Inferring large-scale gene regulatory networks using a low-order constraint-based algorithm†

Mingyi Wang,[a] Vagner Augusto Benedito,[ab] Patrick Xuechun Zhao[a] and Michael Udvardi*[a]

Received 26th August 2009, Accepted 7th January 2010
First published as an Advance Article on the web 19th February 2010
DOI: 10.1039/b917571g

Recently, simplified graphical modeling approaches based on low-order conditional (in-)dependence calculations have received attention because of their potential to model gene regulatory networks. Such methods are able to reconstruct large-scale gene networks with a small number of experimental measurements, at minimal computational cost. However, unlike Bayesian networks, current low-order graphical models provide no means to distinguish between cause and effect in gene regulatory relationships. To address this problem, we developed a low-order constraint-based algorithm for gene regulatory network inference. The method is capable of inferring causal directions using limited-order conditional independence tests and provides a computationally-feasible way to analyze high-dimensional datasets while maintaining high reliability. To assess the performance of our algorithm, we compared it to several existing graphical models: relevance networks; graphical Gaussian models; ARACNE; Bayesian networks; and the classical constraint-based algorithm, using realistic synthetic datasets. Furthermore, we applied our algorithm to real microarray data from *Escherichia coli* Affymetrix arrays and validated the results by comparison to known regulatory interactions collected in RegulonDB. The algorithm was found to be both effective and efficient at reconstructing gene regulatory networks from microarray data.

Introduction

Massive accumulation of genome-wide gene expression data for many organisms presents an opportunity and a challenge to elucidate gene regulatory networks (GRNs) controlling various biological processes. Graphical models[1] are probabilistic tools to analyze and visualize conditional dependencies between random variables, and have the potential to identify, systematically, transcriptional regulatory interactions from a compendium of microarray expression profiles. Such models include relevance networks (RNs),[2] graphical Gaussian models (GGMs),[3] low-order conditional dependence models[4–7] and Bayesian networks (BNs).[8] BNs are capable of identifying non-linear causal relationships between genes using statistical methods. The causal relationships derived from this approach can portray information embedded in microarray data in a manner that is intuitive and familiar to biologists. Generally, a BN is a graphical representation of the relationship (dependence) between multiple interacting entities. This graphical representation is more commonly called a directed acyclic graph (DAG). The nodes or vertices of a DAG represent the random variables in the network, *e.g.* genes, while the edges connecting the vertices represent the causal influence of one node on another. BN-based GRN inference involves searching through multiple possible DAGs for the one that best represents the observed data. This task is also called BN learning. However, in most typical microarray experiments, the number of genes analyzed far exceeds the number of distinct expression measurements. This situation challenges BNs both conceptually and computationally. The main problem with BN learning is that the number of the possible DAGs increases super-exponentially with the number of nodes (genes) in the network, and thus only a small subset of all possible DAGs can be tested. More importantly, an inaccurate estimation of conditional dependencies leads to a high rate of false positive and false negative relationships in the final results. An interpretation of the BN graph within the Markov framework (see details in the next section) is rather difficult. Gene expression databases typically contain measurements for thousands of genes, but most existing algorithms for learning BNs do not scale to such high-dimensional datasets. There are some exceptions,[9,10] which use hybrid approaches to improve computational efficiency. However, the often-exercised discretization they employ leads to information loss, which can influence considerably the results obtained. This is corroborated by a previous study[11] on the popular BN method, which demonstrated that this approach tends to perform poorly on microarray data.

[a] *Plant Biology Division, The Samuel Roberts Noble Foundation, Inc., 2510 Sam Noble Parkway, Ardmore, OK 73401, USA. E-mail: mudvardi@noble.org; Fax: +1 580 224 6692; Tel: +1 580 224 6655*
[b] *Genetics & Developmental Biology Program, Plant & Soil Sciences Division, West Virginia University, 2090 Agricultural Sciences Building, Morgantown, WV 26506, USA*

† Electronic supplementary information (ESI) available: Additional information. The software, datasets and three supplementary files can be downloaded from http://bioinfo.noble.org/manuscript-support/lpc/. See DOI: 10.1039/b917571g

To circumvent these problems, simplified graphical models, such as RNs, GGMs and low-order conditional (in-)dependence models, have received attention for practical use.[12–14] RNs (or co-expression networks) are the simplest approach, and are constructed by computing a similarity score for each pair of genes, *e.g.*, the correlation or mutual information between expression profiles. If similarity is above a certain threshold, the pair of genes is connected in the graph, if not, it remains unconnected. RNs are relatively easy to calculate and reasonably accurate, even when the number of genes is much larger than the number of samples. The results from RNs agree well with functional similarity,[15] and many co-expression relationships are conserved over evolution supporting the conclusion that they represent biologically-meaningful networks.[2] However, RNs contain only limited information about the underlying biological mechanisms since the effect of other genes on the relationship between two genes is ignored. For example, from similarity of expression profiles alone, we cannot distinguish between direct and indirect relationships. In contrast, GGMs can identify a direct correlation between two genes after accounting for the impact of all other genes in the model. In this mode, each gene pair is tested for conditional independence (CI) given the data from all other genes. From these tests, one can tell if the correlation between two genes is direct or mediated through other genes. A problem with GGMs is that full conditional models are hard to estimate if the number of samples is small compared to the number of genes.[3,16] Low-order conditional dependence models represent a compromise between RNs and GGMs and are capable of identifying direct and indirect correlations between any two genes after correcting for the influence of a third gene only. Thus, in contrast to GGMs, low-order conditional independence models do not consider the effects of all other genes on the correlation between any two genes. This facilitates the study of dependence patterns in a more complex and exhaustive way than with only pair-wise correlation-based relationships (*i.e.* RNs), while maintaining high accuracy even from few observations. In this approach, modeling is limited to 0-1 order conditional independencies (thus also called 0-1 graphs[5]). This simplification avoids the need to carry out statistically unreliable and computationally costly searches for conditional independence in large subsets.

In contrast to BN models, the output from most simplified graphical models contains undirected edges between nodes/variables and provides no means to distinguish between response variables and covariates and, thus, between cause and effect. This makes it difficult for biologists to discern regulatory relationships between genes. To redress this difficulty, we revisited basic concepts used in constraint-based algorithms,[17] an important offshoot of BN learning methods, in which dependencies and conditional dependencies are tested in the data and directed graphs are built accordingly. The PC-algorithm (after its authors Peter and Clark) proposed in ref. 17 is a well-known example. However, for the PC-algorithm, in the worst case, all possible combinations of the conditioning set need to be examined which would require an exponential number of tests. Consequently, it is hard to apply the PC-algorithm to large gene expression datasets. Therefore, we developed an algorithm that estimates causal relationships based on a low-order constraint-based approach, in which low-order CI tests rather than full-order CI tests are required. The algorithm has high computational efficiency but still finds most causal relationships.

Methods

Definitions and preliminaries

To illustrate our new algorithm, some formal notions, definitions and assumptions are needed, which can be found in most books on BNs.[18]

Let $\mathbf{V}$ denote a non-empty finite set of random variables. A *Bayesian network* (BN) for $\mathbf{V}$ is defined by a pair $\langle G,\mathbf{\Theta}\rangle$. The structural model is a *directed acyclic graph* (DAG) $G = (\mathbf{V},\mathbf{E})$, in which nodes represent variables in $\mathbf{V}$ (in BN, variable and node can then be used interchangeably) and the set of edges $\mathbf{E}$ is all edges between nodes in $\mathbf{V}$. We use the notation $X \rightarrow Y$ if and only if there is a directed edge between two nodes X and Y, and X–Y if and only if there is an undirected edge between X and Y. The *parents* of a node X (written **Parents**(G,X)) is the set of nodes that have directed edges to X. The *adjacency* set of a node X in graph G, denoted by **Adjacencies**(G,X), are all nodes that are directly connected to X by an edge. The elements of **Adjacencies**(G,X) are also called *neighbors* of X or *adjacent* to X. We call the set of edges connecting the k nodes a *path* from X_1 to X_k. Y is called a *descendant* of X, and X is called an *ancestor* of Y is there is a path from X to Y, and Y is called a *non-descendant* of X if Y is not a descendant of X. For each node there is a probability distribution at that node given the state of its parents in G, denoted by $P(X|\mathbf{Parents}(G,X))$. $\mathbf{\Theta}$ are parameters specifying all these probabilities. BNs follow the *Markov condition*, stating that given its parents each variable is independent of its *non-descendants*. Under the Markov assumption, each BN specifies a decomposition of the joint distribution over all distributions of the nodes, in a unique way: $P(\mathbf{V}) = \Pi_{X\in\mathbf{V}}\, P(X|\mathbf{Parents}(G,X))$.

