Systems biology

An effective structure learning method for constructing gene networks

Xue-wen Chen¹,*, Gopalakrishna Anantha¹ and Xinkun Wang²

¹Bioinformatics and Computational Life Sciences Laboratory, Electrical Engineering and Computer Science Department, 1520 West 15th Street, The University of Kansas, Lawrence, KS 66045, USA and ²Higuchi Biosciences Center, 2099 Constant Avenue, The University of Kansas, Lawrence, KS 66047, USA

Received on December 5, 2005; revised and accepted on March 7, 2006
Advance Access publication March 16, 2006
Associate Editor: John Quackenbush

ABSTRACT

Motivation: Bayesian network methods have shown promise in gene regulatory network reconstruction because of their capability of capturing causal relationships between genes and handling data with noises found in biological experiments. The problem of learning network structures, however, is NP hard. Consequently, heuristic methods such as hill climbing are used for structure learning. For networks of a moderate size, hill climbing methods are not computationally efficient. Furthermore, relatively low accuracy of the learned structures may be observed. The purpose of this article is to present a novel structure learning method for gene network discovery.

Results: In this paper, we present a novel structure learning method to reconstruct the underlying gene networks from the observational gene expression data. Unlike hill climbing approaches, the proposed method first constructs an undirected network based on mutual information between two nodes and then splits the structure into substructures. The directional orientations for the edges that connect two nodes are then obtained by optimizing a scoring function for each substructure. Our method is evaluated using two benchmark network datasets with known structures. The results show that the proposed method can identify networks that are close to the optimal structures. It outperforms hill climbing methods in terms of both computation time and predicted structure accuracy. We also apply the method to gene expression data measured during the yeast cycle and show the effectiveness of the proposed method for network reconstruction.

Contact: xwchen@ku.edu

1 INTRODUCTION

As basic building blocks of life, genes, as well as their products (proteins), do not work independently. Instead, in order for a cell to function properly, they interact with each other and form a complicated network. Gene networks represent the relationship between sets of genes that coordinate to achieve different tasks.

With the advent of high-throughput microarray technologies, mRNA expression levels of tens of thousands of genes can now be measured simultaneously. This whole-genome-scale high-throughput technology has revolutionized research in the life science arena since its invention (Lipshutz et al., 1995; Schena et al., 1995). Large amount of microarray data has been generated since then and deposition of these data in public domains has made gene networking study possible. Construction of gene networks from these experimental data will greatly facilitate cellular functional dissection at the molecular level. Gene networking study will advance the whole spectrum of biomedical research, from cancer research, toxicology, to drug discovery and disease prevention.

While a variety of computational methods have been considered for reconstructing gene networks from observational gene expression data including, for example, linear models (Van Someren et al., 2000; D’Haeseleer and Fuhrman, 1999; Deng et al., 2005; Di Bernardo et al., 2004; De Jong et al., 2004; De Hoon, et al., 2003; Chen et al., 2005; D’Haeseleer et al., 2000) and Boolean network models (Huang, 1999; Kaufman, 1993; Shmulevich, et al., 2002), Bayesian network (BN) based approaches have shown great promise to infer causal relationships between genes and receive increasing attention (Badea, 2004; Bernard and Hartemink, 2005; Friedman et al., 1999a; Friedman et al., 2000; Hartemink et al., 2001; Husmeier, 2003; Murphy and Mian, 1999; Otta et al., 2004; Pe’er et al., 2001; Pena et al., 2005; Perrin et al., 2003; Pournara and Wernisch, 2004; Smith et al., 2002, 2002; Yu et al., 2004; Zhou et al., 2004; Zou and Conzen, 2005). BNs (Cooper and Herskovits, 1992; Heckerman et al., 1995) are especially suitable for learning genetic regulatory networks for the following reasons: (1) the sound probabilistic semantics allows BNs to deal with the noises that are inherent in experimental measurements; (2) BNs can handle missing data and permit the incomplete knowledge about the biological system and (3) BNs are capable of integrating prior biological knowledge into the system.

