Large Causal Gene Regulatory Network Inference by Decomposition into Subnetworks

Leung-Yau Lo[1*], Man-Leung Wong[2], Kin-Hong Lee[1], Kwong-Sak Leung[1]

[1] Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.
[2] Department of Computing and Decision Sciences, Lingnan University, Tuen Mun, Hong Kong.

* Corresponding author. Tel.: (852)64887615; email: lylo@cse.cuhk.edu.hk
Manuscript submitted March 9, 2015; accepted May 15, 2015.
doi: 10.17706/ijbbb.2015.5.3.175-183

Abstract: Inferring the gene regulatory network is an important first step toward understanding the work of the cell and consequently curing diseases related to malfunction of the cell. One thorny problem in gene regulatory network inference is that even with high throughput technology, the available time series expression data is still very limited compared to the network size.

To alleviate this problem, we propose to decompose large network into small subnetworks without prior knowledge of the decomposition. Our algorithm first infers an initial GRN using CLINDE, decomposes it into possibly overlapping subnetworks, then infers each subnetwork by either CLINDE or DD-lasso and finally merges the subnetworks. We have tested this algorithm on synthetic data of networks with 500 and 1000 genes. The results show that our proposed algorithm does improve the GRN inference performance of using either CLINDE or DD-lasso alone on the large network, with statistical significance, and is robust to different variances and slight deviation from Gaussian distribution in error terms.

Key words: Decomposition, large gene network inference, time delay, time series expression data.

1. Introduction

Inferring gene regulatory network (GRN) has been an important problem in bioinformatics. Genes are transcribed into mRNAs and then translated into proteins. Transcription Factors (TFs) trigger or inhibit the transcription of other genes. Non-negligible transcriptional and translational delays have been observed [1]-[3]. These delays are known to affect the network stability, or cause oscillations [4]-[7]. In order to understand the work of the cell and subsequently diseases related to the malfunctio of the cell (such as cancer), it is crucial that the GRN be first mapped out, with regulatory effects and delays. Doing so experimentally is too time-consuming and expensive, therefore computational methods on high-throughput microarray or RNA-seq expression data are attractive complementary means to infer the GRN.

2. Background: Gene Network Inference

There are many GRN inference algorithms and models, with different levels of details, see Ref. [8], [9] for surveys of GRN modelling and Ref. [10] for survey on GRN inference algorithms for microarray expression data.

Most models of GRN are graphs, with vertices being genes, and edges being regulatory relationships. Different levels of details could be achieved by labeling the edges with extra information. In undirected graph, only an association network is captured, e.g. ARACNE [11] and C3NET [12]. Alternatively, directed

edges could be used, as in Ref. [13] which is genetic algorithm method, but it does not label the edges with time delays. Some algorithms consider only delay of one time step, as in Ref. [14], which uses association rule mining to find frequent regulatory patterns. Boolean network, e.g. in Ref. [15], is a classic model of GRN with only one time step. Ordinary Differential Equations (ODE), when discretized in time, reduces to one time step model, as in Ref. [16], which uses Gaussian process for Bayesian inference of an ODE model, and DELDBN [17], which combines ODE model with local Bayesian analysis. Dynamic Bayesian Network (DBN), which allows delays and cycles, is sometimes used as one time step model, as in Ref. [18].

Not many algorithms infer multiple time delays. Reference [19] first estimates the possible delays from pairwise mutual information from discretized expression data, then infers multiple time step DBN by minimizing MDL score using genetic algorithm. Banjo [20] also optimizes a score metric on DBN using discretized expression data by MCMC based method, and updated version of the program allows multiple delays. TD-ARACNE [21] is an extension of ARACNE with time delays. But these algorithms do not label the edges of GRN with regulatory effect. In contrast, in DD-lasso [22], the expression of a gene is a linear combination of expression of its regulators at (possibly different) previous time steps. It first estimates the delays between each gene pairs by maximum likelihood, then uses lasso [23] to remove indirect effects and estimate the coefficients, therefore the edges are labeled with the delays as well as the regulatory effects. CLINDE [24] uses a similar model, but uses conditional independence of the shifted time series to estimate the delays and eliminate indirect effects.

