Hindawi Publishing Corporation
EURASIP Journal on Bioinformatics and Systems Biology
Volume 2008, Article ID 482090, 11 pages
doi:10.1155/2008/482090

Research Article

Inference of Gene Regulatory Networks Based on a Universal Minimum Description Length

John Dougherty, Ioan Tabus, and Jaakko Astola

Institute of Signal Processing, Tampere University of Technology, P.O. Box 553, 33101 Tampere, Finland

Correspondence should be addressed to John Dougherty, john.dougherty@tut.fi

Received 24 August 2007; Accepted 11 January 2008

Recommended by Aniruddha Datta

The Boolean network paradigm is a simple and effective way to interpret genomic systems, but discovering the structure of these networks remains a difficult task. The minimum description length (MDL) principle has already been used for inferring genetic regulatory networks from time-series expression data and has proven useful for recovering the directed connections in Boolean networks. However, the existing method uses an ad hoc measure of description length that necessitates a tuning parameter for artificially balancing the model and error costs and, as a result, directly conflicts with the MDL principle's implied universality. In order to surpass this difficulty, we propose a novel MDL-based method in which the description length is a theoretical measure derived from a universal normalized maximum likelihood model. The search space is reduced by applying an implementable analogue of Kolmogorov's structure function. The performance of the proposed method is demonstrated on random synthetic networks, for which it is shown to improve upon previously published network inference algorithms with respect to both speed and accuracy. Finally, it is applied to time-series *Drosophila* gene expression measurements.

1. Introduction

The modeling of gene regulatory networks is a major focus of systems biology because, depending on the type of modeling, the networks can be used to model interdependencies between genes, to study the dynamics of the underlying genetic regulation, and to provide a basis for the derivation of optimal intervention strategies. In particular, Bayesian networks [1, 2] and dynamic Bayesian networks [3, 4] provide models to elucidate dependency relations; functional networks, such as Boolean networks [5] and probabilistic Boolean networks [6], provide the means to characterize steady-state behavior. All of these models are closely related [7].

When inferring a network from data, regardless of the type of network being considered, we are ultimately faced with the difficulty of finding the network configuration that best agrees with the data in question. Inference starts with some framework assumed to be sufficiently complex to capture a set of desired relations and sufficiently simple to be satisfactorily inferred from the data at hand. Many methods have been proposed, for instance, in the design of Bayesian networks [8] and probabilistic Boolean networks [9]. Here we are concerned with Boolean networks, for which a number of methods have been proposed [10–14]. Among the first information-based design algorithms is the Reveal algorithm, which utilizes mutual information to design Boolean networks from time-course data [11]. Information-theoretic design algorithms have also been proposed for non-time-course data [15, 16].

Here we take an information-theoretic approach based on the minimum description length (MDL) principle [17]. The MDL principle states that, given a set of data and class of models, one should choose the model providing the shortest encoding of the data. The coding amounts to storing both the network parameters and any deviations of the data from the model, a breakdown that strikes a balance between network precision and complexity. From the perspective of inference, the MDL principle represents a form of complexity regularization, where the intent is generally to measure the goodness of fit as a function of some error and some measure of complexity so as not to overfit the data, the latter being a critical issue when inferring gene networks from limited data. Basically, in addition to choosing an appropriate type, one wishes to select a model most suited for the amount of data. In essence, the MDL principle balances error (deviation from the data) and model complexity by using a cost function consisting of a sum of entropies, one relative to encoding the error and the other relative to encoding the model description [18]. The situation is analogous to that of structural risk minimization in pattern recognition, where the cost function for the classifier is a sum of the resubstitution error of the empirical-error-rule classifier and a function of the VC dimension of the model family [19]. The resubstitution error directly measures the deviation of the model from the data and the VC dimension term penalizes complex models. The difficulties are that one must determine a function of the VC dimension and that the VC dimension is often unknown, so that some approximation, say a bound, must be used. The MDL principle was among the first methods used for gene expression prediction using microarray data [20].

Recently, a time-course-data algorithm, henceforth referred to as Network MDL [10], was proposed based on the MDL principle. The Network MDL algorithm often yields good results, but it does so with an ad hoc coding scheme that requires a user-specified tuning parameter. We will avoid this drawback by achieving a codelength via a normalized maximum likelihood model. In addition, we will improve upon Network MDL's efficiency by applying an analogue of Kolmogorov's structure function [21].

2. Background

2.1. Boolean Networks

Using notation modified from Akutsu et al. [12], a Boolean network is a directed graph $G(V, \Lambda, F)$ defined by a set $V = \{v_i\}_{i=1}^{g}$ of g binary-valued nodes representing genes, a collection of structure parameters $\Lambda = \{\lambda_i\}_{i=1}^{g}$ indicating their regulatory sets (predecessor genes), and the Boolean functions $F = \{f_i\}_{i=1}^{g}$ regulating their behavior. Specifically, each structure parameter $\lambda_i = \{i_1, \ldots, i_{k_i}\}$ is the collection of indices $i_1 < i_2 < \cdots < i_{k_i}$ associated with v_i's regulatory nodes. The number k_i of regulatory nodes for node v_i is referred to as the indegree of v_i. We assume that the nodes are observed over $n+1$ equally spaced time points, and we write $y_{i,t} \in \mathcal{B} = \{0,1\}$ to denote the values of node i for $t = 0, 1, \ldots, n$. The value of node v_i progresses according to

$$y_{i,t} = f_i(y_{i_1,t-1}, y_{i_2,t-1}, \ldots, y_{i_{k_i},t-1}) \tag{1}$$

for $t = 1, \ldots, n$. Such synchronous updating is perhaps unrealistic in biological systems, but it provides a framework with more easily tractable models and has proven useful in the present context [22]. For ease of notation, we define the inputs of f_i as the column vector $\mathbf{x}_{i,t} = [y_{i_1,t-1}, y_{i_2,t-1}, \ldots, y_{i_{k_i},t-1}]'$, allowing us to rewrite (1) as

$$y_{i,t} = f_i(\mathbf{x}_{i,t}), \quad t = 1, \ldots, n. \tag{2}$$

The fundamental question we face is the estimation of Λ and F. Note that Λ is usually not included as a parameter of G because it can be absorbed into F, but we choose to write it separately because, under the model we will specify, Λ completely dictates F, making our interest reside primarily in the structure parameter set Λ.

As written, (2) provides us with a completely deterministic network, but this is generally considered to be an inadequate description. Measurement error is inescapable in virtually any experimental setting, and, even if one could obtain noiseless data, biological systems are constantly under the influence of external factors that might not even be identifiable, let alone measurable [6]. Therefore, we consider it incumbent to relocate our model of the network mechanisms into a probabilistic framework. By incorporating this philosophy and switching to matrix notation, (2) becomes

$$\mathbf{Y}_i = f_i(\mathbf{X}_i) \oplus \boldsymbol{\varepsilon}_i \in \mathcal{B}^n, \tag{3}$$

where $\oplus$ denotes modulo 2 sum, f_i acts independently on each column of $\mathbf{X}_i = [\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,n}]$, and $\boldsymbol{\varepsilon}_i$ is a vector of independent Bernoulli random variables with $P(\varepsilon_{i,t} = 1) = \theta_i \in [0,1]$. We further assume that the errors for different nodes are independent. We allow θ_i to depend on i because it can be interpreted as the probability that node i disobeys the network rules, and we consider it natural for different nodes to have varying propensities for misbehaving.