It is necessary to give a brief description of the *conditional independence* (CI) relation. X and Y are said to be conditionally independent given $\mathbf{S}$ (where $X \in \mathbf{V}$, $Y \in \mathbf{V}$ and $\mathbf{S} \subseteq \mathbf{V}\backslash\{X,Y\}$) if $P(\mathbf{S}) \neq 0$ and one of the following holds: (1) $P(X|Y,\mathbf{S}) = P(X|\mathbf{S})$ and $P(X|\mathbf{S}) \neq 0$, $P(Y|\mathbf{S}) \neq 0$; (2) $P(X|\mathbf{S}) = 0$ or $P(Y|\mathbf{S}) = 0$. This CI relation is denoted by $I(X,Y|\mathbf{S})$. A CI relation is characterized by its *order*, which is simply the number of variables in the conditioning set $\mathbf{S}$.

A criterion called *d*-separation captures exactly the CI relationships that are implied by the Markov condition. We say X and Y are *d*-separated by a node set $\mathbf{S} \subseteq \mathbf{V}\backslash\{X,Y\}$ in G if every path between X and Y is *blocked* by $\mathbf{S}$. A path between X and Y is *blocked* by $\mathbf{S}$ if one of the following holds: (1) $W \in \mathbf{S}$ and W does not have converging arrows along the path between X and Y, or (2) W has converging arrows along the path and neither W nor any of its descendants are in $\mathbf{S}$. Here, we say a node W has *converging arrows* along a path if two edges on the path point to W. A probability distribution $\mathbf{\Theta}$ on $\mathbf{V}$ is said to be *faithful* with respect to a graph G if conditional independencies of the distribution can be inferred from so-called *d*-separation in the graph G and vice-versa. More precisely, faithfulness of $\mathbf{\Theta}$ with respect to G means: for any $X,Y \in \mathbf{V}$ with $X \neq Y$ and any set $\mathbf{S} \subseteq \mathbf{V}\backslash\{X,Y\}$, X and Y are

conditionally independent given **S** if and only if node X and node Y are d-separated by the set **S**.

The nodes W, X and Y form a *v-structure* in a DAG G when $X \rightarrow W \leftarrow Y$ is the subgraph of G induced by W, X and Y. Two DAGs are *equivalent* if and when they represent the same d-separation statements. The *equivalence class* of a DAG G is the set of DAGs that are equivalent to G. Even given an infinite number of observations, we cannot distinguish among the different DAGs of an equivalence class. Using published results,[19] we can characterize equivalent classes more precisely: two DAGs are equivalent if, and only if, they have the same skeleton and the same v-structures. The *skeleton* of any DAG is the undirected graph resulting from ignoring the directionality of every edge. A common tool for visualizing equivalence classes of DAGs is a *partial directed acyclic graph* (PDAG), which is a graph that contains both directed and undirected edges. There may be more than one PDAG that correspond to the same equivalence class because extra undirected edges can be oriented sometimes. Thus, *completed PDAG* (CPDAG) is proposed to represent an equivalence class uniquely.[20] The CPDAG corresponding to an equivalence class is the PDAG consisting of a direct edge for every compelled edge in the equivalence class, and an undirected edge for every reversible edge in the equivalence class. A directed edge $X \rightarrow Y$ is *compelled* in G if for every DAG G' equivalent to G, $X \rightarrow Y$ exists in G'. CPDAGs are also called *maximally oriented graphs*.[21] Several orientation rules[21] can be used to generate a CPDAG. The connections (edges) in a BN can be used to interpret *causal relationships* between nodes.[8]

Algorithm

In this section, we present a new algorithm for learning a PDAG from a database D with n nodes and m cases (called sample size). In our case, D represents a microarray dataset with n genes and m measurements/chips. The algorithm adopts similar procedures to those used in the classical PC-algorithm but requires only low-order CI tests and is, therefore, named a low-order PC-algorithm, or LPC. This LPC-algorithm consists of two phases: CI tests and an orientation phase. Only low-order CI tests are performed in our algorithm because their results are more reliable than higher-order tests.[17] In the CI definition, to test $P(X|\mathbf{S})$, if there are many variables in the conditioning set **S**, there may be very few examples in the dataset that satisfy a particular value assignment for **S**, and $P(X|\mathbf{S})$ may be inaccurate if there is noise in the examples. Similar issues may occur for $P(X|Y,\mathbf{S})$. Thus, the high-order CI tests are hard to estimate if the sample size is small. The other benefit of low-order tests is restrained computational complexity. To guarantee that causal relationships inferred by this algorithm are correct, we perform extra tests in the orientation phase. The formal pseudocode of the LPC-algorithm is presented in Appendix A (Table 2). In this algorithm, it receives a dataset D, significance level ε, and maximal order k of CI tests as input, and returns a PDAG as output.

In the first phase, G is initiated as a fully connected undirected graph. For example, suppose we have a simple DAG with only five nodes (Fig. 1a), a complete graph is produced (Fig. 1b) in the first step of our algorithm. Then, iterative CI tests are performed for each connected node pair given a node subset **S** taken from neighbor nodes of the connected node pairs. Under the DAG faithful assumption, correlations or non-correlations, direct or indirect correlations between node pairs can be distinguished by CI tests. In this procedure, we used $\boldsymbol{dep_D}(X,Y|\mathbf{S})$ as a measure of the strength of the conditional dependence between X and Y given **S** with respect to D. In order to decide if $I(X,Y|\mathbf{S})$ is true or not, $\boldsymbol{dep_D}(X,Y|\mathbf{S})$ runs a partial correlation coefficient calculation when D is continuous and then uses ε as the significance level. In our algorithm, the partial correlation coefficient calculation follows the method previously used[17,22] and is described in Appendix C.

For a given connected node pair X, Y in Fig. 1b, the conditioning set is taken from neighbors of X or Y, *i.e.*, any subset **S** (**S** $\subseteq$ **Adjacencies** $(G,X)\backslash\{Y\}$). The size of **S** ($|\mathbf{S}|$) is the

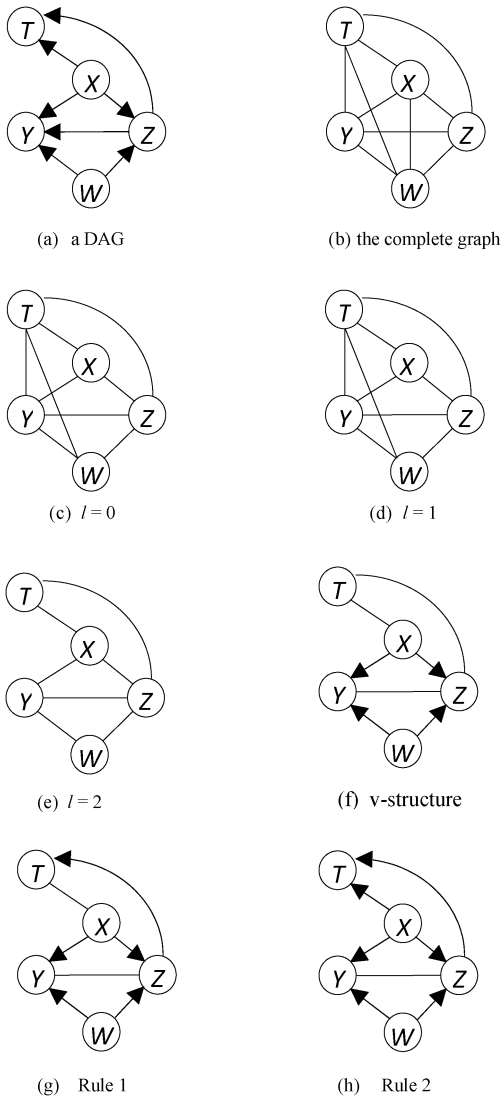

Fig. 1 Given relations in the original DAG (a), the LPC-algorithm starts with a complete graph (b), then creates undirected graphs after 0-order CI tests (c), 1st-order CI tests (d), and 2nd-order CI tests (e) in the first phase, and infers directions (f)–(h) using the orientation rules in the second phase.