Generally, a BN is a graphical representation of the dependence structure between multiple interacting quantities. This graphical representation is more commonly called a directed acyclic graph (DAG). The nodes or the vertices of the DAG represent the random variables in the network while the edges connecting the vertices represent the causal influence of one node on the other. BN-based gene network inference requires the learning of the BN structure, which is an optimization problem in the space of the DAGs. Many structure learning methods have been proposed, which can be categorized as either conditional independence (CI) test-based methods or scoring-based methods. The CI-based methods analyze the dependence and independence relationships among variables via CI tests and construct the networks that characterize these relationships (Cheng et al., 2002; de Campos and Huete, 2000; Meek 1995; Geiger et al., 1993). The CI-based methods are computationally efficient but not very accurate. Scoring-based methods use a score function to measure the goodness of fit of a network structure to the data and search the space of possible network structures using model selection algorithms. The scores can be additive (e.g. BDe, K2 scores) or non-additive (e.g. BIC, BDD, MM, MDI, MDL scores), and consequently the optimization problems have different complexity. hill climbing are used for structure learning. For networks of a moderate size, hill climbing methods are not computationally efficient. Furthermore, relatively low accuracy of the learned structures may be observed. The purpose of this article is to present a novel structure learning method for gene network discovery.

Results: In this paper, we present a novel structure learning method to reconstruct the underlying gene networks from the observational gene expression data. Unlike hill climbing approaches, the proposed method first constructs an undirected network based on mutual information between two nodes and then splits the structure into substructures. The directional orientations for the edges that connect two nodes are then obtained by optimizing a scoring function for each substructure. Our method is evaluated using two benchmark network datasets with known structures. The results show that the proposed method can identify networks that are close to the optimal structures. It outperforms hill climbing methods in terms of both computation time and predicted structure accuracy. We also apply the method to gene expression data measured during the yeast cycle and show the effectiveness of the proposed method for network reconstruction.

Contact: xwchen@ku.edu

¹To whom correspondence should be addressed.
effective. However, the scoring-based methods may produce more accurate results in structure learning than the CI-based methods (Acid and de Campos, 2003; Cooper and Herskovits, 1992). CI-based methods may miss true edges between two nodes that are not strongly related (in terms of the CI-test). In addition, CI-based methods may fail to provide orientations for all the edges in a network (Cheng et al., 2002). The scoring-based methods (Cooper and Herskovits, 1992; Heckerman, 1998; Friedman and Goldszmidt, 1996; Chickering, 2002) consist of two components: (1) a scoring function that assesses how well a network fits the data and (2) a search method to find networks with high scores. They are capable of finding the most probable network structures (with edge directions assigned) given a dataset. In addition, a Bayesian scoring-based metric can prevent the model from over-fitting the data, as the score metric is an average over a family of probability distributions (Hartemink et al., 2001). This is critical in gene regulatory network reconstruction using microarray as only a small number of samples is available. It is worth noting a recent finding by Acid et al. (2004). They conducted a comparative study of different structure learning algorithms on data from an emergency medical service. Since there is no gold standard network structure, Acid et al. (2004) used the Kullback–Leibler distance, K2 score, BDeu score and BIC metrics to evaluate the learned structures. For all the metrics, the best structures are always learned by scoring-based methods. Interestingly, when the networks learned are used for classification purposes, the results are mixed: while some networks learned from scoring-based methods perform better than those learned from CI-based methods for some variables, they perform worse for others in terms of classification accuracy. Similar results are also reported in Friedman et al. (1997). Our work focuses on structure learning and we consider scoring-based approaches. A major disadvantage of scoring-based methods for structure learning is the large computational cost: since the number of possible graphs in the learning process (a process of reconstructing gene networks) is super-exponential in the number of random variables considered in the network, exact structure learning is known to be computationally hard (Chickering, 1996). As an example, the number of possible DAGs for a network with eight random variables is $\sim 7.8 \times 10^{11}$. Thus, heuristic search algorithms have to be employed to reduce the computational complexity. Currently, most of the common search algorithms used in reconstructing gene networks are greedy search algorithm, Hill climbing algorithm and Markov Chain Monte Carlo (MCMC) search techniques (Yu et al., 2004; Friedman et al. 1999b; Zhou et al., 2004). While these search methods work reasonably well for networks of small sizes (say, <10 nodes), they are not computationally efficient for networks of moderate sizes. Furthermore, relatively low accuracy of the learned structures may be observed. The purpose of this article is to present an efficient structure learning method with high accuracy.