Returning to our overall objective, we observe that λ_i and f_i can be estimated separately for each gene. This is possible because, for each evaluation of f_i, $\mathbf{X}_i$ is regarded as fixed and known. Even if a network was constructed so that a gene was entirely self-regulatory, that is, $\lambda_i = \{i\}$, the random vector $\mathbf{Y}_i$ is observed sequentially so that any random variable $Y_{i,t}$ within it is observed and then considered as a fixed value $x_{i,t+1}$ before being used to obtain $Y_{i,t+1}$. Therefore, despite the obvious dependencies that would exist for networks containing configurations such as feedback loops and nodes appearing in multiple predecessor sets, the given model stipulates independence between all random variables. Thus, we restrict ourselves to estimating the parameters for one node and rewrite (3) as

$$\mathbf{Y} = f(\mathbf{X}) \oplus \varepsilon, \tag{4}$$

which we recognize as multivariate Boolean regression. Note that θ_i and k_i now become θ and k, respectively.

We finalize the specification of our model by extending the parameter space for the error rates by replacing θ with $\Theta = \{\theta_l\}_{l=0}^{2^k-1}$, where each θ_l corresponds to one of the 2^k possible values of $\mathbf{x}_t$. This allows the degree of reliability of the network function to vary based upon the state of a gene's predecessors. Note that 2^k is only an upper bound on the number of error rates because we will not necessarily observe all 2^k possible regressor values. This model is specified by the predecessor genes composing $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$, the function f, and the error rates in Θ. Thus, adopting notation from Tabus et al. [23], we refer to the collection of all possible parameter settings as the model class $\mathcal{M}(\Theta, \lambda, f)$.

2.2. The MDL Principle

Given the model formulation, we use the MDL principle as our metric for assessing the quality of the parameter

Table 1: Probability table for "OR" function with $\theta = 0.2$.

(x_1, x_2)	$P(Y = 0)$	$P(Y = 1)$
$(0, 0)$	0.8	0.2
$(0, 1)$	0.2	0.8
$(1, 0)$	0.2	0.8
$(1, 1)$	0.2	0.8

estimates. As stated in Section 1, the MDL principle dictates that, given a dataset and some class of possible models, one should choose the model providing the shortest possible encoding of the data. In our case, the MDL principle is applied for selecting each node's predecessors. However, as we have noted, this technique is inherently problematic because no unique manner of codelength evaluation is specified by the principle. Letting $e_t = 1$ when the node in question is predicted incorrectly and 0 otherwise, basic coding theory gives us a residual codelength of $-\sum_{t=1}^{n} \log_2 P(\varepsilon_t = e_t)$, but the cost of storing the model parameters has no such standard. Thus, we can technically choose any applicable encoding scheme we like, an allowance that inevitably gives rise to infinitely many model codelengths and, as a result, no unique MDL-based solution.

As an example, we refer to the encoding method used in Network MDL, in which the network is stored via probability tables such as Table 1. In this procedure, the model codelength is calculated as the cost of specifying the two predecessor genes plus the cost of storing the probability table. Letting d_i and d_f denote the number of bits needed to encode integers and subunitary floating point numbers, respectively, the model codelength is $2d_i + 4d_f$. Note that we only need 4 of the probabilities since each row in the table adds to 1. This is one of many perfectly reasonable coding schemes, but we present another method that corresponds to our model class and yields a shorter codelength. Also, to demonstrate the risk of using the MDL principle with ad hoc encodings, we compare results obtained by using these two schemes in a short artificial example. Observe that Table 1 corresponds to $\mathcal{M}(\Theta, \lambda, f)$ with each $\theta_l = 0.2$. First, we encode f as the 4 bits 0111 because, providing all predecessor combinations are lexographically sorted, those are the values that Y will be with probability $1 - \theta$. Assuming we select f to minimize the error rates, we can also assume that $\theta_l \in [0, 0.5]$. Since d_f bits are sufficient to encode any decimal less than 1, we really only need $d_f/2$ bits to store each θ_l, yielding a model cost of $2d_i + 2d_f + 4$.

To show the effect of the encoding scheme we generated one hundred 6-gene networks, each of which was observed over 50 time points. Λ and F were fixed so that one gene would behave according to Table 1. The MDL principle was applied for both of the encoding schemes to determine the predecessors of that gene. The results are displayed in Table 2.

We find that the two encoding methods can give different structure estimates because the shorter model codelength allows for a greater number of predecessors. Zhao et al. compensate for this nonuniqueness by adjusting the model codelength with a weight parameter, but, while necessary for ad hoc encodings such as the ones discussed so far, the presence of such tuning parameters is undesirable when compared with a more theoretically based method. Moreover, the MDL principle's notion of "the shortest possible codelength" implies a degree of generality that is violated if we rely upon a user-defined value.

Table 2: Effect of ad hoc encoding schemes on structure inference. Results are reported as percentages. "Fair" and "Poor" indicate missing one and both of the two predecessors, respectively.

	Encoding method	
Model performance	Network MDL	$\mathcal{M}(\Theta, \lambda, f)$
Correct	0.03	0.08
Fair	0.12	0.17
Poor	0.85	0.75

2.3. Normalized Maximum Likelihood

One alternative that alleviates these drawbacks is to measure codelength based on universal models. In this approach, we depart from two part description lengths and their ad hoc parameters by evaluating costs using a framework that incorporates distributions over the entire model class. The fundamental idea for such a model is that, assuming a specific model class, we should choose parameters that maximize the probability of the data [21]. Two such models are the mixture universal model and the normalized maximum likelihood (NML) model, the latter of which will command our attention. For $\mathcal{M}(\Theta, \lambda, f)$ with a fixed λ, the NML model is introduced by the standard likelihood optimization problem $\max_{\Theta} \log P(\mathbf{y}; \Theta, \lambda, f)$. The solution is obtained for $\Theta = \widehat{\Theta}$, the maximum likelihood estimate (MLE), but cannot be used as a model because $P(\mathbf{y}; \widehat{\Theta}, \lambda, f)$ does not integrate to unity. Thus, we will use the distribution $q(\mathbf{y})$ such that its ideal codelength $-\log_2 q(\mathbf{y})$ is as close as possible to the codelength $-\log_2 P(\mathbf{y}; \widehat{\Theta}, \lambda, f)$. This suggests that we should minimize the difference between using $q(\mathbf{y})$ in place of $P(\mathbf{y}; \widehat{\Theta}, \lambda, f)$ for the worst case $\mathbf{y}$. The resulting optimization problem,

$$\min_{q} \max_{\mathbf{y}} \log_2 \frac{P(\mathbf{y}; \widehat{\Theta}, \lambda, f)}{q(\mathbf{y})}, \tag{5}$$

is solved by the NML density function, defined as $P(\mathbf{y}; \widehat{\Theta}, \lambda, f)$ divided by the normalizing constant $\sum_{\mathbf{y} \in \mathcal{B}^n} P(\mathbf{y}; \widehat{\Theta}, \lambda, f)$. Tabus et al. [23] provide the derivations of this NML distribution; the following is a brief outline of the major steps.