order of CI tests and is controlled by l in our algorithm. Thus, in this case, 0-order CI test $\boldsymbol{dep_D}(T,Y|\emptyset)$ is performed when $l = 0$. The 1st-order CI tests $\boldsymbol{dep_D}(T,Y|X)$, $\boldsymbol{dep_D}(T,Y|W)$ and $\boldsymbol{dep_D}(T,Y|Z)$ are performed when $l = 1$. For 2nd-order CI tests ($l = 2$), $\boldsymbol{dep_D}(T,Y|W,Z)$, $\boldsymbol{dep_D}(T,Y|W,X)$ and $\boldsymbol{dep_D}(T,Y|X,Z)$ are performed. All the connected node pairs in G are checked given l (lines 3–12 of Table 2). The order of CI tests is iteratively increased from 0 to the maximal order k. In this procedure, if a connected node pair T and Y are conditionally independent given $\mathbf{S}$, then the edge between T and Y will be removed and $\mathbf{S}$ will be recorded into **Sepset**, which is used to store the conditioning set. From our example in Fig. 1, suppose we set $k = 2$, from d-separations given in Fig. 1a, X–W can be broken from 0-order tests because $I(X,W|\emptyset)$. No link can be broken from 1st-order CI tests and thus Fig. 1d is unchanged. $\{X,Z\}$ blocks each path between T and W and each path between T and Y, that is, $I(T,W|\{X,Z\})$ and $I(T,Y|\{X,Z\})$. Thus, T–W and T–Y are removed after 2nd-order CI tests (Fig. 1e), and the conditioning sets are saved (*i.e.*, $\mathbf{Sepset}(T,W) = \{X,Z\}$ and $\mathbf{Sepset}(T,Y) = \{X,Z\}$). This procedure is repeated until the order of CI tests l is increased to k or no more conditioning sets can be found (lines 2–14 of Table 2). The highest order of CI tests is limited by k specified by the user, thus avoiding an exponential increase in the number of CI tests with the number of neighbors. After the first phase, the skeleton of a PDAG (Fig. 1e in our example) is generated and the weights of all edges are assigned the minimal conditional dependence values for each node pair calculated from all the CI tests.

In the second phase, orientation rules are applied to orient the graph skeleton. First, the v-structures are determined (lines 15–19 of Table 2) for triple nodes X, Y and Z, if X and Y, and Y and Z are connected while X and Z are not connected and $Y \notin \mathbf{Sepset}(X,Z)$, then we can infer directionality $X \rightarrow Y \leftarrow Z$ (any one of the three alternatives $X \rightarrow Y \rightarrow Z$, $X \leftarrow Y \leftarrow Z$ and $X \leftarrow Y \rightarrow Z$ will lead to $I(X,Z|Y)$ and $Y \in \mathbf{Sepset}(X,Z)$, and thus cause a contradiction). In our example, we have $I(X,W|\emptyset)$, $Z \notin \emptyset$ and $\min(|\mathbf{Adjacencies}(G,X)\backslash\{Z\}|, \mathbf{Adjacencies}(G,Z)\backslash\{X\}|) = 2$ and $\min(|\mathbf{Adjacencies}(G,W)\backslash\{Z\}|, \mathbf{Adjacencies}(G,Z)\backslash\{W\}|) = 1$, thus $X \rightarrow Z$ and $W \rightarrow Z$ can be inferred according to the v-structure rule in our algorithm. Similarly, $X \rightarrow Y$ and $W \rightarrow Y$ can be inferred. The other four orientation rules are given in lines 21–24 of the LPC-algorithm (Table 2). Next, the orientation rules (**R1**–**R4**) are repeatedly used (lines 20–25 of Table 2) to determine the directions of undirected edges until no more edges can be oriented. The basic idea is to make sure that all other undirected edges also can be oriented based on the DAG assumption. The details can be found in Appendix A (lines 21–24 of Table 2). According to the orientation rule 1 (line 21 of Table 2), $Z \rightarrow T$ can be determined (Fig. 1g) because $W \rightarrow Z$ and Z–T while W and T are not connected and $|\mathbf{Adjacencies}(G,T)\backslash\{Z\}| = 1$ ($\leq k$). According to the orientation rule 2 (line 22 of Table 2), by satisfying $X \rightarrow Z$ and $Z \rightarrow T$ and $\min(|\mathbf{Adjacencies}(G,X)\backslash\{T\}|, \mathbf{Adjacencies}(G,T)\backslash\{X\}|) = 1(\leq k)$, we can infer $X \rightarrow T$. No other rules can be applied in this case. Fig. 1h is the final structure inferred from the LPC-algorithm.

In the second phase, the key point of the LPC-algorithm is that the neighbor number for linked node pairs is checked before applying each orientation rule, *i.e.*, $\min(|\mathbf{Adjacencies}(G,X)\backslash\{Y\}|, \mathbf{Adjacencies}(G,Y)\backslash\{X\}|) \leq k$ given the node pair between X and Y must be satisfied. This is a non-trivial step that must be completed before applying the orientation rules in the second phase. The goal of this step is to ensure that the orientation rules in the second phase are still correct when only 0–k low-order CI tests are performed in the first phase. Theoretical proofs of soundness are provided in Appendix B. Thus, we can extend the use of the classical constraint-based algorithm to datasets with large variable numbers because only polynomial runtime is used. The number of CI tests in the first phase is bounded by $O((n^4 - 4n^3 + 7n^2 - 4)/4)$ in the worst case when $k = 2$ and time efficiency is maintained. This is particularly important for microarray data when the number of genes is much higher than the number of samples because conditional dependencies are generally hard to estimate with only limited samples and high orders.[17,22]

Because only low-order CI tests are performed, some false edges may be retained in the skeleton and some directions are missed. This is the price that the LPC-algorithm pays to reduce runtime. Note that if k is increased to $n - 2$, the LPC-algorithm is equivalent to the classical PC-algorithm. Thus, the LPC-algorithm can be viewed as a special case of the PC-algorithm.

Results

Because we have insufficient knowledge to construct complete, large-scale models of real gene regulatory networks, it is necessary to employ simulated datasets with known network structures to test novel algorithms.

Synthetic data

To assess the performances of our algorithm, we compared it to several existing methods, using well-defined synthetic datasets. We followed previous studies[4,23] to generate hypothetical or synthetic data, using the reaction kinetics-based system of coupled non-linear continuous time ODEs introduced in ref. 24. Gene expression levels are taken as state variables x_i, $i = 1, \ldots, n$. The influence on the transcription of each gene due to all other genes is described by the $n \times n$ (sparse) matrix of adjacencies G, and the rate law for mRNA synthesis of a gene is obtained by multiplying together the sigmoidal-like contributions of the genes identified as its inhibitors and activators. Consider the i-th column of G, $i = 1, \ldots, n$ and choose randomly a sign to its non-zero indices in this column. Denote by $j_1, \ldots, j_a$ the indices with assigned positive values (activators of the i-th gene) and by $k_1, \ldots, k_b$ the negative ones (inhibitors of the i-th gene) in the i-th column of G. The ODE for x_i is then

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = V_i \prod_{j\in\{j_1,\ldots,j_a\}} \left(1 + \frac{A_j^{h_j}}{A_j^{h_j} + \theta_j^{h_j}}\right) \prod_{k\in\{k_1,\ldots,k_b\}} \left(\frac{\theta_k^{h_k}}{I_k^{h_k} + \theta_k^{h_k}}\right) - \lambda_i x_i$$

where the activator A_j and inhibitor I_k act independently of each other. V_i is a basal rate of transcription, *i.e.*, when there is no action of inhibitors or activators. The constants θ_j and θ_k represent concentrations at which the effect of the activator or

inhibitor is half of its saturating value. The exponents h_j and h_k are the activation and inhibition Hill coefficients and λ_i the degradation rate constants. We set θ_j and θ_k as 100, V_i, h_j and h_k as 1, λ_i as 0.1.[23]

For fairness of directed graph comparisons under BN frameworks, we used DAGs as the network structures. It will not affect the conclusions for the evaluation of undirected graphs if other structures are chosen (such as scale-free networks—see results in the ESI†). We generated 10 DAGs G with 100 nodes, which have different topology structures but have a fixed edge number of 500. The maximal degree of a node in these DAGs is 10. For each G, seven different datasets with different sample sizes (10, 20, 50, 100, 200, 500 and 1000) were generated separately using the above formula. To further evaluate the performances of our algorithm under other network structures, we also generated 10 scale-free networks with feedback loops. The test results for scale-free networks are presented in the ESI†.

Evaluation

In order to evaluate our algorithm, we compared it to several others that are commonly used, including GGMs, BNs, PC, RNs and ARACNE[4] (an algorithm based on low-order CI tests). The implementation of GGM, the BN learning method, PC and ARACNE were taken from ref. 3, 4, 22, and 25 respectively. We chose the BN learning method used in ref. 25 because it can accept continuous data and is a scoring and searching method,[18] an offshoot of BN learning methods that serves as a good reference for our constraint-based algorithm. While the true network is a directed graph, the competitive methods may lead to undirected, directed, or partially directed graphs. To assess the performance of our algorithm, we applied two criteria in different approaches. The first approach, referred to as the undirected graph evaluation, discarded information about the edge direction. The second approach, referred to as the directed graph evaluation, compared the predicted structures with original directed graphs for LPC, PC and BN because only these three methods can output edge directions. For these comparisons, we ran LPC, PC, BN, RN GGM and ARACNE over 70 synthetic datasets consisting of 10 DAGs with 7 different sample sizes, generated as described above. For the LPC and PC algorithms, we set $\alpha = 0.05$ for CI tests (see Appendix B) because it achieved the best performance.