We propose a novel structure learning method that is based on information theory and K2 algorithm. K2 algorithm is a commonly used greedy search algorithm in BN learning. The performance of the K2 algorithm is greatly affected by a prior ordering of input nodes. If all parents in the node ordering occur prior to their children in the node ordering, the algorithm will perform optimally and consequently the results are very accurate (Cooper and Herskovits, 1992). The K2 algorithm is very efficient as the node-ordering information reduces the search space of DAGs (thus making the search non-exhaustive). However, the performance of the algorithm may be poor when using wrong orderings in which most children nodes appear prior to their parents and for orderings that are random in nature. The input node ordering is usually unknown in most cases, especially in our applications. In this paper, we develop an efficient method to identify nodes ordering from the experimental data so as to improve the performance and efficiency of the K2 algorithm in determining the network structure. Our method is evaluated using two benchmark network datasets with known structures. The results show that the proposed method can identify networks that are close to the optimal structures. It outperforms hill climbing methods in terms of both computation time and predicted structure accuracy. We also apply the method to gene expression data measured during the yeast cycle and show the effectiveness of the proposed method for network reconstruction.

The remainder of this article is organized as follows: In Section 2, we review the basic concepts of BNs and K2 algorithms. Section 3 describes our method to process experimental data to obtain an optimal node ordering for the K2 algorithm. In Section 4, we present the results of our algorithm for standard datasets (the ASIA network and the ALARM network). This section also details how our method is applied to actual microarray data. Finally in Section 5, we conclude the work.

## 2 BAYESIAN NETWORKS

A BN is a graphical model that encodes the probabilistic dependencies among a set of variables (Heckerman, 1998). It consists of two important components: a DAG representing the dependency structure among the variables in the network and a conditional probability table or a distribution for each variable in the network given its parent set. To learn the structure of a network that describes the causal relationship among variables, one needs to have (1) a scoring function that assesses how well a network fits the data and (2) a search method to find networks with high scores.

### 2.1 Network scoring function

A BN is a DAG that represents a joint probability distribution of a set of random variables $\{X_1, X_2, \ldots, X_n\}$. The nodes of the DAG are represented by this set of random variables. Let $Pa_i$ denote the set of parents of $X_i$ in the DAG. The DAG encodes the Markov relation which states the following: each variable $X_i$ in the DAG is independent of its non-descendants given its set of parents in the DAG. Mathematically the joint distribution can be decomposed into a product form as

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | Pa_i). \quad (1)$$

This is referred to as the chain rule for BNs.

Learning a BN structure is to find a DAG that best match the dataset. The common method of structure learning is to define a scoring function that evaluates how well the DAG explains the data and then to search for the best DAG that optimizes the scoring function. A commonly used scoring function for discrete data is called BDe scoring metric which computes the posterior probability of a network for the given data (Cooper and Herskovits, 1992). In this article, we will use BDe scoring metric to evaluate BNs.
2.2 K2 algorithm

As the number of possible graph structures is super-exponential in ‘n’ (the number of nodes in the network), searching exhaustively over the space of the DAGs is computationally very challenging. Thus, heuristic methods such as K2 are typically used. The K2 Algorithm (Cooper and Herskovits, 1992) is a greedy search algorithm that learns the network structure of the BN from the data presented to it. It attempts to select the network structure that maximizes the network’s posterior probability given the experimental data. The K2 algorithm reduces this computational complexity by requiring a prior ordering of nodes as an input, from which the network structure will be constructed. The ordering is such that if node $X_i$ comes prior to node $X_j$ in the ordering, then node $X_j$ cannot be a parent of node $X_i$. In other words, the potential parent set of node $X_i$ can include only those nodes that precede it in the input ordering.

The candidate parents $P_0$ for node $X_i$ is initially set to nothing (zero-empty set). The algorithm visits each node $X_i$ according to the sequence specified in the prior node ordering and greedily adds $P_0$ to the parent set of node $X_i$, if the addition of the parent to the node $X_i$ maximizes the score of the network. The addition of parents stops when the addition of no single parent can increase the posterior probability.

The performance of the K2 algorithm is greatly affected by the input node ordering. An improper order will result in poor results. Thus, it is critical to provide the K2 algorithm with correct node ordering. In the task of reconstructing gene networks from microarray data, however, the ordering information is not available. Next, we propose a novel method to find such orderings.