Given a realization $\mathbf{y}$ of the random variable $\mathbf{Y}$, we have residuals

$$\mathbf{e} = \mathbf{y} \oplus f(\mathbf{X}). \tag{6}$$

Recall that the Bernoulli distribution is defined by

$$P(\varepsilon = e) = \theta^{e} (1 - \theta)^{1-e}, \quad e \in \mathcal{B}. \tag{7}$$

Letting $\mathbf{b}_l$ denote the k-bit binary representation of integer l, combine (6) and (7) to define the probability $P(y_t; f, \mathbf{b}_l, \theta_l)$ as

$$P(Y_t = y_t\,;\, \mathbf{x}_t = \mathbf{b}_l) = \theta_l^{y_t \oplus f(\mathbf{b}_l)} (1-\theta_l)^{1-y_t\oplus f(\mathbf{b}_l)}. \quad (8)$$

This representation allows us to formally write our model class as

$$\mathcal{M}(\Theta,\lambda,f) = \{P(y_t; f, \mathbf{b}_l, \theta_l) = \theta_l^{y_t \oplus f(\mathbf{b}_l)} (1-\theta_l)^{1-y_t\oplus f(\mathbf{b}_l)}\}. \quad (9)$$

2.3.1. NML Model for $\mathcal{M}(\Theta,\lambda,f)$

Consider any $\mathbf{y} \in \mathcal{B}^n$ and fixed λ. Let m_l denote the number of times each unique regressor vector $\mathbf{b}_l \in \mathcal{B}^k$ occurs in $\mathbf{X}$, and let m_{l_1} count the number of times $\mathbf{b}_l$ is associated with a unitary response. As pointed out by Tabus et al. [23], the MLE for this model is not unique. The network could have $f(\mathbf{b}_l) = 0$, in which case $\hat{\theta}_l = m_{l_1}/m_l$, or $f(\mathbf{b}_l) = 1$, giving $\hat{\theta}_l = 1 - m_{l_1}/m_l$. Either way, the NML model is given by

$$\hat{P}(\mathbf{y}) = \frac{P(\mathbf{y};\lambda,\hat{f},\mathbf{X},\hat{\Theta})}{\prod_{l:\mathbf{b}_l\in\mathbf{X}} C_{m_l}}, \quad (10)$$

where

$$P(\mathbf{y};\lambda,\hat{f},\mathbf{X},\hat{\Theta}) = \prod_{l:\mathbf{b}_l\in\mathbf{X}} \left(\frac{m_{l_1}}{m_l}\right)^{m_{l_1}} \left(1-\frac{m_{l_1}}{m_l}\right)^{m_l - m_{l_1}}, \quad (11)$$

$$C_{m_l} = \sum_{i=0}^{m_l} \binom{m_l}{i} \left(\frac{i}{m_l}\right)^{i} \left(1-\frac{i}{m_l}\right)^{m_l - i}. \quad (12)$$

Of course, this means that our model does not explicitly estimate f. However, considering that Θ represents error rates, the obvious choice is to minimize each $\hat{\theta}_l$ by taking $\hat{f}(\mathbf{b}_l) = 0$ whenever $m_{l_1} < m_l - m_{l_1}$, and 1 otherwise. In the event that $\hat{\theta}_l = 1/2$, we set $\hat{f}(\mathbf{b}_l) = 0$ if the portion of $\mathbf{y}$ corresponding to $\mathbf{b}_l$ is less than $m_l/2$ in binary. Assuming independent errors, this removes any bias that would result from favoring a particular value for $\hat{f}(\mathbf{b}_l)$ when $\hat{\theta}_l = 1/2$. This effectively reduces the parameter space for each θ_l from $[0,1]$ to $[0,1/2]$ which, in turn, affects $\hat{P}(\mathbf{y})$ by halving every C_{m_l}. However, we will later show that the algorithm does not change whether or not we actually specify $\hat{f}$, and we opt not to do so.

Also note that computing C_{m_l} exactly may not be feasible. For example, Matlab loses precision for the binomial coefficient $\binom{m_l}{i}$ when $m_l > 53$. In these cases, we use

$$C_{m_l} \approx \sqrt{\frac{\pi m_l}{2}} + \frac{2}{3} + \left(\frac{1}{24}\right)\sqrt{\frac{2\pi}{m_l}}, \quad (13)$$

an approximation given in [24]. For the sake of efficiency, we compute every C_{m_l} prior to learning the network so that calculating the denominator of (10) takes at most $\min(n, 2^k)$ operations.

2.3.2. Stochastic Complexity

We take as the measure of a selected model's total codelength the stochastic complexity of the data, which is defined as the negative base 2 logarithm of the NML density function [21]. As was already the case for the residual codelength, the stochastic complexity is a theoretical codelength and will not necessarily be obtainable in practice, but it is precisely this theoretical basis that frees us from any tuning parameters. Given (10), our stochastic complexity is given by

$$-\log \hat{P}(\mathbf{y}) = \sum_{l:\mathbf{b}_l\in\mathbf{X}} \left[m_l h\left(\frac{m_{l_1}}{m_l}\right) + \log C_{m_l}\right], \quad (14)$$

where $h(\cdot)$ denotes the binary entropy function. Note that the previous and all future logarithms are base 2. Returning to the issue of picking values for $\hat{f}$, we recall that doing so halves each C_{m_l}. This translates to a unit reduction in stochastic complexity for each $\mathbf{b}_l$, but we observe that it also requires 1 bit to store $\hat{f}(\mathbf{b}_l)$. Regardless of whether or not we choose to specify $\hat{f}$, the total codelength remains the same.

The NML model assumes a fixed λ to specify the set of predecessor genes, so encoding the network requires that we store this structure parameter as well. The simplest ways to accomplish this are by using g (the total number of genes) bits as indicators or by using $\log g$ bits to represent the number of predecessors (assuming a uniform prior on k) and $\log\binom{g}{k}$ bits to select one of the $\binom{g}{k}$ possible sets of size k. However, the indegrees of genetic networks are generally assumed to be small [25], in light of which we prefer a codelength that favors smaller indegrees and choose to use an upper bound on encoding the integer $k \le g$ to store k with $\log(k+1) + \log(1+\ln g)$ bits [21]. Note that we use $k+1$ because the given bound only applies for positive integers, and we must accommodate any $k \ge 0$. Hence, the total codelength is

$$L_T(\mathbf{y},\lambda) = -\log \hat{P}(\mathbf{y}) + L_\lambda, \quad (15)$$

where

$$L_\lambda = \min\left\{g, \log\binom{g}{k} + \log(k+1) + \log(1+\ln g)\right\}. \quad (16)$$

2.4. Kolmogorov's Structure Function

If we compute $L_T(\mathbf{y},\lambda)$ for every possible λ, we can simply select the one that provides the shortest total codelength, thus satisfying the MDL principle; however, this requires computing $\sum_{i=0}^{g}\binom{g}{i} = 2^g$ codelengths. A standard remedy for this problem is assuming a maximum indegree K [12], but, even with $K = 3$, a 20-gene network would still result in 1351 possible predecessor sets per gene. Moreover, a fixed K introduces bias into the method so, while we obviously cannot afford to perform exhaustive searches, we prefer to refrain from limiting the number of predecessors considered.

Instead, we utilize Kolmogorov's structure function (SF) to avoid excessive computations without sacrificing the

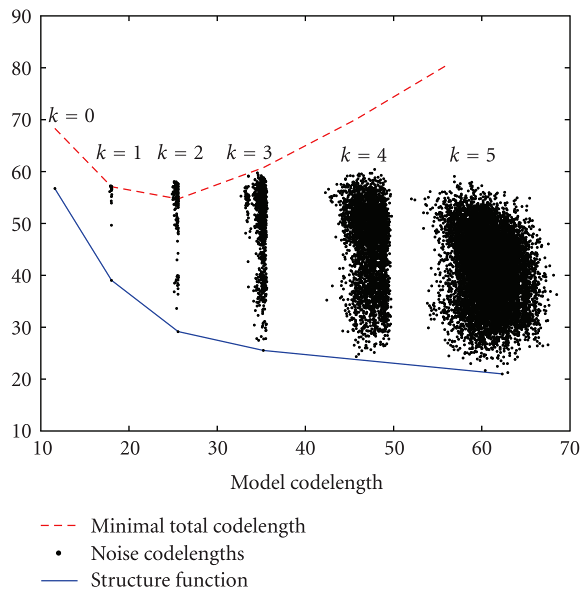


FIGURE 1: The SF for a single gene. The leftmost point is for $k = 0$, and each subsequent vertical band corresponds to a unit increase in k. The slope of the SF goes above -1 after $k = 2$, the same indegree for which the total codelength $L_M(\mathbf{y}, \lambda, d) + L_N(\mathbf{y}, \lambda, d)$ is minimized.