Evaluation of undirected graphs

Although the main goal of the LPC-algorithm is to identify a directed network, the skeleton itself already contains interesting information. Hence, initially, we compared the LPC-algorithm to other methods without directions. Since genetic networks are sparse, *i.e.*, most gene pairs are not connected by a direct regulatory link, the numbers of true negative (TN) instances (N_{TN}) far exceed the true positive (TP) numbers (N_{TP}). Traditional receiver operator characteristic (ROC) analysis is insensitive to the false positive rate (1-specificity: $N_{FP}/(N_{FP} + N_{TN})$) and, therefore, inappropriate for the final area-under-the-curve (AUC) calculation. Therefore, we adopted the Precision *versus* Recall (PvsR) curves[4] and the TP number with fixed FP numbers[25] as measures of the quality of network reconstruction. For a PvsR curve, precision ($N_{TP}/(N_{TP} + N_{FP})$) is plotted against the recall ($N_{TP}/(N_{TP} + N_{FN})$), where precision and recall are computed over a range of pruning thresholds, then AUC value is obtained as a measurement score, with higher scores indicating better performance. Each of the five reverse engineering methods compared in this study generates a matrix of scores associated with the edges in a network. These scores are of different nature: absolute values of Pearson correlation coefficients for RNs, partial correlation coefficients for GGMs, LPC and PC, marginal posterior probabilities for BNs and conditional mutual information for ARACNE. However, all these scores define a ranking of edges. We compared each weight matrix with the real adjacency matrix DAGs under different thresholds. The mean values of AUC(PvsR) and TP for a fixed acceptable FP (here 20) run over 10 datasets under equal conditions at same sample sizes are presented in Fig. 2a and b.

From the AUC(PvsR) evaluation (Fig. 2a), the performances of LPC and PC were better than other methods at sample sizes of 100 and 1000, while GGM outperformed all other algorithms at other sample sizes. The average AUC(PvsR)s were low when sample sizes were only 10, 20 and 50, meaning that very few TPs were captured in such cases. In terms of AUC(PvsR)s, the performances of LPC, PC and BN improved steadily with increasing sample sizes, whereas, AUC(PvsR)s of RN stabilized and of GGM decreased with sample sizes above 200. This indicates that simple graphical models (such as RNs) can detect connections well, and better than the sophisticated models LPC and PC at very low sample size but their performance improves relatively little when sample sizes increase. GGM achieved best performance overall in the AUC(PvsR) evaluation. This can be attributed to a small sample stable estimation procedure used in GGM for genomic data with small sample sizes.[3]

From TP for fixed FP tests (Fig. 2b), the LPC-algorithm was superior to RN, GGM, ARACNE and BN at almost all sample sizes and was also slightly better than PC. Overall, LPC and PC outperformed RN, ARACNE and BN in this test. For ARACNE, the major reason for its poor performance was the non-linear similarity measures employed, which were less precise for this simulated data. This is also corroborated by a previous study.[23]

Evaluation of directed graphs

LPC, PC and BNs, yield directed graphs and we compared their predicted graphs with true graphs to evaluate the quality of structure prediction. We adopted the structure hamming distance (SHD) metric[10] to evaluate the predictive powers of these methods. Briefly, the SHD counts the number of edge insertions, deletions, and flips required to convert the estimated PDAG into the correct representation (CPDAG) of the original DAG.[20] Thus, a large SHD indicates a poor fit, while a small SHD indicates a good fit. SHD comparisons are presented in Fig. 2c, while the distribution of structure errors, such as the numbers of missing, extra, and flipped edges are shown in Fig. 2d–f.

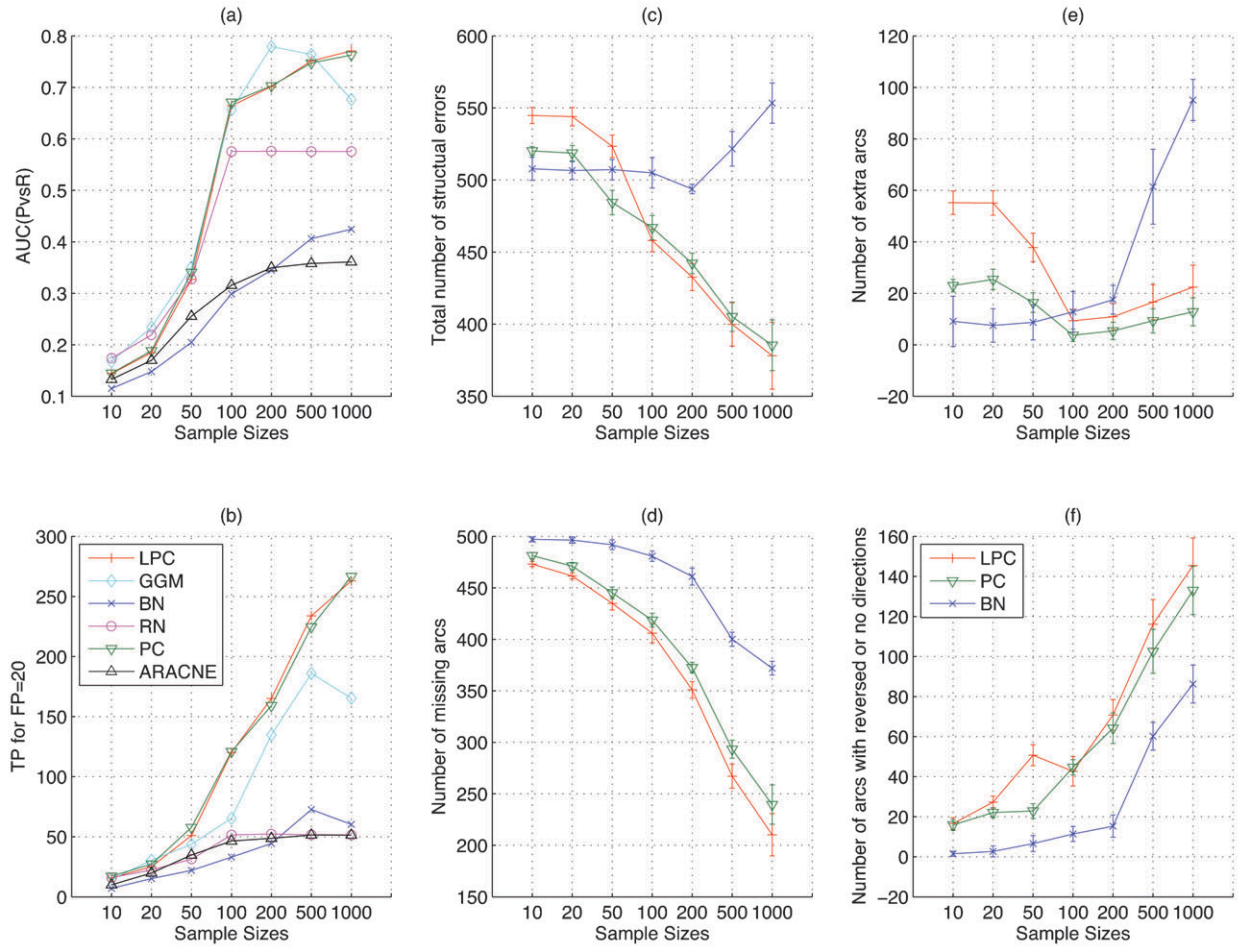

Fig. 2 The performance comparisons between several methods. For (a) and (b), the average AUC(PvsR)s and true positives under the fixed 20 false positives were plotted for LPC, GGM, BN, RN, PC and ARACNE under 7 different sample sizes. For (c)–(f), the predictive errors comparing between LPC, PC and BN. The average errors over 10 datasets under 7 sample sizes: (c) total SHDs; (d) missing edges; (e) extra edges; (f) reversed edges or missing directions.

In terms of SHD metrics, the LPC-algorithm performed similarly to the PC-algorithm and much better than BN. LPC outperforms PC for sample sizes greater or equal to 100 while it performs worse than PC for sample sizes smaller than 100 (Fig. 2c). The major prediction errors for both methods were from missing edges, the numbers of which varied from ~480 to 220 at different sample sizes. This indicates that some associations are difficult to detect by partial correlation calculations. The LPC-algorithm produced more spurious edges than the PC-algorithm (Fig. 2e) and fewer missing edges (Fig. 2d), presumably because higher-order CI tests were not performed in LPC. PC is slightly better than LPC in direction prediction (Fig. 2f), as more directions are missed when only limited-order CI tests are performed as in LPC. Overall, however, prediction accuracy of LPC was similar to that of PC (Fig. 2c).