3 METHODS

The main concept of our algorithm is to process the data to identify an appropriate node ordering that can be used by the K2 algorithm to learn the structure of BNs. It consists of two steps: (1) The first step is to construct an undirected network (i.e., the edge directions in the network are not assigned) based on mutual information (MI). This undirected network allows us to search the best DAG in a reduced space. The degree of dependency between random variables will give us an approximate estimate of how the variables in the network are related. (2) The second step is to assign directions to these edges in the undirected network. This provides nodes ordering information that will be used as an input in K2 algorithm. In this step, the undirected network is split into sub-graph. For each sub-graph, we can identify the directions based on the BDe score. Next, we detail the two steps.

3.1 Constructing undirected networks

We first construct the undirected networks based on MI. The MI between two random variables $X$ and $Y$, denoted by $I(X; Y)$, is defined as the amount of information shared between the two variables. It is used to detect general dependencies in data. $I(X; Y)$ is mathematically defined as follows:

$$ I(X; Y) = H(X) - H(X|Y), $$

where, $H(X)$ represents the entropy of the random variable $X$, $H(Y)$ represents the entropy of the random variable $Y$, and $H(X|Y)$ represents the conditional entropy of random variable $X$ given the random variable $Y$ (Proakis, 2001). Entropy and conditional entropy are mathematically defined by the following equations:

$$ H(X) = \sum_{i=1}^{n} P(X_i) \log(P(X_i)) $$

$$ H(X|Y) = -\sum_{i=1}^{n} \sum_{j=1}^{m} P(X=x_i, Y=y_j) \log((P(X=x_i|Y=y_j))). $$

In these equations, $n$ and $m$ are the number of discrete states taken by the random variables $X$ and $Y$, respectively. $P(X=x_i, Y=y_j)$ is the joint probability distribution of the two random variables $X$ and $Y$.

MI measures the dependency between two random networks. The greater the MI values between two random variables, the more closely they are related. Thus, if there is a direct edge (connection) between two nodes $X_i$ and $X_j$, there exists a strong dependency between these two nodes. In other words, if $I(X_i; X_j) = \max I(X_i; X_k), \forall k \neq i, j$ or $I(X_i; X_j) = \max I(X_i; X_k), \forall k \neq i, j$

it is most likely that the two nodes $X_i$ and $X_j$ are directly connected. Consequently, we can construct a partial network (undirected) based on the MI values. The steps to construct the undirected structures are as follows: (1) we first calculate the MI between all pairs of nodes in the network; (2) the pairwise MI is then arranged in descending order and (3) an edge is added between the pair of nodes associated with the MI values one at a time in the sorted MI list, if there is no other existing path (one edge or more edges) between these two nodes. The algorithm stops when all the nodes are connected in the network with at least one edge.

The undirected network will not contain any loops. This will avoid redundant paths. For example, in ASIA network (details about this network are in Section 4 ‘Results’), the true connections among nodes $X_3$, $X_6$ and $X_8$ are shown in Figure 1 and in the MI list, $I(X_3; X_6)$, $I(X_8; X_6)$ and $I(X_8; X_3)$ are the highest, third highest and fourth highest. Consequently, an edge between nodes $(X_3, X_6)$ and then $(X_8, X_3)$ will be established. Since there exists a path between $X_3$ and $X_6$ (through $X_8$ with two edges), the direction connection (edge) between $X_3$ and $X_6$ will not be created [although $I(X_3; X_6)$ is the fourth highest value in the MI list]. Apparently, the large MI between $X_3$ and $X_6$ is because both are connected directly to $X_8$.

For some simple networks, this step may completely recover the true network structures (without edge directions). For most networks, however, step 1 only reconstructs partially the network structures with all the nodes. Our primary goal here is to discover the node ordering information from the partial networks, which can then be used in K2 algorithm to construct the original structures completely. The next task in our algorithm is to find the directional orientations among the different nodes in the undirected structure to obtain an ordering which the K2 algorithm can use.

3.2 Graph splitting for direction assignment

To specify the node ordering, we use the concept of graph splitting and the score function to determine the directional orientations of the edges in the undirected networks constructed from previous step.