ability to identify predecessor sets of arbitrary size. The SF was originally developed within the algorithmic theory of complexity and is noncomputable, so, in order to use this theory for statistical modeling, we need a computable alternative. The details are beyond the scope of this paper, but obtaining a computable SF requires, for fixed λ, partitioning the parameter space for Θ so that the Kullback-Leibler distance between any two adjacent partitions, each of which represents a different model, is d/n for some d [21]. When using an NML model class, this partitioning yields an asymptotically uniform prior so that any model $P(\mathbf{y}; \lambda, f, \mathbf{X}, \Theta)$ can be encoded with length

$$L_M(\mathbf{y}, \lambda, d) = \sum_{l:\mathbf{b}_l \in \mathbf{X}} \log C_{m_l} + \frac{w}{2} \log \frac{w\pi}{2d} + L_\lambda, \qquad (17)$$

where $w \leq 2^k$ is the number of error estimates in $\hat{\Theta}$ [21]. Again, the inequality is necessary for data in which not all possible regressor vectors are observed. The partitioning also increases the noise codelength [21] to

$$L_N(\mathbf{y}, \lambda, d) = -\log P(\mathbf{y}; \lambda, \hat{f}, \mathbf{X}, \hat{\Theta}) + \frac{d}{2}. \qquad (18)$$

We refer to L_M and L_N as the model and noise codelengths, respectively, which together constitute a universal sufficient statistics decomposition of the total codelength. The summation of these values is clearly different from the stochastic complexity, but this is a result of partitioning the parameter space.

The appropriate analogue of the SF is then defined as

$$h_{\mathbf{y}}(\alpha) = \min_{\lambda, d} \{L_N(\mathbf{y}, \lambda, d) : L_M(\mathbf{y}, \lambda, d) \leq \alpha\}. \qquad (19)$$

We see that $h_{\mathbf{y}}(\alpha)$ is a nonincreasing function of the model constraint α and displays the minimum possible amount of noise in the data if we restrict the model codelength to be less than α. Rissanen shows that this criterion is minimized for $d = w$ [21], but the optimal λ cannot be solved analytically. However, by plotting $h_{\mathbf{y}}(\alpha)$ we obtain a graph similar to a rate-distortion curve (Figure 1), and by making a convex hull we can find a near-optimal predecessor set. Simply select the truncation point at which the magnitude of the slope of the hull drops below 1. In other words, locate the truncation point at which allowing an additional bit for the model yields less than a 1-bit reduction in the noise codelength because, once past this point, increasing the model complexity no longer decreases the total encoding cost.

Of particular use in this scenario is the way in which the model codelength is somewhat stable for each k, producing the distinct bands in Figure 1. The noise codelengths are still widely dispersed so we are required to compute all possible codelengths up to some total number of predecessors. We would like that number to be variable and not arbitrarily specified in advance, but this may not be feasible for highly connected networks. However, as mentioned earlier, the indegrees of genetic networks are generally assumed to be small (hence, the standard $K = 3$), and, when looking for a single gene's predecessors in a 20-gene network, our method only takes 70 minutes to check every possible set up to size 6. Thus, we are still constrained by a maximum indegree, but we can now increase it well beyond the accepted number that we expect to encounter in practice without risking extreme computational repercussions. Additionally, choosing a $K \leq g/2$ makes L_λ a nondecreasing function of k, meaning that we can also stop searching if L_λ ever becomes larger than the current value of $L_M(\mathbf{y}, \hat{\lambda}, d) + L_N(\mathbf{y}, \hat{\lambda}, d)$. The method is summarized in Algorithm 1.

Note that we termed the resulting predecessors "near-optimal." It is possible to encounter genes for which adding one predecessor does not warrant an increase in model codelength but adding two predecessors does. Nevertheless, these differences tend to be small for certain types of networks. Moreover, depending on the kind of error with which one is concerned, these near-optimal predecessor sets can even provide a better approximation of the true network in the sense that any differences will be in the direction of the SF finding fewer predecessors. Thus, assuming a maximum indegree K, the false positive rate from using the SF can never be higher than that from checking all predecessor sets up to size K.

3. Results

3.1. Performance on Simulated Data

A critical issue in performance analysis concerns the class from which the random networks are to be generated. While it might first appear that one should generate networks using the class $\mathcal{G}_g$ composed of all Boolean networks containing g genes, this is not necessarily the case if one wishes to achieve simulated results that reflect algorithm performance

```
(1) Initialize $\hat{\lambda} \Leftarrow \varnothing$
(2) $L_N(\hat{\lambda}) \Leftarrow nh(sum(\mathbf{y})/n) + 1/2$
(3) $L_M(\hat{\lambda}) \Leftarrow \log C_n + (1/2)\log(\pi/2) + \log(1 + \ln g)$
(4) for $k = 1$ to $K$ do
(5)     compute $L_\lambda$ using (16)
(6)     if $L_\lambda > L_M(\hat{\lambda}) + L_N(\hat{\lambda})$ then
(7)         return $\hat{\lambda}$
(8)     end if
(9)     $H \Leftarrow$ collection of all $\lambda$'s such that $|\lambda| = k$
(10)    for $i = 1$ to $|H|$ do
(11)        $\mathbf{X}_i \Leftarrow$ rows of $\mathbf{X}$ specified by $H_i$
(12)        for $l = 1$ to $2^k$ do
(13)            compute $m_l$ and $m_{l_1}$ for $\mathbf{X}_i$
(14)        end for
(15)        $w, d \Leftarrow$ number of nonzero $m_l$'s
(16)        compute $L_N(H_i)$ and $L_M(H_i)$
              using (11), (17), and (18)
(17)    end for
(18)    use $\mathbf{L}_N$, $\mathbf{L}_M$, $L_N(\hat{\lambda})$, and $L_M(\hat{\lambda})$ to form a convex
        hull with truncation points $\{(tpM_j, tpN_j)\}$
(19)    $idx \Leftarrow \max_j\{(j : tpN_j - tpN_{j-1})/$
                        $(tpM_j - tpM_{j-1}) < -1\}$
(20)    if isempty $(idx)$ then
(21)        return $\hat{\lambda}$
(22)    else
(23)        update $\hat{\lambda}$, $L_N(\hat{\lambda})$, and $L_M(\hat{\lambda})$ using truncation
            point indexed by $idx$
(24)    end if
(25) end for
```

Algorithm 1: The NML MDL method for one gene.

on realistic networks. An obvious constraint is to limit the indegree, either for biological reasons [26] or for the sake of inference accuracy when data are limited. In this case, one can consider the class $\mathcal{G}_g^\kappa$ composed of all Boolean networks with indegrees bounded by κ. Other constraints might include realistic attractor structures [27], networks that are neither too sensitive nor too insensitive to perturbations [28], or networks that are neither too chaotic nor too ordered [29].

Here we consider a constraint on the functions that is known to prevent chaotic behavior [5, 26]. A canalizing function is one for which there exists a gene among its regulatory set such that if the gene takes on a certain value, then that value determines the value of the function irrespective of the values of the other regulatory genes. For example, $f(x_1, x_2, x_3) = (x_1 \text{ and } x_3)$ OR x_3 is canalizing with respect to x_3 because $f(x_1, x_2, 1) = 1$ for any values of x_1 and x_2. There is evidence that genetic networks under the Boolean model favor this kind of functionality [30]. Corresponding to class $\mathcal{G}_g^\kappa$ is class $\mathcal{C}_g^\kappa$, in which all functions are constrained to be canalizing.