Some algorithms use perturbation data, e.g. [25], [26]; or need TF binding information, e.g. [27]; or use a combination of perturbation data and time series expression, e.g. TSNI [28].

3. Motivation: Difficulty of Inferring Large Network from Limited Data

In high-throughput the time series expression data, the number of genes is usually very large (in the thousands), but the number of time points is far smaller (in the tens), which still poses a severe challenge to inferring a causal GRN. Besides the cost, there is another difficulty to obtaining longer time series. Current technology requires a sample of cells to get sufficient signals of gene expressions at one time point, so the cell cycles of the sample cells have to be synchronized at the beginning of the experiment. However, the inherent stochastic nature of the operation of the cell cycle would make the cells increasingly unsynchronized. Consequently, the expression values obtained would be increasingly "blurred" as the time series gets longer. Therefore, the useful length of a time series is practically limited.

One way to alleviate this problem is to perform biological replicates, and make the inference algorithm utilize multiple relatively short time series instead of a long one. Both CLINDE and DD-lasso can accept multiple time series. In this paper, we focus on another possibility: to decompose a large GRN into smaller possibly overlapping subnetworks without prior knowledge, then infer each subnetwork, and finally "stitch" the subnetworks to get an overall GRN. This is feasible owing to the sparsity of GRNs. The main difficulty is to decompose a GRN into subnetworks without prior knowledge of the GRN structure.

4. Methods

4.1. Overview

We assume the true GRN consists of loosely connected and sparse subnetworks, as illustrated in Fig. 1(a). These assumptions may suggest a clustering based method for decomposition, but there are some issues to address. Firstly, the similarity measure needs to take into account the time delays. This could be solved by considering the correlation between shifted time series, and trying the possible time delays (up to a maximum allowed delay) to define the similarity measure. Secondly, indirect effects, which lead to correlation between genes not directly dependent, need to be taken into account. For example, if $a \rightarrow b \rightarrow c$,

then a high correlation between a and c will be observed. Indirect effects may cause more genes in different subnetworks to appear dependent, which may make the clustering more difficult. This is illustrated in Fig. 1(b). Thirdly, most clustering methods give disjoint clusters as output, in which case either further processing is needed to find the edges across subnetworks, or these edges are ignored. Therefore, either overlapping clusters are found, or disjoint clusters have to be "expanded".

Because of the above, we propose to first infer an initial GRN using CLINDE, which handles time delays and helps eliminate indirect effects, then decompose the initial GRN into overlapping subnetworks.

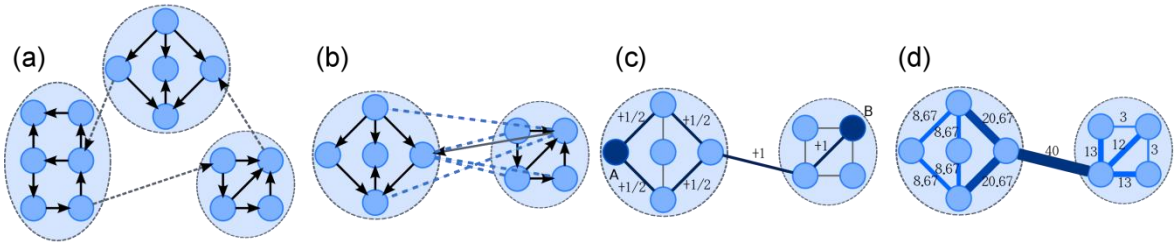

Fig. 1. (a) Structure of subnetworks; (b) Indirect effect across subnetworks; (c) Contribution of vertices pairs to edge betweenness; (d) Example of edge betweenness.

The overall flow is given in Fig. 2. The steps are 1) initial GRN inference using CLINDE, 2) decomposition using edge betweenness to get subsets of genes, 3) each gene subset is used to infer a subnetwork using either CLINDE or DD-lasso, 4) the subnetworks are merged to obtain the final GRN. In the following, the input data, GRN model, and the steps of the proposed algorithm are described.