In both tests described above, the BN learning method performed poorly compared to other methods as most edges were not be recovered by this method (Fig. 2d and e). Apart from the reasons mentioned above, we presume that the implementation of the BN learning method is very restrictive for some data types because it is based on the multivariate Gaussian assumption.[25,26] Scoring and searching methods for continuous data are further stumbling blocks for the BN community because they are computationally expensive and lack the causality-related theoretical correctness guarantee.[27]

Effects of order on the LPC-algorithm

We evaluated the effect of the order k chosen in the LPC-algorithm on computational expense and quality as follows. The number of CI tests performed is indicative of the computational effort spent for this algorithm. Using simulated datasets with varying sample sizes, we calculated the average number of CI tests performed in LPC when k varied from 0 to 9 or the highest order reached in the algorithm at each sample size (Fig. 3a). The average numbers of CI tests *versus* the order k for different sample sizes (SS in short) are plotted in Fig. 3a. The error bars represent one standard deviation across 10 datasets at each sample size. For SS = 10 and 20, the highest orders reached in the LPC-algorithm are only 4 and the numbers of CI tests are much lower than other sample sizes. The numbers of CI tests

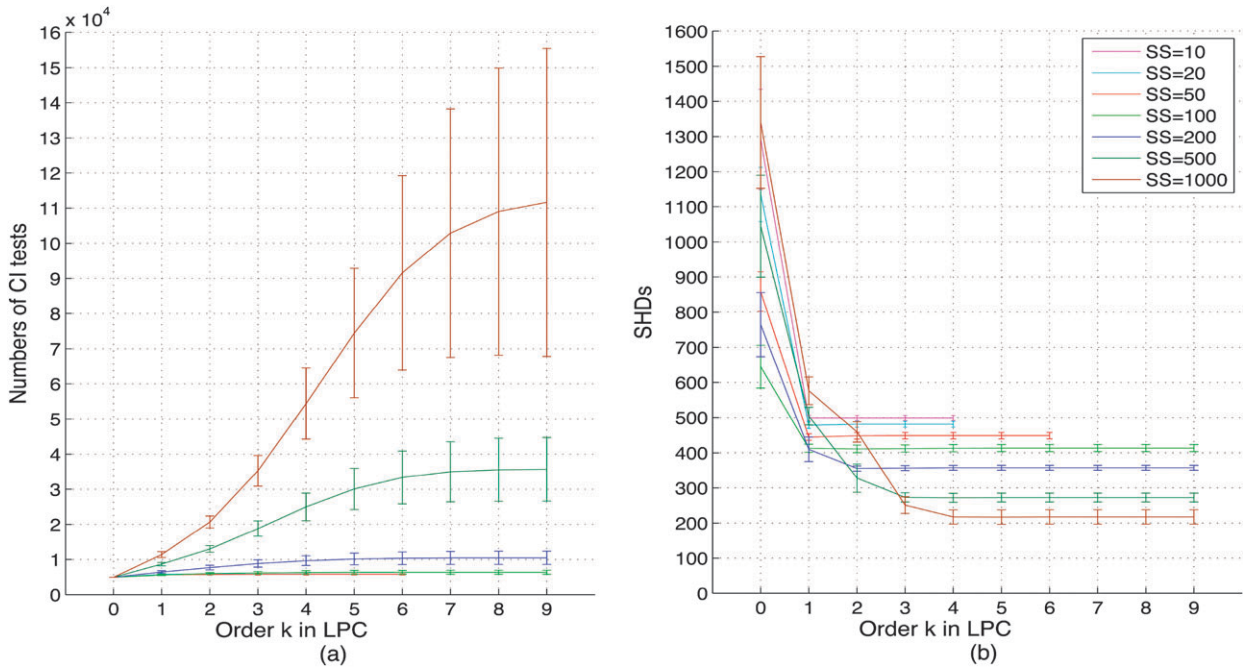

Fig. 3 The average numbers of CI tests performed in the LPC-algorithm over 10 independent DAGs when the order k is varied from 0 to 9. The average numbers and standard deviations (error bars) are shown in (a). The average SHDs and their standard deviations from the LPC-algorithm *versus* the order k are plotted in (b). Each sample size (SS) is depicted by one color curve (see legend) in these two sub-figures.

are basically unchanged after k is above 2 for SS = 10 and 20, thus, the curves for those sample sizes (SS = 10, 20 and 50) are almost flat and indistinguishable in Fig. 3a because they overlap. The highest order varied according to sample sizes, indicating that increased sample sizes increase sensitivities and help identify more hypothetical neighbors for each node. For datasets with low sample sizes (10, 20, 50 and 100), there was little increase in the number of CI tests with increasing k (Fig. 3a). However, CI test numbers increased substantially with increasing order k at big sample sizes (200, 500 and 1000). Setting $k = 2$ reduced the numbers of CI tests by 26.2%, 63.5% and 81.5% for datasets with sample sizes of 200, 500 and 1000, respectively. Clearly, low-order tests can reduce computational time significantly for sample sizes greater than 200.

To confirm the time efficiency of our LPC-algorithm, runtime comparisons were done on a Pentium Xeons, 2.33 GHz and 4 GB RAM running Windows XP. The average processor times for estimating the CPDAG under different sample sizes are given in Table 1. For every sample size, the LPC-algorithm ran faster than PC on average. The difference increased with higher sample sizes. The explanation for this is that the relative number of CI tests is reduced in our algorithm, compared to PC, at higher sample sizes (Fig. 3a and Table 1), while this is not the case for the PC-algorithm, which has no limit on k. We do not list runtime for the BN learning method because it is much slower than LPC and PC. The BN method required more than 2 days to complete the same tests for each sample size.

Table 1 Average processor time for estimating graph structures (in seconds, with standard errors in brackets at each sample size). Maximal k in PC means the highest order of CI tests performed in the PC-algorithm

Sample size	LPC ($k = 2$)	PC	Maximal k in PC
10	1.11(0.38)	1.39(0.07)	4
20	1.09(0.41)	1.41(0.06)	4
50	1.23(0.46)	1.41(0.07)	6
100	1.46(0.45)	1.78(0.65)	9
200	2.74(0.50)	11.24(15.11)	10
500	36.7(38.31)	138.99(106.99)	12
1000	641.10(757.97)	1430.92(1380.84)	14

Notice that this test was based on moderate-sized networks of 100 nodes each. Many more CI tests would be spared by our LPC-algorithm compared to the PC-algorithm when networks are scaled up to several thousand nodes. However, due to the computational complexity, it was not feasible to perform these tests using the PC-algorithm (or set k values high in the LPC-algorithm) for networks having over 1000 nodes.

We also examined the effect on performance of limited (0–k) order CI tests in LPC. We compared average SHD measurements at different orders, *i.e.*, k is varied from 0 to 9 (Fig. 3b). From Fig. 3b, we can see that the average SHDs are decreased considerably after initial low-order CI tests (when k is varied from 0 to 4 at most). For SS = 10, 20, 50 and 100, the SHD remains constant after $k > 2$. Even for the sample size 1000, the SHD are not improved further for $k > 4$. This shows that the major performance improvement of LPC comes from low-order CI tests. SHD values decreased with increasing order, although performance improved only from order $k = 1$–3. Little was gained by increasing to higher-order ($k > 4$) CI tests at all sample sizes. Therefore, limitation of our algorithm to order 2 or 3 is justified on the grounds of greater efficiency.

Escherichia coli network inference

To test the ability of LPC to scale up to thousands of genes in a dataset, we applied our algorithm to real microarray data. We downloaded *E. coli* gene expression data from M3D (http://m3d.bu.edu/norm/). This dataset consists of 445 arrays from 13 different collections corresponding to various treatments, including different growth media, environmental stresses (*e.g.* DNA damaging drugs, pH changes), genetic perturbations, and growth phases. The experiments were all carried out using Affymetrix GeneChip *E. coli* Antisense Genome arrays, containing 4345 gene probe-sets. Since transcription factors (TFs) are the primary regulators of gene expression in a cell, we used 328 validated or computationally-predicted transcription factors (TFs) as prior knowledge, and only links between these 328 TFs and all other genes (including TF–TF links) were allowed for the initial graph pattern used by the LPC-algorithm, instead of the fully connected graph. We set $k = 3$ and $\alpha = 0.001$ for this inference.

We used the *E. coli* K12 transcriptional network compiled in the RegulonDB database version 6.3[28] as the 'true' network, from which we derived a directed graph of 3071 interactions. We then compared our predicted gene interactions to the true network. All 16 884 gene pair edges resulting from LPC analysis were oriented (see results in the ESI†). Importantly, only 2.9% of gene pairs were directed from non-TF to TF genes. In other words, 97.1% of all predicted relationships were directed from TF genes to other genes, including other TF genes, as expected. Among all the predicted gene interactions, 208 are validated by known regulatory relations listed in RegulonDB. Of these, 199 TFs were predicted correctly to regulate known target genes, while the direction of interaction between TFs and known target genes was incorrectly predicted in only 9 cases.