To evaluate the performance of our model selection method, referred to as NML MDL, on synthetic Boolean networks, we consider sample sizes ranging from 20 to 100, $\theta \in \{0.1, 0.2, 0.3\}$, and $\kappa \in \{1, 2, 3, 4\}$. We test each of the (n, θ, κ) combinations on 30 randomly generated networks from $\mathcal{G}_{20}^\kappa$ and $\mathcal{C}_{20}^\kappa$. Note that $\mathcal{G}_{20}^1$ is equivalent to $\mathcal{C}_{20}^1$.

We use the Reveal and Network MDL methods as benchmarks for comparison. As mentioned earlier, Network MDL requires a tuning parameter, which we set to 0.3 since that paper uses 0.2–0.4 as the range for this parameter in its simulations. Also, its application in [10] limits the average indegree of the inferred network to 3 so we assume this as well. Reveal is run from a Matlab toolbox created by Kevin Murphy, available for download at http://bnt.sourceforge.net/, and requires a fixed K, which we also set to 3. We implement our method with and without including the SF approach to show that the difference in accuracy is often small, especially in light of the reduction in computation time.

As performance metrics, we use the number of false positives and the Hamming distance between the estimated and true networks, both normalized over the total number of edges in the true network. False positives are defined as any time a proposed network includes an edge not existing in the real network, and Hamming distance is defined as the number of false positives plus the number of edges in the true network not included in the estimated network.

3.1.1. Random Networks

In this section, we consider performance when the network is generated from $\mathcal{G}_{20}^\kappa$. Figures 2–5 show a selection of the performance-metric results for all four methods and several combinations of κ and θ. The remaining figures can be found in the supporting data, available at http://www.stat.tamu.edu/~jdougherty/nmlmdl.

With respect to false positives, NML MDL is uniformly the best, and there is at most a minor difference between the two modes. NML MDL is also the best overall method when looking at Hamming distances. Figures 2 and 3 show the cases for which it most definitively improves upon Network MDL and Reveal, both of which have $\theta = 0.1$. The way in which the two NML methods diverge as κ increases is a general trend, but both remain below Network MDL. Increasing θ to 0.2 narrows the margins between the methods, but the relationships only change significantly for $\kappa = 4$. As shown in Figure 4, NML MDL with the SF loses its edge, but NML MDL with fixed K remains the best choice. Raising θ to 0.3 is most detrimental to Reveal, pulling its accuracy well away from the other three methods. Figure 5 shows this for $\kappa = 4$, but the plots for smaller values of κ look very similar, especially in how the two NML MDL approaches perform almost identically. We point out that this is the worst scenario for NML MDL, but, even then, it is still superior for small n and only worse than Network MDL for $n = 80$.

In terms of computation time, Reveal was fairly constant for all of the simulation settings, taking an average of 6.35 seconds to find predecessors for gene using Matlab on a Pentium IV desktop computer with 1 GB of memory. NML MDL with $K = 3$ increases slightly with n in a linear fashion, but its most noticeable increase is with κ. For $\kappa = 1$, this method took an average of 0.33 to 0.48 seconds per gene as

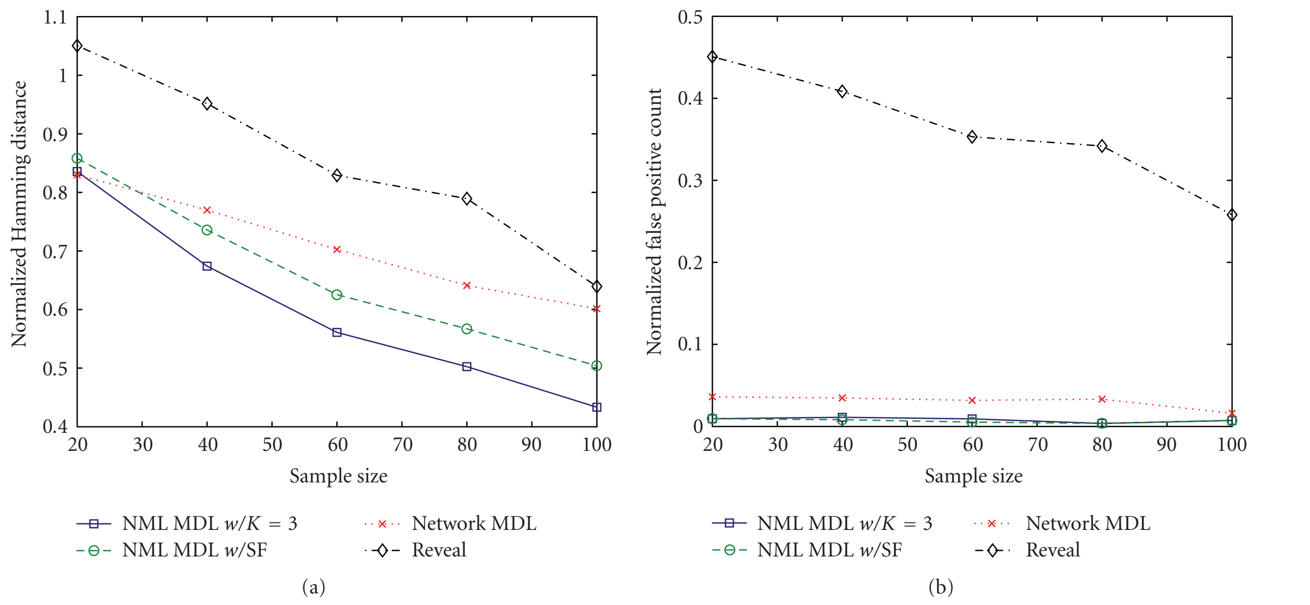


FIGURE 2: (a) Hamming distances and (b) false positive counts for random networks generated from $\mathcal{G}^3_{20}$ with $\theta = 0.1$. Results are normalized over the true number of connections and averaged over 30 networks.

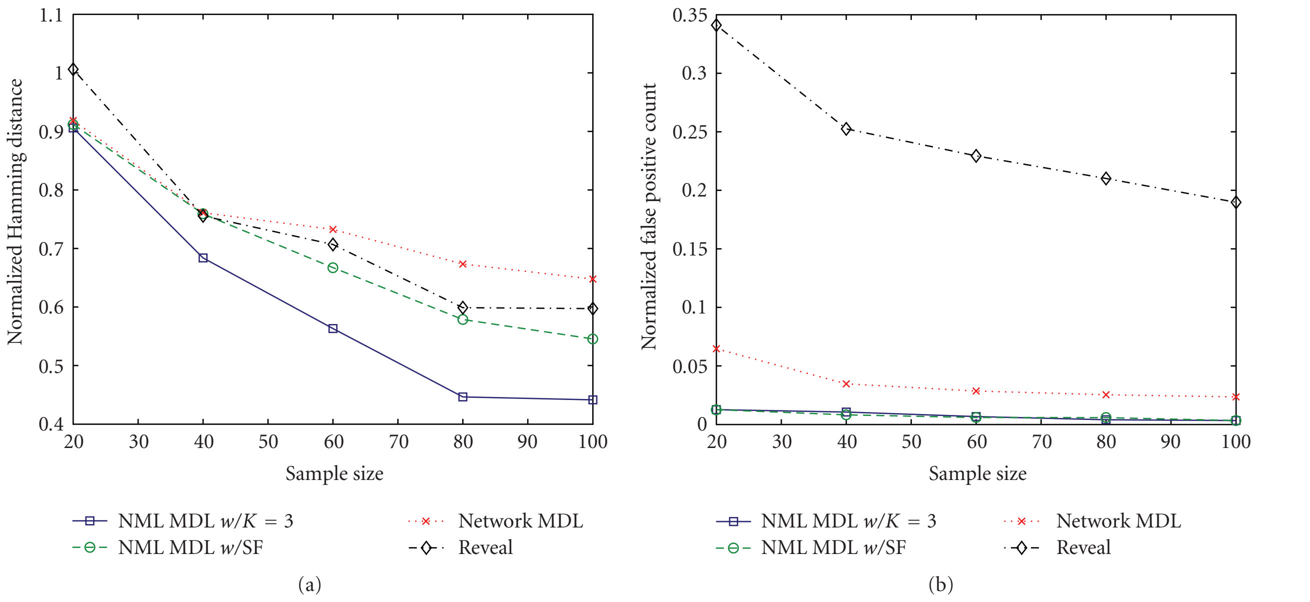
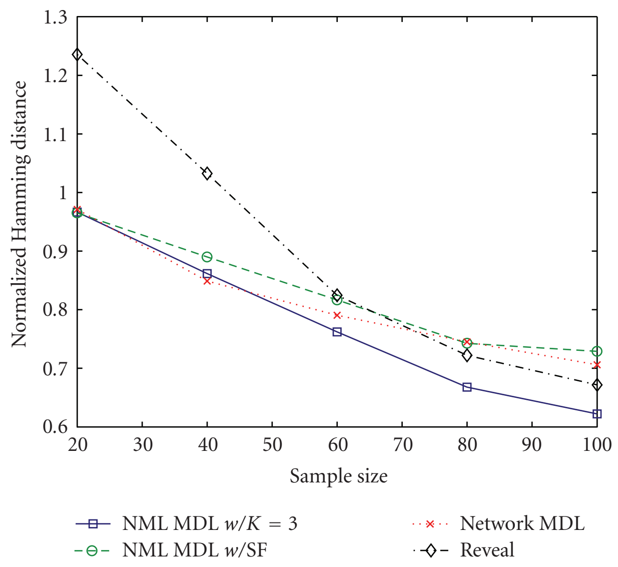