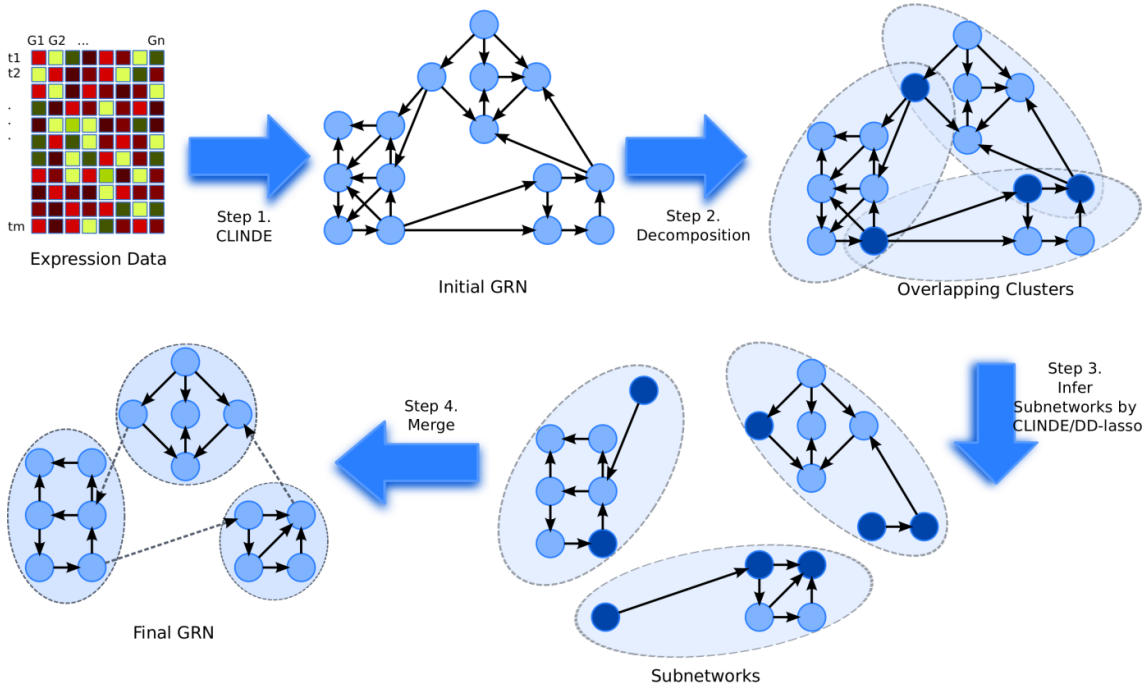

Fig. 2. Overall flow of the algorithm.

4.2. Data and Model

The given data is $\{x_i(t)\}$, for $i = 1, \ldots, n$, $t = 1, \ldots, m$, where $x_i(t)$ is the expression value of gene i at time t, and there are n genes and m equidistant time points, and the model is:

$$x_j(t) = \sum_{i=1}^{n} a_{ij} x_i(t - \tau_{ij}) + \varepsilon_j(t) \quad (1)$$

so that a_{ij} is the regulatory effect of gene i on gene j, where the regulatory effect is repressive if a_{ij} is negative, activatory if positive, and absent if zero; and τ_{ij} is the positive time delay of the edge $i \to j$ (if $a_{ij} \neq 0$); and $\varepsilon_j(t)$ is the error term for gene at time t. We assume that the error terms are zero-mean, and are mutually independent, but otherwise we do not make stringent assumptions on the distribution of the error terms. Note that this model allows self-regulation and cycles (with positive delays) in the GRN.

4.3. Initial GRN

Only a few GRN inference algorithms handle multiple time delays, CLINDE [24] and DD-lasso [22] are two of them. And from [24], CLINDE outperforms DD-lasso when the number of time points is small relative to the number of genes, so we choose CLINDE to infer an initial GRN.