We compared the performance of LPC with CLR, a state-of-the-art method used for gene network reconstruction that performed well in a large-scale benchmark evaluation.[29] When we selected the same number of gene pairs (16 884) from CLR analysis, based on top-ranking, 209 true interactions were recovered. Therefore, the performance of LPC (208 true interactions in our case) was comparable to that of CLR in terms of undirected graph prediction. However, CLR can only predict an undirected network while our LPC-algorithm can also infer the direction of edges connecting nodes (genes). We did not compare our results to those of two recent studies[30,31] using the same *E. coli* dataset because these previous studies used (semi-) supervised learning methods to infer TF–gene regulations. In other words, additional prior data were used to train the prediction models, making comparisons with our method of unsupervised learning unfair. It is important to note that the level of prior knowledge for *E. coli* is unmatched in most other species, which would severely handicap supervised approaches in more complex or less-studied species.

Discussion

The motivation for this work was to develop a time-efficient algorithm for GRN reconstruction from genome-wide transcript data, as a basis for hypothesis generation and experimental testing. Theoretical proofs and the results of evaluation tests show that our LPC-algorithm is useful for network inference. This new algorithm is computationally scalable to high-dimensional datasets, does not require discretization of transcript levels, and can predict causal relationships. The algorithm is a good alternative to BN learning methods for large-scale GRN reconstruction.

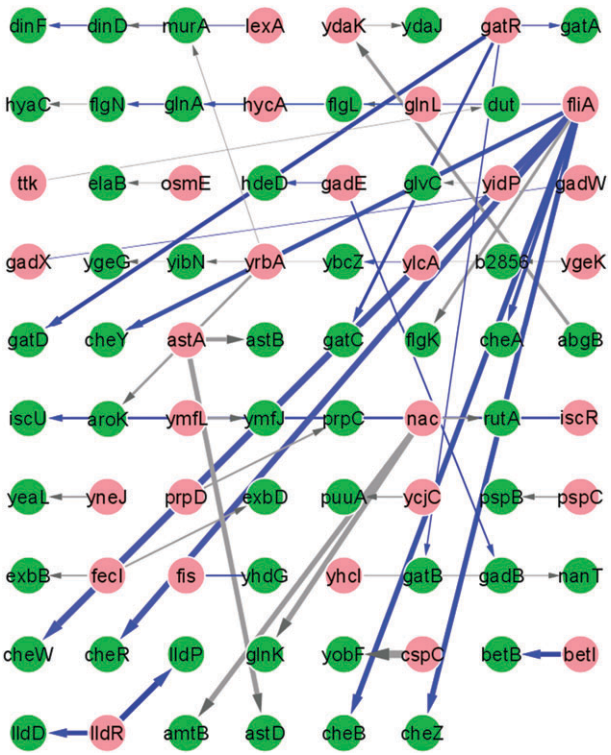

Fig. 4 The top 50 transcriptional regulatory relationships inferred by LPC. Directed edges that have experimental support in RegulonDB release 6.3 are indicated by blue arrows, edges that are not yet supported by data in RegulonDB are depicted in grey, and known edges with incorrectly predicted directions are depicted as undirected blue lines. TFs are marked by red circles and regulated genes by green circles. Line width is proportional to the Fisher's *Z*-transformed partial correlation value returned by LPC.

As part of our analysis of publicly available *E. coli* transcriptome data, a sub-network consisting of the 50 most significant gene interactions predicted by our algorithm was generated (Fig. 4). Importantly, the majority of predicted regulatory relationships have been demonstrated experimentally in the past, as documented in RegulonDB (blue arrows), while others are not included in the current version of RegulonDB (marked in grey in Fig. 4). However, additional published data are consistent with regulatory relationships between some of the latter: *nac*, *glnK* and *amtB* are involved in responses to nitrogen starvation;[32–34] and FliA is a sigma factor that is necessary for transcription of class 3 flagellar genes, such as *flgK*.[35] Furthermore, the sigma factor FecI and the membrane proteins ExbB and ExbD act in coordination for iron uptake.[36,37] Thus, there is good agreement between this subset of our LPC-predicted GRN and previous experimental data that established gene interactions in *E. coli*. More importantly, novel gene regulatory relations predicted by our LPC

algorithm should guide experimental work to confirm and expand GRNs, not only in *E. coli*, but in the many other organisms for which little experimental information beyond genome and transcriptome data is available.

One reason for the apparent success of the LPC algorithm in reconstructing *E. coli* genetic networks is that GRNs are sparsely connected in most cases,[38] *i.e.*, there are far fewer directly connected genes for each gene than the total number of genes minus one. Therefore, the LPC-algorithm can infer most causal relationships in GRNs based on low-order CI tests.

All methods that attempt to reconstruct GRNs based on transcriptome data are subject to false positive and false negative errors arising from biological and computational sources. For example, the regulatory link between a TF and its target genes, whether it is positive or negative, will be missed (false negative) if transcriptional regulation of the TF is not the primary level of control for that gene/protein.[39] Thus, if the TF gene is constitutively-expressed and post-translational modification of the TF protein is the primary means by which TF activity is regulated, then a correlation between transcript levels of the TF gene and its target gene may not exist, and a regulatory link between the two genes never established, at least in wild-type cells. This may be less often the case in prokaryotes, where TF genes are often part of the operons that they control, than in eukaryotes, where this is not the case. Genetic approaches, especially the use of mutants defective in specific TF genes could help to overcome this problem. Use of transcriptome data from defined TF mutants in our LPC should help to reveal regulatory relationships that are hidden in wild-type data.

False positive inferences of regulatory relationships can arise when co-expressed genes are confused as co-regulated. Consider, for instance, two TFs involved in distinct signaling pathways with different roles, different environmental stimulation, and different target genes. If, in a given set of experiments the target genes have similar expression profiles, they will be falsely considered as co-regulated (false positive). Moreover, either of the two TFs could be mistaken as the common regulator of all the target genes, based on the limited experimental data, leading to a false inference of transcriptional regulation.[40] To avoid such problems, more informative experiments should be selected,[40] or alternate data sources such as promoter sequence information or ChIP-on-chip data should be incorporated with microarray data to enhance GRN inferences.

False positive and false negative associations between genes can also arise from the computational methods and parameters employed. Obviously, the threshold that is set for CI tests between two genes will have a major impact on the predicted GRN, with a higher threshold reducing the number of false positives and increasing the number of false negatives. This is a problem for all algorithms that rely on statistical treatment of data, including the LPC-algorithm. With the accumulation of known regulatory relationships with gene expression data, it is expected that optimal threshold values could be determined wisely. Similar to the PC-algorithm, another limitation of the LPC-algorithm is that false negatives arising from statistical tests cannot be corrected at a later stage and will lead to spurious final results.

Another source of errors in predicted GRNs are the statistical assumptions that are made, which may not correctly reflect the nature of the regulatory relationship between some genes. For example, the DAG assumption for transcriptional regulation does not accommodate more complex relationships, such as feedback loops. This is, in fact, a shortcoming of all static BN learning methods. Nevertheless, from the two phases of the LPC-algorithm, the undirected graph returned from the first phase still makes sense even for the feedback loop situation. Previous studies[41] showed that conditional independencies are still sound even in *directed cyclic graphs* (DCGs) within linear models. Thus, the undirected graph from LPC would identify regulatory connections between genes. The simulation tests over scale-free networks with feedback loops also support this notion (in the ESI†). The second phase of our algorithm may wrongly assign or fail to assign directions to edges when the DAG assumption is broken. However, due to the sparseness of connections in a GRN, our algorithm is still able to recover causal relationships when local network structures satisfy the assumptions in BNs. Furthermore, if sufficient time-series microarray data are available, the LPC-algorithm can be extended to dynamic BN models[42] to overcome feedback loop problems encountered in the DAG assumption. As GRN simulation technology continues to mature (such as inclusion of the interampatteness property to explain feedback loops,[43]) and suitable experimental time-series microarray data become available, it will be interesting to test the ability of our approach to model synthetic and real GRNs containing feedback loops.

Appendix A: pseudocode of the LPC-algorithm

The formal pseudocode of the LPC-algorithm is presented in Table 2.

Appendix B: proofs for the LPC-algorithm

The correctness of causal relationships identified by the LPC-algorithm is guaranteed in Theorem 1. The lemmas are per definition important since they are needed to prove the theorem.