The undirected graph structure is first split into smaller sub-networks. For each node, the number of edges connecting to it is counted; nodes are then arranged in descending order in terms of the number of nodes connecting to it. A node and all the nodes that are directly connected to it form a sub-network. For each sub-network, we use exhaustive search method to find the optimal edge orientations that maximize BDe score for this sub-network. Decomposing a large network into many small networks and finding optimal subnetworks accordingly are reasonable because to maximize the BDe for the whole network, we need only to find all the subnets that optimize the associated BDe (Cooper and Herskovits, 1992). In addition, although exhaustive search is computationally prohibitive for original networks
Ordered K2

\[\{\text{Input: a set of } n \text{ nodes and a dataset } D = \{X_1, \ldots, X_n\} \text{ containing } m \text{ cases.}\}\]
\[\{\text{Output: the parents for each node}\}\]
1. set up an empty MI list: MIL = \{empty set\}.
2. for \(i = 1 \text{ to } n - 1\) do
   for \(j = i + 1 \text{ to } n\) do
     compute \(I_{ij} = I(Y_i; X_j)\)
     \[\text{MIL} = \text{MIL} \cup \{I_{ij}\}\]
   end\ (for \(j\))
end\ (for \(i\)): Compute the MI between every pair that exists in the network.
3. sort the MIL in descending order to form a new list.
\[\text{MIL}^* = \{I_1(Y_1; Z_2), \ldots, I_n(Y_n; Z_k)\}\]
   where \(k = n(n-1)/2\); \(Y_i\) and \(Z_i\) are the two \(X\) nodes with the \(i\)-th highest MI \(I(Y_i; Z_i)\) in the list.
4. set \(k = 1\).
5. while (not all the nodes are included in the network) do
   \[\text{if (there is no other existing path between } Y_i \text{ and } Z_k)\] then
     Create an edge between \(Y_i\) and \(Z_k\).
   end\ (if)
   \[k = k + 1;\]
end\ (while); Begin the construction of the undirected graph by establishing edges between strongly related nodes.
6. for each node, obtain the number of edges connecting to it; arrange the nodes in descending order in terms of this number \(\{\text{node}_1, \text{node}_2, \ldots, \text{node}_n\}\).
set \(k = 1\).
7. while (not all the edges are assigned directions) do
   \[\text{form a undirected sub-network } \text{subNet}_k, \text{consisting of } \text{node}_k, \text{and all the } \text{nodes that are directly connected with it};\]
   \[\text{form the directed sub-networks } \text{subNet}_{ij}\] by obtaining all possible directional orientations for the edges in the sub-network and compute the BDe score for each \(\text{subNet}_{ij}\).
   \[\text{select the sub-network } \text{subNet}_{ij} \text{ that gives the maximum BDe score and the corresponding edge directions are assigned.}\]
   \[k = k + 1;\]
end\ (while); Assign directions for all the edges
8. obtain the node ordering from the directed network constructed in previous step.
9. run k2 algorithm: the ordered nodes are used as the input of K2.

4 RESULTS

To evaluate the proposed ordered K2 method, we first perform tests for two standard networks—ASIA network (Olesen and Madsen, 2002), which is an eight variable network, and the ALARM network (Beinlich et al., 1989), which is a 37 variable network. Both networks are representative of real life BNS with known structures. The random variables in the ASIA network take two discrete states while those in the ALARM network take two, three and four states (depending on the node). The data samples used in the testing process are randomly generated for each run of the algorithm. We then apply this method to gene expression data for inferring genetic regulatory networks.

4.1 Results on known structures

The algorithm developed was tested with experimental data samples randomly generated for each run of the test. The random generation of data samples was done to ensure the robustness of the algorithm. In case of the ASIA and ALARM networks, the existence of the known network structures allows us to define three important terms, which indicate the performance of the algorithm (in terms of the number of graphical errors in the learnt structure).

- Missing edges (M): Edges present in the original network but not in the learnt network structure.
- Wrongly oriented edges (WO): Edges present in the learnt network structure, but having opposite orientation when compared with the corresponding edge in the original network structure.
- Wrongly connected edges (WC): Edges not present in the original network but included in the learnt network structure.