CLINDE is based on the PC algorithm [29]-[30], and consists of two stages. Stage 1 considers all (directed) pairs of genes x and y, and all possible delays d up to a maximum delay, to determine if $x \to y$ is significant with the delay d based on either correlation test, or mutual information test. The test is considered significant if the score of the test is larger than a score threshold. In the (partial) correlation test, the score is $-log_{10}(p\_value)$, and in the (conditional) mutual information test, the score is the (conditional) mutual information. For correlation test, the regulatory effect (positive or negative) is also estimated. After stage 1, there may be multiple edges from x to y, but with different delays. Stage 2 attempts to prune the edges by partial correlation tests or conditional mutual information tests. Iteratively, the remaining edges are considered for pruning by conditioning first on $h = 1$ neighbor, then $h = 2$ neighbors, and so on up to $h = N_0$, for a given parameter N_0, where the neighbors are shifted properly using the delays estimated in stage 1. If the conditional test is not significant, the edge is pruned.

For our purpose of decomposition into subnetworks, after stage 2, we "condense" the multiple edges $x \to y$ with between the same pair of genes to retain only the one with the most significant p-value.

4.4. Decomposition

Given an initial GRN, we decompose it by identifying edges likely to be across subnetworks, by using "edge betweenness" [31]. In an undirected graph, consider the shortest path(s) between two vertices. If there are k shortest paths, for each shortest path, each constituent edges receives a weight of $1/k$. The "edge betweenness" for an edge is the sum of weights after considering all vertex pairs. This is illustrated in Fig. 1(c). Intuitively, edge with higher edge betweenness is more likely to be across subnetworks, because the shortest paths of all vertex pairs in different subnetworks have to go through those few edges across subnetworks, this is illustrated in Fig. 1(d). Reference [31] gives a fast method to calculate edge betweenness of all edges in a graph with E edges and V vertices in $O(EV)$ time.

The steps for decomposition are as follows, starting from the initial GRN (considered as undirected):

1) Identify all the components (the maximally connected subgraphs).
2) If a component has size of at least S_0, calculate the edge betweenness, and remove the edge with the highest edge betweenness.
3) Go back to step 2 until all components are smaller than S_0.

The size threshold S_0 should not be too small, because removing more edges means higher chance that the genes are grouped wrongly. On the other hand, S_0 should not be too large to avoid difficulty in subnetwork learning due to limited data. The default threshold S_0 used is 60, based on the previous performance of CLINDE and DD-lasso.

After that, we consider three ways to obtain the final subsets of gene for the subnetworks:

1) **Component**: Simply output each component as a subset, but this gives disjoint partitions.
2) **Parents**: For each component, include its parents based on the initial GRN. Presumably the removed edges are those across subnetworks, we include the parents so that these cross edges could be identified in the subnetworks. And in this case, the subnetworks are likely overlapping.
3) **XParents**: For each component, include its parents based on the initial GRN. But for each subnetwork, after inference, the edges between the parents (not in the component) are removed. The motivation is to help remove indirect effects between the parents.

4.5. Infer Subnetworks

The possibly overlapping subsets of genes (either one of *Component*, *Parents* or *XParents*) has been obtained. The corresponding subset of expression data could be obtained and each subnetwork could be re-learned. Both CLINDE and DD-lasso could be used for this purpose.

DD-lasso is based on lasso [23], which is a regularized regression method that also has the effect of feature selection. DD-lasso extends lasso to handle time delays. DD-lasso consists of three stages. In stage 1, for all directed pairs of genes x and y, determine the delay d such that $x \to y$ has the maximum absolute correlation when shifted by delay d. In stage 2, for each gene g, treat all the genes as potential parents, with the delays determined in stage 1. Then, lasso is used to predict the real parents of g through the feature selection nature of lasso. In stage 3, backward elimination is done for each gene to further remove parents that are likely only indirect effects. Note that in DD-lasso, it is assumed that between any two genes there is only one edge with a single delay, whereas in CLINDE this is not assumed.