FIGURE 3: Error rates for $\mathcal{G}^4_{20}$ and $\theta = 0.1$.

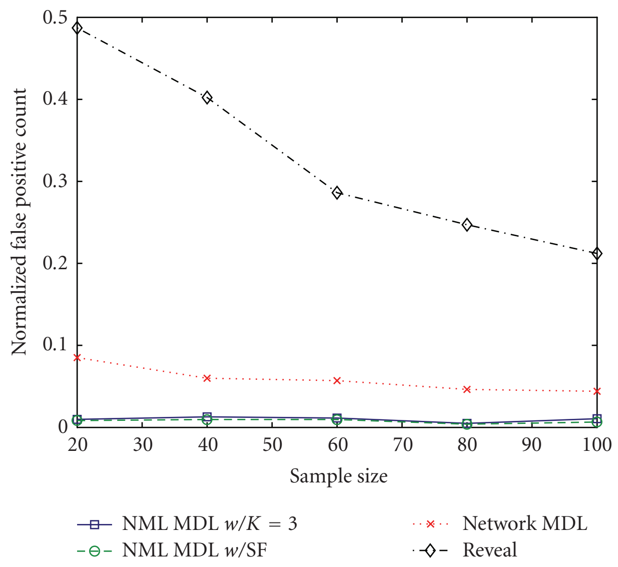
n goes from 20 to 100, but this range increased from 0.59 to 0.73 for $\kappa = 4$. Alternatively, Network MDL's runtime is sporadic with respect to n and decreases when κ is raised, taking an average of 2.50 seconds per gene for $\kappa = 1$ but needing only 0.33 second per gene when $\kappa = 4$, the only case for which it was noticeably faster than NML MDL with fixed K. However, NML MDL with the SF proved to be the most efficient algorithm in almost every scenario. For $\theta = 0.2$ and 0.3 it was uniformly the fastest, taking an average of 0.06 and 0.02 seconds per gene, respectively. The runtime begins to increase more rapidly with n for $\theta = 0.1$ and $\kappa \geq 3$, but the only observed case when it was not the fastest method was for $n = 100$ and $\kappa = 4$, and even then the needed time was still less than 1 second per gene.

3.1.2. Canalizing Networks

Next, we impose the canalizing restriction and generate networks from $\mathcal{C}^{\kappa}_{20}$. The general impact can be seen by comparing Figures 3 and 6. There is essentially no difference

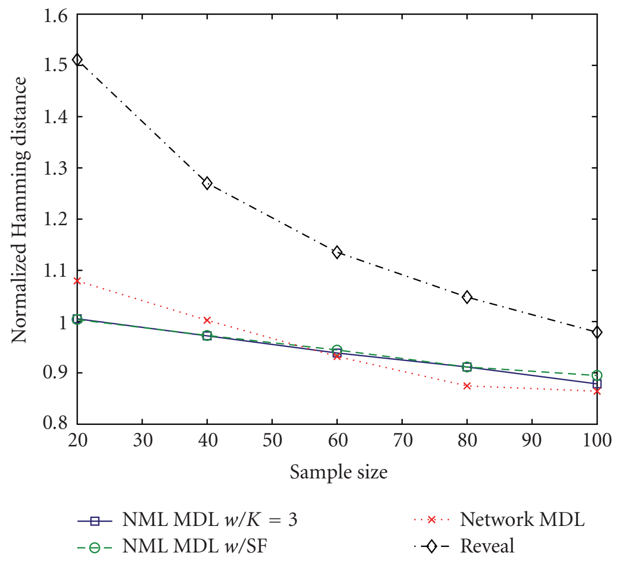


(a)

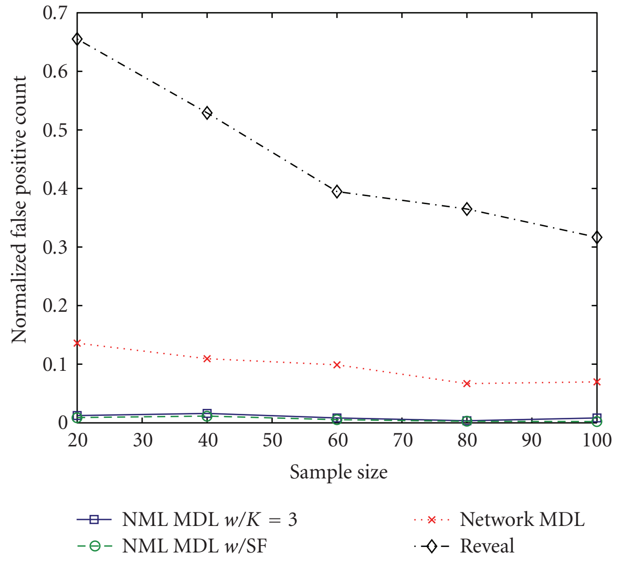


(b)

FIGURE 4: Error rates for $\mathcal{G}_{20}^{4}$ and $\theta = 0.2$.



(a)



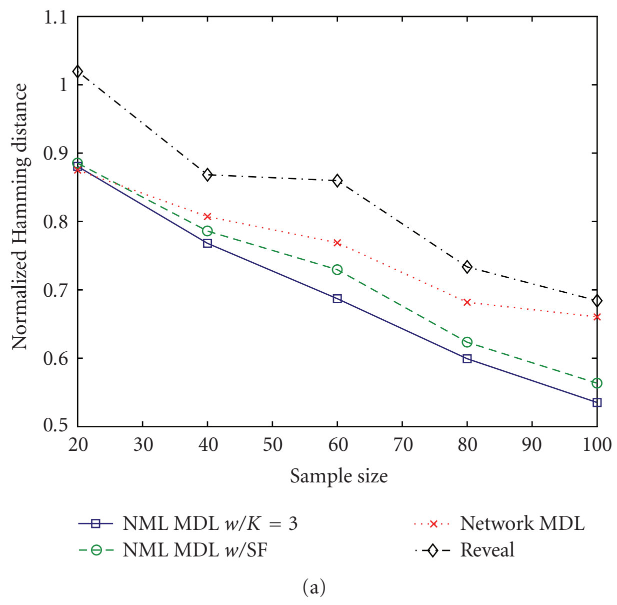
(b)

FIGURE 5: Error rates for $\mathcal{G}_{20}^{4}$ and $\theta = 0.3$.

in the false positive rates (or runtimes), but the behavior of the Hamming distances is clearly different. We observe that NML MDL with fixed K performs better over all Boolean functions, although invoking the SF yields error rates much closer to the fixed K approach when we are restricted to canalizing functions. This is expected because one canalizing gene can provide a significant amount of predictive power, whereas a noncanalizing function may require multiple predecessors to achieve any amount of predictability.