Lemma 1

In a faithful BN, let $G = (\mathbf{V},\mathbf{E})$ be a DAG and $X, Y \in \mathbf{V}$. Then if X and Y are d-separated by some set, they are d-separated either by the set consisting of neighbors of X or the set consisting of neighbors of Y.

Proof

Clearly, if X and Y are adjacent in G, no set d-separates them as no set can block the path consisting of the edge between them.

In the other direction, suppose X and Y are not adjacent. Either there is no path from X to Y or there is no path from Y to X for otherwise we could have a cycle. Without loss of generality, assume there is no path from Y to X. We will show that X and Y are d-separated by the set $\mathbf{Adjacencies}(G,Y)\backslash X$ consisting of neighbors of Y. Consider any path π between

Table 2 Low-order PC-algorithm: LPC

Input:	D:	a dataset containing n nodes in m cases
	k:	the highest order can be performed in CI tests
	ε:	the threshold for CI tests
Output:	G:	the partial directed graph over n nodes

/* Phase 1: learn a graph skeleton */

Form a complete connected undirected graph G with n nodes;
Create the two-dimensional $n \times n$ arrays **Sepset**;

1: $l := 0$;
2: **repeat**
3: **repeat**
4: Select a new ordered pair of node X, Y that are adjacent in G such that |**Adjacencies** $(G,X)\backslash\{Y\}| \geq l$;
5: **repeat**
6: choose a new $\mathbf{S} \subseteq$ **Adjacencies** $(G,X)\backslash\{Y\}$ with $|\mathbf{S}|=l$;
7: **if** $dep_D(X,Y|\mathbf{S}) < \varepsilon$ **then**
8: Delete edge X and Y in G;
9: Save **S** in **Sepset**(X,Y) and **Sepset**(Y,X);
10: **end if**
11: **until** edge X and Y is deleted or all $\mathbf{S} \subseteq$ **Adjacencies** $(G,X)\backslash\{Y\}$ with $|\mathbf{S}|=l$ have been chosen
12: **until** all ordered pairs of adjacent variables X and Y such that |**Adjacencies** $(G,X)\backslash\{Y\} \geq l$ **and** $\mathbf{S} \subseteq$ **Adjacencies** $(G,X)\backslash\{Y\}$ with $|\mathbf{S}| = l$ have been tested for conditional independence
13: $l := l+1$;
14: **until** there is no adjacent nodes X, Y satisfying | **Adjacencies** $(G,X)\backslash\{Y\}| \geq l$ **or** $l > k$

/* Phase 2: Orientation */

15: **for** each triple of nodes X, Y, Z such that the pairs X, Y and pairs Y, Z are each adjacent in G but the pair X, Z are not adjacent in G
16: **if** min(| **Adjacencies**$(G,X)\backslash\{Y\}|$,| **Adjacencies**$(G,Y)\backslash\{X\}|) \leq k$ **and** min(| **Adjacencies**$(G,Z)\backslash\{Y\}|$,| **Adjacencies**$(G,Y)\backslash\{Z\}|) \leq k$ **and** $Y \notin$ **Sepset**(X,Z) **then**
17: orient X—Y—Z into $X \rightarrow Y \leftarrow Z$;
18: **end if**
19: **next**
20: **repeat**
21: **R1** Orient Y—Z into $Y \rightarrow Z$ whenever there is arrow $X \rightarrow Y$ such that X and Z are nonadjacent **and** min(| **Adjacencies**$(G,Y)\backslash\{Z\}|$, **Adjacencies**$(G,Z)\backslash\{Y\}|) \leq k$;
22: **R2** Orient X—Y into $X \rightarrow Y$ whenever there is a chain $X \rightarrow Z \rightarrow Y$ **and** min(| **Adjacencies**$(G,X)\backslash\{Y\}|$,| **Adjacencies**$(G,Y)\backslash\{X\}|) \leq k$;
23: **R3** Orient X—Y into $X \rightarrow Y$ whenever there are two chains X—$Z \rightarrow Y$ and X—$W \rightarrow Y$ such that Z and W are nonadjacent **and** min(| **Adjacencies**$(G,X)\backslash\{Y\}|$,| **Adjacencies**$(G,Y)\backslash\{X\}|) \leq k$ **and** min(| **Adjacencies**$(G,X)\backslash\{Z\}|$, | **Adjacencies**$(G,Z)\backslash\{X\}|) \leq k$ **and** min(|**Adjacencies**$(G,X)\backslash\{W\}|$,| **Adjacencies**$(G,W)\backslash\{X\}|) \leq k$;
24: **R4** Orient X—Y into $X \rightarrow Y$ whenever there are two chains X—$Z \rightarrow W$ and $Z \rightarrow W \rightarrow Y$ such that Z and Y are nonadjacent **and** min(| **Adjacencies**$(G,X)\backslash\{Y\}|$,| **Adjacencies**$(G,Y)\backslash\{X\}|) \leq k$ **and** min(| **Adjacencies**$(G,X)\backslash\{Z\}|$,| **Adjacencies**$(G,Z)\backslash\{X\}|) \leq k$;
25: **until** no more edges in G can be oriented

X and Y that does not pass through any node in **Parents**(G,Y). There must be a node Z having converging arrow along the π, otherwise it would be a path from Y to X. Consider Z is closest to X on path π, Z cannot be an ancestor or parent of any node in **Parents**(G,Y) and $Z \notin$ **Parents**(G,Y) because otherwise we have a cycle. Clearly, any path between X and Y are blocked by **Parents**(G,Y) according to the d-separation definition. **Adjacencies**$(G,Y)/X$, which completes the proof.∴

Lemma 2

In a faithful BN, let $G = (\mathbf{V},\mathbf{E})$ be a DAG and $X,Y \in \mathbf{V}$ and |**Adjacencies**$(G,X)\backslash\{Y\}| \leq$ |**Adjacencies**$(G,Y)\backslash\{X\}|$) and |**Adjacencies**$(G,X)\backslash\{Y\}| = k$. Then if X and Y are d-separated by some set, they are d-separated by a set **S** such that $\mathbf{S} \subseteq$ **Adjacencies**$(G,X)\backslash\{Y\}$ and $|\mathbf{S}| \leq k$.

Proof

Follows from Lemma 1, if X and Y are d-separated by some set, they are d-separated by a subset (denoted as **S**) of **Adjacencies**$(G,X)\backslash\{Y\}$. Since |**Adjacencies**$(G,X)\backslash\{Y\}| \leq$ |**Adjacencies**$(G,Y)\backslash\{X\}|$ and |**Adjacencies**$(G,X)\backslash\{Y\}| = k$, the cardinality of any subset of **Adjacencies**$(G,X)\backslash\{Y\}$ is no more than k because |**Adjacencies**$(G,X)\backslash\{Y\}| = k$. Hence X and Y are d-separated by the set **S** such that $\mathbf{S} \subseteq$ **Adjacencies**$(G,X)\backslash\{Y\}$ and $|\mathbf{S}| \leq k$.∴

Theorem 1 (soundness)

Under faithful assumption, the set of all undirected edges returned by the LPC-algorithm is a superset of undirected edges in the CPDAG. Any directed edge created by the LPC-algorithm is a directed edge in the CPDAG.

Proof

The first phase of our algorithm is similar to the PC-algorithm,[17] only difference is only 0–k order CI tests were performed (in line 14 of Table 2). The edges in skeleton can be broken by higher-order CI tests in the PC-algorithm will be kept in the LPC-algorithm. All the edges broken by 0–k order CI tests in the LPC-algorithm also can be broken by the PC-algorithm. Follows from Th. 5.1, Ch. 13 in ref. 17 which claims the correctness of the skeleton from the PC-algorithm. Hence, the set of all edges returned by the LPC-algorithm is a superset of edges in the CPDAG.

In the second phase of the LPC-algorithm, for each orientation rule, the numbers of adjacencies of each node per node pair (denoted by X and Y) are examined (in line 16 and lines 21–24 of Table 2), it follows from Lemma 2 and min(|**Adjacencies**$(G,X)\backslash\{Y\}|$, |**Adjacencies**$(G,Y)\backslash\{X\}|$) $\leq k$, without loss of generality, X and Y is also adjacent in CPDAG because otherwise the edge between X and Y are removed by 0–k order CI tests in the phase 1. The conditions of all the orientation rules in phase 2 are still satisfied. It follows from Theorem 2 in ref. 21 that all the orientation rules are still sound. Hence, all the directed edges inferred from the orientation phase in LPC are also presented in CPDAG.∴

Appendix C: partial correlation calculations

For conditional independence (CI) tests used in our algorithm, we followed the partial correlation calculation used in ref. 17 and 22, that is, the sample partial correlation ρ, for any $Z \in \mathbf{S}$,

$$\rho_{X,Y|\mathbf{S}} = \frac{\rho_{X,Y|\mathbf{S}\backslash Z} - \rho_{X,Z|\mathbf{S}\backslash Z}\rho_{Y,Z|\mathbf{S}\backslash Z}}{\sqrt{\left(1-\rho^2_{X,Y|\mathbf{S}\backslash Z}\right)\left(1-\rho^2_{X,Y|\mathbf{S}\backslash Z}\right)}} \qquad (1)$$

The zeroth-order partial correlation $\rho_{X,Y|\mathbf{\emptyset}}$ is defined to be the regular Pearson correlation coefficient $\rho_{X,Y}$. Actually, the

k-th order partial correlation (*i.e.*, with $|\mathbf{S}| = k$) can be easily computed from three $(k-1)$th order partial correlations. Thus the sample partial correlation $\rho_{X,Y|\mathbf{S}\setminus Z}$ can be calculated recursively by using eqn (1).