Since the initial GRN is estimated using CLINDE, so after decomposing into subnetworks, using DD-lasso may lead to greater improvement on the subnetworks, even though DD-lasso may not perform well on the large network, given the limited data, as it has different performance characteristics from CLINDE.

4.6. Merge the Subnetworks

After the subnetworks are re-learnt using either CLINDE or DD-lasso, they can simply be unioned to obtain the final estimate of the GRN, because the overlapping nature of the subnetworks avoids the need of post-processing for the cross edges.

5. Experiment Results and Discussions

Since it is difficult to find known large gene networks and sufficiently long time series data of the involved genes, we use synthetic data to evaluate our proposed algorithm, where we know the underlying gene network, and there is no lack of sufficient expression data.

5.1. Performance Metrics

We assess the performance mainly on Effects, which is correct if and only if both the link and the sign of the effect a_{ij} are correct. From [24], it is usual to have the effects and delays being correct at the same time, rather than getting one but not the other. We use *F-score = (*2 × *Recall* × *Precision)/(Recall + Precision)* as an overall measure of performance.

5.2. Generation of Synthetic Network and Expression Data

To generate a subnetwork, we randomly choose 1 up to M_0 parents for each gene. For each link, the delay is uniformly chosen from 1 up to τ_0, and the coefficient has absolute value randomly chosen between 0.5 and 1.5, with random sign. The coefficients are scaled to make the network stable. To generate a large network of size n as in Fig. 1(a), we first generate ceil(n/n_0) subnetworks, each with size n_0; then

generate a network of size ceil(n/n_0), each vertex representing a subnetwork, and for each link, we connect two random genes from the corresponding subnetwork. Finally permute the gene indices to prevent the subnetworks from being easily identified from the indices.

Having generated the network, we generate expression data according to (1), where each error term is $\varepsilon_j(t) = sign(z_{jt})|z_{jt}|^{\alpha}$, where z_{jt} is $N(0,\sigma^2)$. Therefore, σ controls the variance, and α controls the gaussianity, and varying α allows us test slight deviation from Gaussian distribution.

5.3. Experiment Settings

We have tested these settings: network size n=500, 1000; subnetwork size of 50; σ=0.5, 2, 8; α=0.5, 1, 2, 3; maximum number of parents M_0=4; maximum delay τ_0=4; number of time points m=200. For each setting, we generate 20 random replicates, with a total of 480 networks and time series.

In using CLINDE for learning initial GRN and subnetworks, we use the default (partial) correlation tests, and try the score thresholds 2, 3, 4. But due to limited space, we show the results for score threshold 3 only, as the results for 2 and 4 are similar. For other parameters, we use the default values.

5.4. Synthetic Data Results

Table 1. Median Effects *F-Score* of CLINDE and DD-lasso by Decomposition on the Networks