For example, consider $f(x_1, x_2) = x_1 \text{ OR } x_2$. If x_1 is found to be the best predecessor set of size 1, adding x_2 may not give enough additional information to warrant the increased model codelength, in which case NML MDL will miss one connection. Alternatively, if $f(x_1, x_2) = x_1 \text{ XOR } x_2$, either input tells almost nothing by itself, and the SF will probably stop the inference too soon. However, using both inputs will most likely result in the minimum total codelength, in which case NML MDL with fixed K will find the correct predecessor set.

For the same reason, we also see that Network MDL is better suited to canalizing functions, but Reveal does better without this constraint. Of particular interest is that,



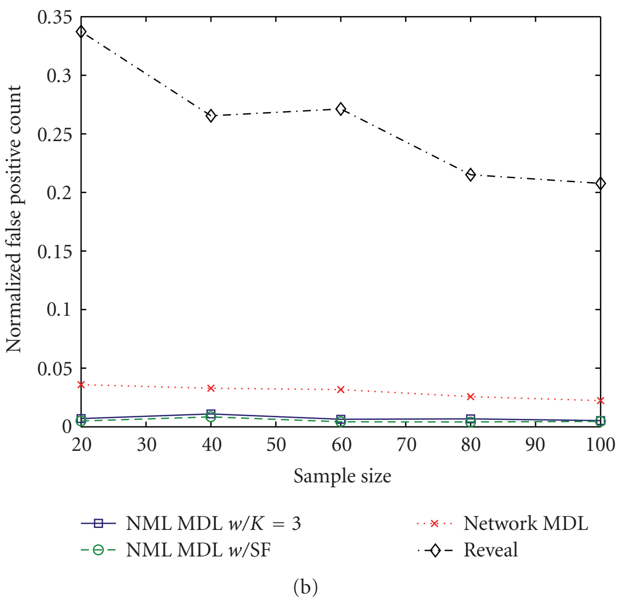


Figure 6: Error rates for $\mathfrak{C}_{20}^{4}$ and $\theta = 0.1$.

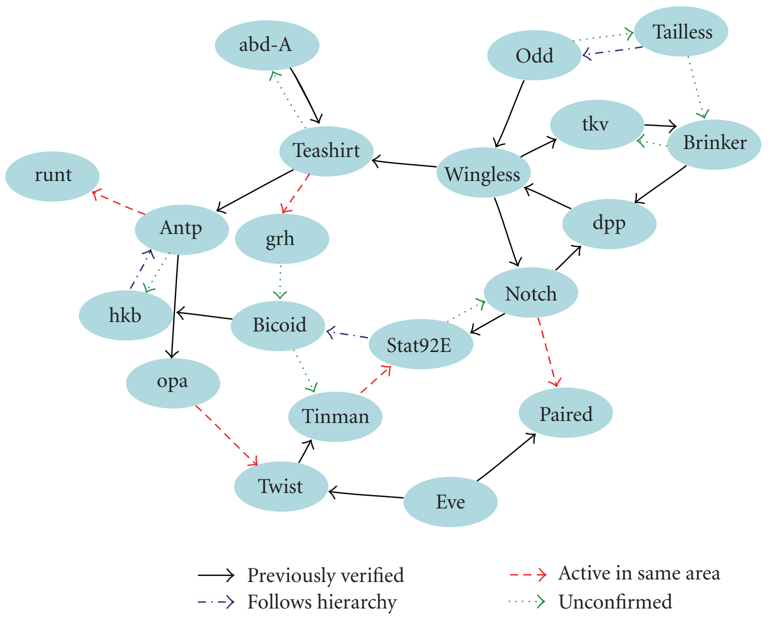


Figure 7: Inferred gene regulatory network for *Drosophila*.

for these methods, the change can be so drastic that they comparatively switch their rankings depending on which network class we use, whereas NML MDL provides the most accurate inference either way. Similar results can be observed for the other cases in the supporting data. Based on these findings, we recommend using the SF primarily for networks composed of canalizing functions and networks too large to run NML MDL with fixed K in a reasonable amount of time. We also suggest using the SF when θ is large because, as pointed out in Section 3.1.1, the performance of the two NML MDL varieties is no longer different when $\theta = 0.3$.

3.2. Application to Drosophila Data

In order to examine the proficiency of NML MDL on real data, we tested it on time-series Drosophila gene expression measurements made by Arbeitman et al. [31]. The dataset in question consists of 4028 genes observed over 67 time points, which we binarized according to the procedure outlined in [10]. We selected 20 of these genes based on type (gap, pair-rule, etc.) and the availability of genetically verified directed interactions in the literature. Of the 32 edges identified by NML MDL (Figure 7), 16 have been previously demonstrated [32–43], and 3 more follow the standard genetic hierarchy [44]. Observe that 3 of the 12 other edges are simply reversals of known relationships and, therefore, could possibly represent unknown feedback mechanisms. Additionally, 5 of the remaining inferred relationships are between genes that are active in the same area such as the central nervous system (*Antp*/*runt*) and reproductive organs (*Notch*/*paired*) (the Interactive Fly website, hosted by the Society for Developmental Biology).

4. Concluding Remarks

Using a universal codelength when applying the MDL principle eliminates the relativity of applying ad hoc codelengths and user-defined tuning parameters. In our case, this has resulted in improved accuracy of Boolean network esimation. Using the theoretically grounded stochastic complexity instead of ad hoc encodings genuinely reflects the intent of the MDL principle. In addition, the structure function makes the proposed method faster than other published methods. Computation time does not heavily rely on bounded indegrees and increases only slightly with n.

Acknowledgments

This work was supported by the Academy of Finland (Application no. 213462, Finnish Programme for Centres of Excellence in Research 2006–2011), and the Tampere Graduate School in Information Science and Engineering. Partial support also provided by the National Cancer Institute (Grant no. CA90301).

References

[1] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Francisco, Calif, USA, 1988.

[2] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, "Using Bayesian networks to analyze expression data," *Journal of Computational Biology*, vol. 7, no. 3-4, pp. 601–620, 2000.

[3] T. Dean and K. Kanazawa, "A model for reasoning about persistence and causation," *Computational Intelligence*, vol. 5, no. 2, pp. 142–150, 1989.

[4] K. Murphy, "Dynamic Bayesian networks: representation, inference and learning," Ph.D. thesis, Computer Science Division, UC Berkeley, Berkeley, Calif, USA, 2002.

[5] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of Theoretical Biology*, vol. 22, no. 3, pp. 437–467, 1969.

[6] 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.

[7] H. Lähdesmäki, S. Hautaniemi, I. Shmulevich, and O. Yli-Harja, "Relationships between probabilistic Boolean networks and dynamic Bayesian networks as models of gene regulatory networks," *Signal Processing*, vol. 86, no. 4, pp. 814–834, 2006.

[8] D. Pe'er, A. Regev, G. Elidan, and N. Friedman, "Inferring subnetworks from perturbed expression profiles," *Bioinformatics*, vol. 17, supplement 1, pp. S215–S224, 2001.

[9] X. Zhou, X. Wang, R. Pal, I. Ivanov, M. Bittner, and E. R. Dougherty, "A Bayesian connectivity-based approach to constructing probabilistic gene regulatory networks," *Bioinformatics*, vol. 20, no. 17, pp. 2918–2927, 2004.