Tables 1 and 2 show the performances of the algorithm for the ASIA (8 variables and 8 edges) and ALARM networks (37 variables and 46 edges), respectively, with 10 000 data samples generated.
randomly 100 times. Both the mean results (the results averaged over 100 trial runs) and the best results (best result among 100 trial runs) are presented in Tables 1 and 2. As a means to compare our algorithm, we present the results obtained while running the hill-climbing method (one of the most commonly used structure search methods) and K2 algorithm both with the correct order (of which we have the knowledge, since the network structure is known) and with the random order. The results for K2 with correct order are the optimal results one can get.

Tables 1 and 2 show that our method can produce results that are close to the optimal results obtained from the K2 algorithm with the correct node ordering. On average, for the ASIA network, our method only misses 0.38 edges (out of 8 edges); the K2 with correct node ordering misses 0.33 edges. For the ALARM network, our method misses 0.91 edges (out of 46 edges), while the K2 with correct node ordering misses 0.95 edges; among these identified edges, our methods yields 3.54 wrong directions (i.e. the interactions are identified, but the directions are wrong). The best results obtained from our method and the K2 with correct ordering method are very close. We also compare our method with random-ordering K2 methods and hill climbing. The proposed method is substantially better than both the methods. Thus, our method is capable of reconstructing the networks from data. It is interesting to note that for both datasets, hill climbing methods may produce network structures that are far away from original structures.

We compare the computational time of K2 algorithm with our method with hill climbing methods. The tests are on an Intel Xeon 2.8 GHz CPU and 1 GB RAM. From Table 3, we can see that for ASIA network with small size, the computation time of our method is ~19 s and hill climbing needs ~67 s. The difference of computational time is significantly large when considering networks of moderate size: for the ALARM network, our method needs only ~17 min, while hill climbing methods need >900 min.

### 4.2 Inferring gene networks from microarray data

The microarray dataset we employed is from a previous study on yeast cell-cycle gene regulation (Spellman et al., 1998) and is publicly available. This previous study was originally designed to identify yeast genes whose transcription levels vary periodically within the cell cycle. The microarray dataset contains measurements for 6000 genes across about 80 time points (i.e. 6000 variables and 80 data samples). To create this large dataset, DNA samples extracted from yeast cultures synchronized by three independent protocols were used for microarray hybridization. A total of 800 genes were identified to be regulated by cell cycle in the original report. In this paper, we used 77 data samples for these 800 genes.

A 2-level quantization scheme was chosen to discretize the continuous data in the microarray dataset. This quantization scheme is based on the distance of a particular sample of the gene expression from the mean for that particular gene (variable). Two groups of genes are selected for the network construction based on their known interaction patterns (although the currently known interactions are still incomplete at present). The first group (group I) included eight histone genes—HHT1, HHT2, HHF1, HHF2, HTA1, HTA2, HTB1 and HTB2. These eight genes encode for the four histones (H2A, H2B, H3 and H4), which are used to form the code of nucleosome, i.e. the fundamental packaging unit of chromatin. Prior to cell division in the cell cycle, chromosomes, which are composed of both DNA and histones, need to be replicated. In order for the replication process to be executed properly, expression of the histone genes need to be tightly regulated. The network built for the eight histone genes with our method (Fig. 2) uncovered 86% (12 out of 14) of the currently known interactions among them. Table 3 lists the edges in this network supported by previous experimental reports. The search of these previous reports were conducted with PathwayAssist (version 3.0), which is based on a comprehensive gene (or protein) interaction database compiled by a text mining tool from the entire PubMed (Nikitin et al., 2003). Since PathwayAssist is based on the currently available experimental data, the search for supporting evidence of our established edges was not exhaustive. Some presently unsupported edges in the constructed network may find experimental evidence in the future. Therefore, these unsupported edges are not necessarily false ones.

To further evaluate the performance of the proposed method, we selected another group (Group II), which consisted of 20 genes. These genes perform more diverse functions (including DNA replication initiation, DNA damage-induced checkpoint arrest, DNA damage repair, formation of mitotic spindle, and so on) in the process of cell cycle than the histone genes. However, similar to the histone genes, their expression is also strictly regulated in order for the whole process to proceed normally. Our method found 34 edges among these 20 genes (or nodes, Fig. 3). PathwayAssist search identified that a total of seven edges (19.4%) are supported by previously published data (Table 4). For the remaining 27 edges...
extremely high dimensionality (number of genes), learning a
tion levels. Owing to a relatively small number of samples and the
algorithms depends on the number of genes used and the quantiza-
evidence.