					Component		**Parents**		**XParents**	
n	***α***	***σ***	**CL-init**	**DD**	**CL-sub**	**DD-sub**	**CL-sub**	**DD-sub**	**CL-sub**	**DD-sub**
500	0.5	0.5	0.773	0.073	0.772	0.783**	0.770	**0.824**++	0.774	0.823++
		2	0.769	0.073	0.775**	0.791+	0.766	**0.826**++	0.771	0.825++
		8	0.768	0.073	0.770+	0.772**	0.764	**0.811**++	0.768*	**0.811**++
	1	0.5	0.772	0.075	0.772*	0.787**	0.768	**0.830**++	0.773*	0.829++
		2	0.775	0.072	0.770	0.781*	0.774	**0.824**++	0.777**	0.822++
		8	0.768	0.072	0.769*	0.786+	0.766	**0.826**++	0.770+	0.825++
	2	0.5	0.759	0.078	0.756	0.779+	0.746	**0.820**++	0.753	**0.820**++
		2	0.747	0.073	0.751*	0.774++	0.741	**0.808**++	0.746	0.807++
		8	0.746	0.073	0.753	0.765+	0.741	**0.807**++	0.747	0.806++
	3	0.5	0.643	0.082	0.623	0.636	0.611	0.673++	0.643	**0.675**++
		2	0.629	0.080	0.589	0.561	0.592	**0.647**++	0.622	0.646++
		8	0.635	0.080	0.612	0.619	0.593	0.665++	0.630	**0.666**++
1000	0.5	0.5	0.750	0.081	0.758+	0.775++	0.746	**0.822**++	0.752**	0.820++
		2	0.737	0.082	0.743+	0.756++	0.732	**0.811**++	0.737**	0.810++
		8	0.740	0.082	0.752++	0.767++	0.733	**0.816**++	0.740**	0.813++
	1	0.5	0.742	0.081	0.750**	0.771+	0.737	**0.815**++	0.745**	0.814++
		2	0.743	0.081	0.750+	0.770+	0.737	**0.820**++	0.745++	0.818++
		8	0.741	0.082	0.743*	0.759+	0.734	**0.810**++	0.741*	0.808++
	2	0.5	0.719	0.077	0.731+	0.752++	0.709	**0.797**++	0.722	0.795++
		2	0.714	0.077	0.717	0.730+	0.706	**0.784**++	0.713	0.783++
		8	0.712	0.077	0.720**	0.736++	0.701	**0.787**++	0.711*	0.786++
	3	0.5	0.603	0.070	0.577	0.589	0.551	0.617++	0.598	**0.629**++
		2	0.595	0.069	0.554	0.537	0.543	0.604**	0.588	**0.614**+
		8	0.596	0.069	0.564	0.547	0.542	0.609+	0.590	**0.615**++

Medians are taken over 20 replicates. *: p-value < 0.1. **: p-value < 0.01. +: p-value < 1e-3. ++: p-value < 1e-5.

Table 1 shows the median (taken over the 20 replicates) *Effects F-score* of first inferring an initial GRN by CLINDE (CL-init), then decompose the initial GRN (*Component*, *Parents* and *XParents*), then use CLINDE and DD-lasso to infer the subnetworks (CL-sub and DD-sub respectively), and finally merge the subnetworks. DD is DD-lasso alone on the large network. We have highlighted the best entry in each row. We have also

performed one-sided Wilcoxon signed-rank test on whether the median *F-score* of CL-sub and DD-sub are better than CL-init, and label the significant entries.

Firstly, note that DD alone has poor performance, while CL-init has reasonable performance. The performance of CL-sub is comparable to or sometimes better than CL-init with statistical significance. Also note that DD-sub is better than CL-init with statistical significance in almost all cases, except for *Component* with α=3. Also DD-sub is substantially better than DD. This shows the effectiveness of our proposed algorithm, and that decomposing the large network without prior knowledge of the decomposition can improve the inference performance.

Comparing *Component*, *Parents* and *XParents*, the trend is not very clear for CL-sub, but for DD-sub, *Parents* and *XParents* are better than *Component*, and *Parents* is slightly better than *XParents*. This shows that including the parents of the component helps, presumably that helps to recover the links across subnetworks. Also, comparing *Parents* and *XParents*, it shows that removing the links between parents not in the component does not help much, presumably it is because DD-lasso can already effectively remove indirect effects on the subnetworks.

Also note that the results are quite consistent with different σ and α, which shows that this algorithm is robust to different variances and slight deviation from Gaussian distribution for the error terms.

6. Conclusion

We have demonstrated the effectiveness of our algorithm in improving the inference of large causal GRN with time delays from limited data by decomposing into subnetworks without prior knowledge of the decomposition. For future work, we intend to study the effects of using different GRN inference algorithm for initial GRN, and using different methods for decomposition into subnetworks.

Acknowledgment

This research is partially supported by GRF Grant (Project References 414413, LU310111) from the Research Grants Council of Hong Kong Special Administrative Region.

References

[1] Pavlov, M. Y., & Ehrenberg, M. (1996). Rate of translation of natural mrnas in an optimized *in vitro* system. *Arch*. *Biochem*. *Biophys*., *328*, 9-16.