For testing whether a partial correlation is zero or not, we apply Fisher's Z-transformation of the partial correlation:

$$\sqrt{m-\mathbf{S}-3}|Z(X,Y|\mathbf{S})| > \Phi^{-1}(1-\alpha/2) \qquad (2)$$

Classical decision theory yields the following rule when using significance level α. Reject the null-hypothesis $H_0(X,Y|\mathbf{S})$: $\rho_{X,Y|\mathbf{S}} = 0$ against the two-sided alternative $H_A(X,Y|\mathbf{S})$: $\rho_{X,Y|\mathbf{S}} \neq 0$ if

$$Z(X,Y|\mathbf{S}) = \frac{1}{2}\log\left(\frac{1+\hat{\rho}_{X,Y|\mathbf{S}\setminus Z}}{1-\hat{\rho}_{X,Y|\mathbf{S}\setminus Z}}\right) \qquad (3)$$

where $\Phi(\cdot)$ denotes the cumulative distribution function of a Gaussian distribution with zero mean and unit standard deviation and m is the sample size. Thus, the left-hand side of eqn (3) is used as $\boldsymbol{dep}_{\mathrm{D}}(X,Y|\mathbf{S})$ and the right-hand side of eqn (3) is used as the threshold ε in LPC.

Acknowledgements

This research was funded by the National Science Foundation (Grant No. DB-0703285), Oaklahoma Centre for the Advancement of Science Technology (Project No. PSB09-032) and the Samuel Roberts Noble Foundation.

Notes and references

1 S. L. Lauritzen, *Graphical Models*, Oxford University Press, New York, USA, 1996.
2 J. M. Stuart, E. Segal, D. Koller and S. K. Kim, *Science*, 2003, **302**, 249–255.
3 J. Schafer and K. Strimmer, *Bioinformatics*, 2005, **21**, 754–764.
4 A. A. Margolin, I. Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R. Dalla Favera and A. Califano, *BMC Bioinformatics*, 2006, **7**(Suppl 1), S7.
5 A. Wille and P. Buhlmann, *Stat. Appl. Genet. Mol. Biol.*, 2006, **5**, Article 1.
6 A. Reverter and E. K. Chan, *Bioinformatics*, 2008, **24**, 2491–2497.
7 S. Lebre, *Stat. Appl. Genet. Mol. Biol.*, 2009, **8**, Article 9.
8 N. Friedman, M. Linial, I. Nachman and D. Pe'er, *J. Comput. Biol.*, 2000, **7**, 601–620.
9 M. Wang, Z. Chen and S. Cloutier, *Comput. Biol. Chem.*, 2007, **31**, 361–372.
10 I. Tsamardinos, L. E. Brown and C. F. Aliferis, *Mach. Learn.*, 2006, **65**, 31–78.
11 D. Husmeier, *Bioinformatics*, 2003, **19**, 2271–2282.
12 S. Ma, Q. Gong and H. J. Bohnert, *Genome Res.*, 2007, **17**, 1614–1625.
13 K. Basso, A. Margolin, G. Stolovitzky, U. Klein, R. Dalla-Favera and A. Califano, *Nat. Genet.*, 2005, **37**, 382–390.
14 A. Wille, P. Zimmermann, E. Vranova, A. Furholz, O. Laule, S. Bleuler, L. Hennig, A. Prelic, P. von Rohr, L. Thiele, E. Zitzler, W. Gruissem and P. Buhlmann, *Genome Biol.*, 2004, **5**, R92.
15 C. J. Wolfe, I. S. Kohane and A. J. Butte, *BMC Bioinformatics*, 2005, **6**, 227.
16 F. Markowetz and R. Spang, *BMC Bioinformatics*, 2007, **8**(Suppl 6), S5.
17 P. Spirtes, C. Glymour and R. Scheines, *Causation, Prediction, and Search*, MIT Press, 2000.
18 R. Neapolitan, *Learning Bayesian Networks*, Prentice Hall Upper Saddle River, NJ, 2004.
19 T. Verma and J. Pearl, in *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, ed. M. Herion, R. Schachter, L. Kanal and J. Lemmer, 1990, 255–270.
20 D. Chickering, *J. Mach. Learn. Res.*, 2002, **2**, 445–498.
21 C. Meek, *Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence*, Montreal, QC, ed P. Besnard and S. Hanks, 1995, 403–441.
22 M. Kalisch and P. Bühlmann, *J. Mach. Learn. Res.*, 2007, **8**, 613–636.
23 N. Soranzo, G. Bianconi and C. Altafini, *Bioinformatics*, 2007, **23**, 1640–1647.
24 P. Mendes, W. Sha and K. Ye, *Bioinformatics*, 2003, **19**(Suppl 2), ii122–ii129.
25 A. V. Werhli, M. Grzegorczyk and D. Husmeier, *Bioinformatics*, 2006, **22**, 2523–2531.
26 D. Geiger and D. Heckerman, in *Proceedings of the 10th Annual Conference on Uncertainty in Artificial Intelligence*, ed R. Mantaras and D. Poole, 1994, 235–243.
27 J. P. Pellet and A. Elisseeff, *J. Mach. Learn. Res.*, 2008, **9**, 1295–1342.
28 H. Salgado, S. Gama-Castro, M. Peralta-Gil, E. Diaz-Peredo, F. Sanchez-Solano, A. Santos-Zavaleta, I. Martinez-Flores, V. Jimenez-Jacinto, C. Bonavides-Martinez, J. Segura-Salazar, A. Martinez-Antonio and J. Collado-Vides, *Nucleic Acids Res.*, 2006, **34**, D394–397.
29 J. J. Faith, B. Hayete, J. T. Thaden, I. Mogno, J. Wierzbowski, G. Cottarel, S. Kasif, J. J. Collins and T. S. Gardner, *PLoS Biol.*, 2007, **5**, e8.
30 F. Mordelet and J.-P. Vert, *Bioinformatics*, 2008, **24**, i76–82.
31 J. Ernst, Q. K. Beg, K. A. Kay, G. Balazsi, Z. N. Oltvai and Z. Bar-Joseph, *PLoS Comput. Biol.*, 2008, **4**, e1000044.
32 H. Jiang, L. Shang, S. H. Yoon, S. Y. Lee and Z. Yu, *Biotechnol. Lett.*, 2006, **28**, 1241–1246.
33 M. S. Kabir, T. Sagara, T. Oshima, Y. Kawagoe, H. Mori, R. Tsunedomi and M. Yamada, *Microbiology*, 2004, **150**, 2543–2553.
34 T. A. Blauwkamp and A. J. Ninfa, *Mol. Microbiol.*, 2002, **46**, 203–214.
35 S. Kalir, J. McClure, K. Pabbaraju, C. Southward, M. Ronen, S. Leibler, M. G. Surette and U. Alon, *Science*, 2001, **292**, 2080–2083.
36 M. Ochs, A. Angerer, S. Enz and V. Braun, *Mol. Gen. Genet.*, 1996, **250**, 455–465.
37 A. Stiefel, S. Mahren, M. Ochs, P. T. Schindler, S. Enz and V. Braun, *J. Bacteriol.*, 2001, **183**, 162–170.
38 H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai and A. L. Barabasi, *Nature*, 2000, **407**, 651–654.
39 M. J. Herrgard, M. W. Covert and B. O. Palsson, *Genome Res.*, 2003, **13**, 2423–2434.
40 E. Szczurek, I. Gat-Viks, J. Tiuryn and M. Vingron, *Mol. Syst. Biol.*, 2009, **5**, 287.
41 T. Richardson and P. Spirtes, *Computation, Causation, and Discovery*, 1999, pp. 253–304.
42 K. Murphy and S. Mian, technique report, *Modelling Gene Expression Data using Dynamic Bayesian Networks*, University of California, Berkeley, 1999.
43 T. E. M. Nordling and E. W. Jacobsen, *IET Syst. Biol.*, 2009, **3**, 388–405.