Assist. In total, 22 edges, i.e. 65% of all edges in our BN, were
the currently available experimental evidence found by PathwayAssist; dotted line, connection between these two nodes is also
connected via a single intermediate node in the graph obtained using
PathwayAssist; bold faced dash line, found from PathwayAssist; bold faced solid line, direct connection between
found from PathwayAssist; bold faced solid line, direct connection between
these two nodes identified using the PathwayAssist software; bold faced dash line, connection between these two nodes is not direct, i.e. these nodes are connected via two intermediate nodes in the graph obtained using PathwayAssist.

in our BN, there were no corresponding direct interactions found by
PathwayAssist. However, a closer examination of the PathwayAssist search results revealed that, out of the 27 edges, 15 were
also indirectly supported by existing experimental data. By indirectly,
we mean that these 15 edges (or interactions) are existent through intermediate nodes (or genes). Among the 15 edges, a total
of 6 edges were present through a single intermediate node and the
remaining 9 edges were through two intermediate nodes based on all
the currently available experimental evidence found by PathwayAssist. In total, 22 edges, i.e. 65% of all edges in our BN, were
supported either directly or indirectly by biological experimental evidence.

It is worth noting that the performance of structuring learning
algorithms depends on the number of genes used and the quantization levels. Owing to a relatively small number of samples and the extremely high dimensionality (number of genes), learning a

Table 4. Direct experimental support for the interactions uncovered by our algorithm

<table>
<thead>
<tr>
<th>Group</th>
<th>Interacting genes</th>
<th>Our method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group I</td>
<td>HTA2–HTA1</td>
<td>von Mering et al. (2002)</td>
</tr>
<tr>
<td></td>
<td>HTA2–HHT2</td>
<td>von Mering et al. (2002)</td>
</tr>
<tr>
<td></td>
<td>HTA2–HTB2</td>
<td>von Mering et al. (2002)</td>
</tr>
<tr>
<td></td>
<td>HTA2–HHT1</td>
<td>von Mering et al. (2002)</td>
</tr>
<tr>
<td></td>
<td>HTB2–HHF1</td>
<td>von Mering et al. (2002)</td>
</tr>
<tr>
<td></td>
<td>HTB2–HTB1</td>
<td>von Mering et al. (2002)</td>
</tr>
<tr>
<td></td>
<td>HHT2–HHF2</td>
<td>von Mering et al. (2002)</td>
</tr>
<tr>
<td></td>
<td>HHF1–HHT1</td>
<td>von Mering et al. (2002)</td>
</tr>
<tr>
<td></td>
<td>HHF2–HHF2</td>
<td>von Mering et al. (2002)</td>
</tr>
<tr>
<td></td>
<td>HHF2–HHT1</td>
<td>Venditti et al. (1999)</td>
</tr>
<tr>
<td></td>
<td>HHF2–HTA1</td>
<td>von Mering et al. (2002)</td>
</tr>
<tr>
<td></td>
<td>HTA1–HTB1</td>
<td>Mosammaparast et al. (2001)</td>
</tr>
<tr>
<td>Group II</td>
<td>POL30–MCD1</td>
<td>von Mering et al. (2002)</td>
</tr>
<tr>
<td></td>
<td>MSH2–RFA3</td>
<td>von Mering et al. (2002)</td>
</tr>
<tr>
<td></td>
<td>PRI1–RAD53</td>
<td>von Mering et al. (2002)</td>
</tr>
<tr>
<td></td>
<td>CLB5–RAD53</td>
<td>von Mering et al. (2002)</td>
</tr>
<tr>
<td></td>
<td>HTA2–HTA1</td>
<td>von Mering et al. (2002)</td>
</tr>
<tr>
<td></td>
<td>HTA1–HTB1</td>
<td>Mosammaparast et al. (2001)</td>
</tr>
<tr>
<td></td>
<td>POL30–MCD1</td>
<td>von Mering et al. (2002)</td>
</tr>
</tbody>
</table>