[2] Mier-y Ter´an-Romero, L., Silber, M., & Hatzimanikatis, V. (2010). The origins of time-delay in template biopolymerization processes. *PLoS Comput Biol*, *6*, e1000726.

[3] Swinburne, I. A., & Silver, P. A. (2008). Intron delays and transcriptional timing during development. *Developmental Cell*, *14*, 324-330.

[4] Chen, L., & Aihara, K. (2002). Stability of genetic regulatory networks with time delay. *Circuits and Systems I: Fundamental Theory and Applications*, *IEEE Transactions on*, *49*, 602-608.

[5] Lewis, J. (2003). Autoinhibition with transcriptional delay: A simple mechanism for the zebrafish somitogenesis oscillator. *Curr*. *Biol*., *13*, 1398-1408.

[6] Mather, W., Bennett, M. R., Hasty, J., & Tsimring, L. S. (2009). Delay-induced degrade-and-fire oscillations in small genetic circuits. *Phys. Rev. Lett.*, *102*, 068105.

[7] Tigges, M., Marquez-Lago, T. T., Stelling, J., & Fussenegger, M. (2009). A tunable synthetic mammalian oscillator. *Nature*, *457*, 309-312.

[8] Karlebach, G., & Shamir, R. (2008). Modelling and analysis of gene regulatory networks. *Nature Reviews Molecular Cell Biology*, *9*, 770-780.

[9] Schlitt, T., & Brazma, A. (2007). Current approaches to gene regulatory network modelling. *BMC*

Bioinformatics, *8*, S9.

[10] Bansal, M., Belcastro, V., Ambesi-Impiombato, A., & di Bernardo, D. (2007). How to infer gene networks from expression profiles. *Molecular Systems Biology*, *3*, 78.

[11] Margolin, A., Nemenman, I., Basso, K., Wiggins, C., Stolovitzky, G., *et al*. (2006). Aracne: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, *7*, S7.

[12] Altay, G., & Emmert-Streib, F. (2010). Inferring the conservative causal core of gene regulatory networks. *BMC Systems Biology*, *4*, 132.

[13] Ram, R., Chetty, M., & Dix, T. (2006). Causal modeling of gene regulatory network. *In: Computational Intelligence and Bioinformatics and Computational Biology, 2006. CIBCB '06. 2006 IEEE Symposium on* (pp. 1-8).

[14] Kuo, H. C., Tsai, P. C., & Huang, J. P. (2009). Finding time-delayed gene regulation patterns from microarray data. *Proceedings of Ninth International Conference on Hybrid Intelligent Systems, 1*, 117-122.

[15] Maucher, M., Kracher, B., Kühl, M., & Kestler, H. A. (2011). Inferring Boolean network structure via correlation. *Bioinformatics*, *27*, 1529-1536.

[16] Äijö, T., & Lähdesmäki, H. (2009). Learning gene regulatory networks from gene expression measurements using non-parametric molecular kinetics. *Bioinformatics*, *25*, 2937-2944.

[17] Li, Z., Li, P., Krishnan, A., & Liu, J. (2011). Large-scale dynamic gene regulatory network inference combining differential equation models with local dynamic Bayesian network analysis. *Bioinformatics*, *27*, 2686-2691.

[18] Tienda-Luna, I. M., Huang, Y., Yin, Y., Padillo, D. P. R., & Perez, M. C. C. (2007). Uncovering gene regulatory networks from time-series microarray data with variational bayesian structural expectation maximization. *EURASIP J. Bioinformatics and Systems Biology*, *2007*, 71312.

[19] Xing, Z., & Wu, D. (2006). Modeling multiple time units delayed gene regulatory network using dynamic bayesian network. *Proceedings of the Sixth IEEE International Conference on Data Mining – Workshops* (pp. 190-195). Washington, DC, USA: IEEE Computer Society, ICDMW '06.