[10] W. Zhao, E. Serpedin, and E. R. Dougherty, "Inferring gene regulatory networks from time series data using the minimum description length principle," *Bioinformatics*, vol. 22, no. 17, pp. 2129–2135, 2006.

[11] S. Liang, S. Fuhrman, and R. Somogyi, "Reveal, a general reverse engineering algorithm for inference of genetic network architectures," *Pacific Symposium on Biocomputing*, vol. 3, pp. 18–29, 1998.

[12] T. Akutsu, S. Miyano, and S. Kuhara, "Identification of genetic networks from a small number of gene expression patterns under the Boolean network model," *Pacific Symposium on Biocomputing*, vol. 3, pp. 17–28, 1999.

[13] I. Shmulevich, A. Saarinen, O. Yli-Harja, and J. Astola, "Inference of genetic regulatory networks via best-fit extensions," in *Computational and Statistical Approaches to Genomics*, pp. 197–210, chapter 11, Kluwer Academic Publishers, New York, NY, USA, 2002.

[14] H. Lähdesmäki, I. Shmulevich, and O. Yli-Harja, "On learning gene regulatory networks under the Boolean network model," *Machine Learning*, vol. 52, no. 1-2, pp. 147–167, 2003.

[15] A. A. Margolin, I. Nemenman, K. Basso, et al., "ARACNE: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context," *BMC Bioinformatics*, vol. 7, supplement 1, p. S7, 2006.

[16] I. Nemenman, "Information theory, multivariate dependence, and genetic network inference," Tech. Rep. NSF-KITP-04-54, KITP, UCSB, Santa Barbara, Calif, USA, June 2004.

[17] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[18] J. Rissanen, "Stochastic complexity and modeling," *Annals of Statistics*, vol. 14, no. 3, pp. 1080–1100, 1986.

[19] V. Vapnik, *Estimation of Dependencies Based on Empirical Data*, Springer, New York, NY, USA, 1982.

[20] I. Tabus and J. Astola, "On the use of MDL principle in gene expression prediction," *EURASIP Journal on Applied Signal Processing*, vol. 2001, no. 4, pp. 297–303, 2001.

[21] J. Rissanen, *Information and Complexity in Statistical Modeling*, Springer, New York, NY, USA, 2007.

[22] A. Wuensche, "Genomic regulation modeled as a network with basins of attraction," *Pacific Symposium on Biocomputing*, vol. 3, pp. 89–102, 1998.

[23] I. Tabus, J. Rissanen, and J. Astola, "Normalized maximum likelihood models for Boolean regression with application to prediction and classification in genomics," in *Computational and Statistical Approaches to Genomics*, pp. 173–196, chapter 10, Kluwer Academic Publishers, New York, NY, USA, 2002.

[24] W. Szpankowski, "On asymptotics of certain recurrences arising in universal coding," *Problems of Information Transmission*, vol. 34, no. 2, pp. 55–61, 1998.

[25] D. Thieffry, A. M. Huerta, E. Pérez-Rueda, and J. Collado-Vides, "From specific gene regulation to genomic networks: a global analysis of transcriptional regulation in *Escherichia coli*," *BioEssays*, vol. 20, no. 5, pp. 433–440, 1998.

[26] S. A. Kauffman, *The Origins of Order*, Oxford University Press, Oxford, UK, 1993.

[27] R. Pal, I. Ivanov, A. Datta, M. L. Bittner, and E. R. Dougherty, "Generating Boolean networks with a prescribed attractor structure," *Bioinformatics*, vol. 21, no. 21, pp. 4021–4025, 2005.

[28] I. Shmulevich and S. A. Kauffman, "Activities and sensitivities in Boolean network models," *Physical Review Letters*, vol. 93, no. 4, Article ID 86394, 4 pages, 2004.

[29] B. Derrida and Y. Pomeau, "Random networks of automata: a simple annealed approximation," *Europhysics Letters*, vol. 1, pp. 45–49, 1986.

[30] S. Harris, B. Sawhill, A. Wuensche, and S. A. Kauffman, "A model of transcriptional regulatory networks based on biases in the observed regulation rules," *Complexity*, vol. 7, no. 4, pp. 23–40, 2002.

[31] M. Arbeitman, E. Furlong, F. Imam, et al., "Gene expression during the life cycle of *Drosophila melanogaster*," *Science*, vol. 297, no. 5590, pp. 2270–2275, 2002.

[32] J. Bhojwani, L. S. Shashidhara, and P. Sinha, "Requirement of teashirt (tsh) function during cell fate specification in developing head structures in *Drosophila*," *Development Genes and Evolution*, vol. 207, no. 3, pp. 137–146, 1997.

[33] D. M. Cimbora and S. Sakonju, "*Drosophila* midgut morphogenesis requires the function of the segmentation gene *odd-paired*," *Developmental Biology*, vol. 169, no. 2, pp. 580–595, 1995.

[34] M. Fujioka, J. Jaynes, and T. Goto, "Early *even-skipped* stripes act as morphogenetic gradients at the single cell level to establish *engrailed* expression," *Development*, vol. 121, no. 12, pp. 4371–4382, 1995.

[35] M. González-Gaitan and H. Jäckle, "Invagination centers within the *Drosophila* stomatogastric nervous system anlage are positioned by *Notch*-mediated signaling which is spatially controlled through *wingless*," *Development*, vol. 121, no. 8, pp. 2313–2325, 1995.

[36] L. D. Mathies, S. Kerridge, and M. P. Scott, "Role of the *teashirt* gene in *Drosophila* midgut morphogenesis: secreted proteins mediate the action of homeotic genes," *Development*, vol. 120, no. 10, pp. 2799–2809, 1994.

[37] S. Morimura, L. Maves, Y. Chen, and F. M. Hoffmann, "*Decapentaplegic* overexpression affects *Drosophila* wing and leg imaginal disc development and *wingless* expression," *Developmental Biology*, vol. 177, no. 1, pp. 136–151, 1996.

[38] B. S.-L. Dréan, A. Nasiadka, J. Dong, and H. M. Krause, "Dynamic changes in the functions of Odd-skipped during early *Drosophila* embryogenesis," *Development*, vol. 125, no. 23, pp. 4851–4861, 1998.

[39] V. Schaeffer, D. Killian, C. Desplan, and E. A. Wimmer, "High Bicoid levels render the terminal system dispensable for *Drosophila* head development," *Development*, vol. 127, no. 18, pp. 3993–3999, 2000.

[40] P. Steneberg, J. Hemphälä, and C. Samakovlis, "Dpp and Notch specify the fusion cell fate in the dorsal branches of the *Drosophila* trachea," *Mechanisms of Development*, vol. 87, no. 1-2, pp. 153–163, 1999.

[41] I. S. Torres, H. López-Schier, and D. St. Johnston, "A Notch/Delta-dependent relay mechanism establishes anterior-posterior polarity in *Drosophila*," *Developmental Cell*, vol. 5, no. 4, pp. 547–558, 2003.

[42] J. Torres-Vazquez, S. Park, R. Warrior, and K. Arora, "The transcription factor Schnurri plays a dual role in mediating Dpp signaling during embryogenesis," *Development*, vol. 128, no. 9, pp. 1657–1670, 2001.

[43] Z. Yin, X.-L. Xu, and M. Frasch, "Regulation of the twist target gene *tinman* by modular *cis*-regulatory elements during early mesoderm development," *Development*, vol. 124, no. 24, pp. 4971–4982, 1997.

[44] M. D. Schroeder, M. Pearce, J. Fak, et al., "Transcriptional control in the segmentation gene network of *Drosophila*," *PLoS Biology*, vol. 2, no. 9, p. e271, 2004.