network for all the genes (in our case, 6000) is infeasible.
Consequently, current methods focus on a small group of genes.
The genes that are considered in reconstructing regulatory networks can be selected using either methods like correlation analysis and clustering algorithms (e.g. Pe’er et al., 2001) or from some biological knowledge (e.g. Hartemink et al., 2001; Soinov et al., 2003). More genes could be handled if extra information from different sources such as protein–protein interaction and transcription factor groups is available and integrated into the BN model (Nariai et al., 2004). Generally speaking, the number of genes that can be handled depends on several factors, most importantly, the number of samples available and the connectivity of gene networks: the more samples and the less connectivity (e.g. each gene has at most two parents), more genes can be handled by the scoring-based methods. Discretization level is another factor that may affect the performance of structure learning algorithms. The more levels to use, more information will be kept; however, the more samples are needed to build a robust network. For small number of samples like microarray data, a large number of discretization levels may lead to an unstable model. The most commonly used levels are two (Hartemink et al., 2001; Soinov et al., 2003) and three (Yu et al., 2004; Friedman et al., 2000). For the particular datasets in our studies, the network structures are similar when the data are discretized into two or three levels. For more than three levels of discretization, the algorithm did not yield meaningful results: many genes are isolated, i.e. not connected to any other genes. As suggested by Yu et al. (2004), this occurs because the finer discretization increases the difficulty of finding dependencies between nodes. Thus, more samples are required for high discretization levels.

Fig. 3. BN built with our algorithm for yeast cell cycle-related genes. A total of 20 nodes (genes) and 34 edges (interactions) were incorporated. This network captured 65% of all currently reported direct and indirect interactions among these genes. Legend used: solid line, predicted connections that are not found from PathwayAssist; bold faced solid line, direct connection between two nodes identified using the PathwayAssist software; bold faced dash line, connection between these two nodes is not direct, i.e. these nodes are connected via a single intermediate node in the graph obtained using PathwayAssist; dotted line, connection between these two nodes is also not direct, i.e. these nodes are connected via two intermediate nodes in the graph obtained using PathwayAssist.
5 DISCUSSIONS AND CONCLUSIONS

Cellular processes occurring in organisms are carried out through networks of regulatory interactions among genes and their products. To uncover these regulatory networks from experimental data remains one of the most challenging problems for functional genomics. While the microarray technology has generated large amount of gene expression data for gene networking research, reconstructing gene regulatory networks with in silico methods is still not a trivial problem. In this paper, we described a new K2 algorithm-based approach for gene network discovery. The proposed method identifies the ordering of nodes for K2 algorithm using information theory and graph splitting strategy. Thus, the inherent efficiency of the K2 algorithm can be made use of in the process of learning BNs.

To test the performance of the proposed algorithm, it was first applied to two well-known network structures—ASIA and ALARM networks. The results demonstrated that the proposed method is capable of generating results that are close to optimal. Furthermore, the proposed methods outperformed random-order based K2 methods and yielded significantly better results. The proposed method also outperformed hill climbing methods in terms of both the accuracy of reconstructed structures and computational time. Another search method used in structure learning is MCMC. However, MCMC algorithms are much slower than hill climbing, especially for learning networks of moderate sizes.

To evaluate its efficacy to real-world microarray data, we applied it to a publicly available microarray dataset for two groups of genes whose interaction patterns are mostly known. Although the performance of the algorithm cannot be accurately determined, primarily because of the incompleteness of currently available biological information, at least 86 and 65% of gene interactions in the two inferred gene networks were found to be consistent with results published previously based on experimental data. For inferring gene networks from microarray data, one may also combine the proposed structure learning method with bootstrap approach (Friedman et al., 1999a).

For the currently experimentally unsupported interactions in the two inferred Bayesian gene network, we assume that some represent false relationships which were picked up by our algorithm by chance or by data noises. However, it is also possible that some of these interactions are novel and currently unknown, because the available experimental data for these genes are still far from complete at present. The purpose of our Bayesian gene network construction is not just to confirm currently existing data, but also lead to the generation of new interactions among genes and moreover, new hypothesis. The validation of the new interactions, however, is beyond the scope of this article.

ACKNOWLEDGEMENTS

The authors thank Kevin P. Murphy, Department of Computer Science and Statistics, University of British Columbia for the Bayesnet Toolbox. The authors also thank Olivier Francois, PSI Laboratory, France for his useful help. This article is based upon the work supported by the NASA EPSCoR Grant and by the J. R. & Inez Jay Fund from Higuchi Biosciences Center at the University of Kansas.

Conflict of Interest: none declared.

REFERENCES