[20] Yu, J., Smith, V. A., Wang, P. P., Hartemink, A. J., & Jarvis, E. D. (2004). Advances to bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, *20*, 3594-3603.

[21] Zoppoli, P., Morganella, S., & Ceccarelli, M. (2010). Timedelay-aracne: Reverse engineering of gene networks from time-course data by an information theoretic approach. *BMC Bioinformatics*, *11*, 154.

[22] ElBakry, O., Ahmad, M., & Swamy, M. (2013). Inference of gene regulatory networks with variable time delay from time-series microarray data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *10*, 671-687.

[23] Tibshirani, R. (1994). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, *Series B*, *58*, 267-288.

[24] Lo, L. Y., Leung, K. S., & Lee, K. H. (2015). Inferring time-delayed causal gene network using time-series expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (to be published).

[25] Gregoretti, F., Belcastro, V., di Bernardo, D., & Oliva, G. (2010). A parallel implementation of the network identification by multiple regression (nir) algorithm to reverse-engineer regulatory gene networks. *PLoS ONE*, *5*, e10179.

[26] Julius, A., Zavlanos, M., Boyd, S., & Pappas, G. J. (2009). Genetic network identification using convex programming. *IET Systems Biology*, *3*, 155-166.

[27] Xing, B., & Van Der Laan, M. J. (2005). A causal inference approach for constructing transcriptional regulatory networks. *Bioinformatics*, *21*, 4007-4013.

[28] Bansal, M., Gatta, G. D., & di Bernardo, D. (2006). Inference of gene regulatory networks and compound mode of action from time course gene expression profiles. *Bioinformatics*, *22*, 815-822.

[29] Pearl, J. (2000). Causality: Models, reasoning, and inference. *Information Management Knowledge Management Business Intelligence Strategy, 75(299),* 49.

[30] Spirtes, P., Glymour, C., & Scheines, R. (2001). *Causation, Prediction, and Search*. Cambridge(2nd ed.). MA, USA: The MIT Press.

[31] Newman, M. E. (2001). Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Physical Review E*, *64*, 016132.

Leung-Yau Lo received his B.Sc. degree in risk management science from the Chinese University of Hong Kong in 2008, where he is currently working towards the Ph.D. degree in the Department of Computer Science and Engineering under the supervision of Prof. K.S. Leung and Prof. K. H. Lee. His research interests include bioinformatics and artificial intelligence.

Man-Leung Wong is an associate professor at the Department of Computing and Decision Sciences at Lingnan University, Tuen Mun, Hong Kong. His research interests are knowledge discovery in databases, machine learning, electronic commerce, evolutionary computation, knowledge acquisition, fuzzy logic, and approximate reasoning. His articles on these topics have been published in *Management Science*, *Decision Support Systems*, *IEEE Transactions on Evolutionary Computation*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *IEEE Transactions on Systems, Man, and Cybernetic*, *IEEE Intelligent Systems*, *IEEE Engineering in Medicine and Biology*, *Expert Systems with Applications*, *Journal of the American Society for Information Science and Technology, Fuzzy Sets and Systems*, *International Journal of Approximate Reasoning*, etc. He received his B.Sc., M.Phil., and Ph.D. in computer science from the Chinese University of Hong Kong in 1988, 1990, and 1995, respectively.

Kin-Hong Lee received the B.S. and M.S. degrees in computer science from the University of Manchester, Manchester, U.K. He was an associate professor with the Department of Computer Science and Engineering, Chinese University of Hong Kong and retired in 2013. His research interests include computer architecture and bioinformatics. He has published over 120 papers in these two fields.

Kwong-Sak Leung received his B.Sc. (Eng.) and Ph.D. degrees in 1977 and 1980, respectively, from the University of London, Queen Mary College. He joined the Computer Science and Engineering Department at the Chinese University of Hong Kong in 1985, where he is currently a professor of Computer Science & Engineering. His research interests are in bioinformatics and soft computing including evolutionary computation, parallel computation, probabilistic search, information fusion and data mining, fuzzy data and knowledge engineering.