 4):S11, Dec. 2005.
[45] M. Sezgin and B. Sankur, "Survey over Image Thresholding Techniques and Quantitative Performance Evaluation," *J. Electronic Imaging,* vol. 13, no. 1, pp. 146-165, 2004.
[46] C. Mircean, I. Tabus, and J. Astola, "Quantization and Distance Function Selection for Discrimination of Tumors Using Gene Expression Data," *Proc. SPIE,* vol. 4623, no. 1, pp. 1-12, Jan. 2002.
[47] D. Lockhart and E. Winzeler, "Genomics, Gene Expression and DNA Arrays," *Nature,* vol. 405, no. 6788, pp. 827-836, 2000.
[48] D. Allison, X. Cuia, G. Page, and M. Sabripour, "Microarray Data Analysis: From Disarray to Consolidation and Consensus," *Nature Rev. Genetics,* vol. 7, no. 1, pp. 55-66, 2006.
[49] E.S. Lander, "Array of Hope," *Nature Genetics,* vol. 21, no. 1, pp. 3-4, 1999.
[50] H.H. McAdams and A. Arkin, "It's a Noisy Business! Genetic Regulation at the Nanomolar Scale," *Trends in Genetics,* vol. 15, no. 2, pp. 65-69, 1999.
[51] I. Shmulevich, E.R. Dougherty, S. Kim, and W. Zhang, "Probabilistic Boolean Networks: A Rule-Based Uncertainty Model for Gene Regulatory Networks," *Bioinformatics,* vol. 18, no. 2, pp. 261-274, 2002.
[52] U. Braga-Neto, "Classification and Error Estimation for Discrete Data," *Current Genomics,* vol. 10, no. 7, pp. 446-462, Jan. 2009.

Martin Hopfensitz studied computer science at Ulm University and received the diploma degree in 2008. He is currently working toward the PhD degree in bioinformatics.

Christoph Müssel studied computer science at Ulm University. He received the diploma degree in 2008 and is currently working toward the PhD degree in bioinformatics in the same university.

Christian Wawra studied computer science at Ulm University. After his diploma thesis on pattern matching, he joined the International PhD Programme in molecular medicine at Ulm as a graduate student. During this time, he worked on the robustness of signaling network models as well as discrete reconstruction methods and received the PhD degree in 2009.

Markus Maucher studied computer science at Ulm University and received the PhD degree in 2009. He is currently working in the Bioinformatics and Systems Biology group at Ulm University.

Michael Kühl studied biochemistry at Free University of Berlin and received the PhD degree in 1995. After working in Ulm, Seattle, and Göttingen, he heads the Institute for Biochemistry and Molecular Biology. He is also head of the International Graduate School in molecular medicine at Ulm. His research interests include the analysis of intracellular signaling pathways and gene regulatory networks during vertebrate embryonic development.

Heiko Neumann studied computer science at the Technical University of Berlin and received a doctoral degree in computer science at the University of Hamburg in 1988. He received the Habilitation degree in 1995. Since 1995, he has been a full professor in the Department of Neural Information Processing at Ulm University. He spent several research sabbaticals at the Center for Adaptive Systems at Boston University. His research interests include neural modeling in computational and cognitive neuroscience, biologically inspired computational vision, and object recognition.

Hans A. Kestler studied electrical engineering at the Technical University of Munich and received a doctoral degree in computer science at Ulm University in 2002. Currently, he heads the Bioinformatics and Systems Biology group within the Faculties of Computer Science and Medicine of Ulm University. He has published more than 140 papers in journals, books, and conferences. His research interests include methodological foundations of pattern recognition, bioinformatics, and molecular systems biology. He is a member of the IEEE and the IEEE Computer Science.