Journal of
Health & Medical Informatics

Lo et al., J Health Med Informat 2018, 9:3
DOI: 10.4172/2157-7420.1000315

Research Article Open Access

Decomposition Approach for Learning Large Gene Regulatory Network

Leung-Yau Lo[1*], Man-Leung Wong[2], Kwong-Sak Leung[1], Wing-Lun Lam[2] and Chi-Wai Chung[2]

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

Abstract

Gene Regulatory Network (GRN) represents the complex interaction between Transcription Factors (TFs) and other genes with time delays. They are important in the working of the cell. Learning GRN is an important first step towards understanding the working of the cell and consequently curing diseases related to malfunctioning of the cell. One significant problem in learning GRN is that the available time series expression data is still limited compared to the network size. To alleviate this problem, besides using multiple expression replicates, we propose to decompose large network into small subnetwork without prior knowledge. Our algorithm first infers an initial GRN using CLINDE, then 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 many networks with 500 and 1000 genes. We have also tested on real data on 41 human TF regulatory networks. Results show that our proposed algorithm does improve the GRN learning performance of using either CLINDE or DD-lasso alone on the large network.

Keywords: Gene regulatory network; Causal learning; Time series expression data

Introduction

Learning Gene Regulatory Network (GRN) is an important problem in Bioinformatics. In the cell, the genes produce proteins when transcribed into mRNAs and subsequently translated. Some proteins are Transcription Factors (TFs), which will trigger or inhibit the transcription of other genes. Also involved are miRNAss which inhibit the translations of some mRNAs. Moreover, transcription and translation each takes time, though the two may be done simultaneously (e.g. in bacteria), non-negligible translational time delays have been observed [1,2]. The transcription rate has been observed ranging from 12 to 100 nucleotide per second (nt/s), under different conditions in different organisms [3,4]. Also, it has been observed that RNA polymerase, a main working protein in transcription, may pause during transcription, adding a cumulative of 204-307s over a 2.3 kb region [4]. These delays have been known to affect the network stability, or cause oscillations [5-8]. Therefore, there are complex regulatory relationships in the cell between all these components, with different time delays. In order to understand the working of the cell and subsequently to understand the cause of diseases related to the malfunctioning of the cell (such as cancer), it is crucial that the GRN be first mapped out. It would be too time-consuming and expensive to experimentally determine the regulators and regulatory targets of each TF, therefore computational methods are attractive complementary means to infer the GRN. With the advent of high-throughput microarray and RNA-seqtechnology, it is possible to obtain the expression levels of thousands of genes at one time, and a time series of expression levels of many genes could be obtained when this is done on a number of time points.

With high-throughput experimental techniques, in the time series expression data, the number of genes is usually very large (in the thousands), but the number of time points is usually far smaller (in the tens), which still poses a challenge to inferring a causal GRN with direction and regulatory effects in the edges. Besides 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 would make the cells increasingly unsynchronized in the cell cycles. 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 the problem of limited data is to perform biological replicates, and make the learning algorithm utilize multiple relatively short time series instead of a long one. Both CLINDE and DD-lasso can accept multiple time series. Another possibility, which we explore in this paper, is to decompose a large GRN into smaller possibly overlapping subnetworks, then do GRN learning on each subnetwork, then finally \stitch" the subnetworks to get an overall GRN. This is feasible owing to the sparsity of GRNs. The difficulty lies in decomposing a GRN into subnetworks without prior knowledge of the GRN structure. In this paper, we develop a GRN learning algorithm that aims at inferring larger GRN by first estimating an initial GRN from the provided time series expression data, then decomposing the initial GRN into possibly overlapping subnetworks, infer each subnetwork again from the data, and finally combining the subnetworks. In the rest of the paper, we first give an overview of existing algorithms for learning GRN. In Section 3, our proposed algorithm is discussed. The experiments and the results are reported in Section 4. In Section 5, conclusions and future research are discussed.

Background

There have been many GRN learning algorithms and models, with different levels of details [9,10] for surveys of GRN modelling and [11] for survey on GRN learning algorithms for microarray expression data.

Due to the nature of GRN, most models of GRN could be described as a graph, where the vertices are the genes under consideration, and the edges represent the regulatory relationships.

***Corresponding author:** Leung-Yau Lo, Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, Tel: (852)64887615; E-mail: lylo@cse.cuhk.edu.hk

Received June 08, 2018; **Accepted** June 18, 2018; **Published** June 20, 2018

Citation: Lo LY, Wong ML, Leung KS, Lam WL, Chung CW (2018) Decomposition Approach for Learning Large Gene Regulatory Network. J Health Med Informat 9: 315. doi: 10.4172/2157-7420.1000315

Different levels of details could be achieved by labelling the edges with extra information. In the simplest case, an undirected graph could be used, in which case only an association network of the genes is captured. ARACNE [12] infers undirected network using mutual information, but also uses Data Processing Inequality to try to eliminate indirect interactions. C3NET [13] first identifies gene pairs with significant mutual information, then links each gene to the neighbour (if any) with highest mutual information, and output an undirected and conservative network, with no explicit means of eliminating indirect effect. But without direction in the edges, there is no causal interpretation. Alternatively, directed edges could be used, as in [14], which uses genetic algorithm to optimize a score based on partial correlation, estimated regulatory direction and effect, but the output edges are not labelled with time delays. Some algorithms consider only delay of one time step, as in [15], which considers discretized expression data, and uses association rule mining to find frequent regulatory patterns, but without eliminating indirection association. Boolean network, e.g. in [16], is a classic model of GRN where the expression of each gene is discretized to only *on* or *off*, and the expression of each gene at the next time step is a boolean function of expression of its regulators at the current time step. Another popular class of GRN model is Ordinary Differential Equations (ODE), where the rate of change of expression of each gene is a (linear or nonlinear) function of the expression of the gene and its regulators. When discretized in time, it reduces to one time step model. Examples are [17], which uses Gaussian process for Bayesian inference of an ODE model; and DELDBN [18], which combines ODE model with local Bayesian analysis, and uses estimated markov blanket as the regulators of each gene. There is also Dynamic Bayesian Network (DBN) based models, which avoids the limitation of plain Bayesian network that no cycles are allowed. An example is [19], which utilizes Bayesian structural expectation maximization to infer a onetime step DBN model. There are relatively few algorithms that infer multiple time delays. First estimates the possible delays from pairwise mutual information from discretized expression data, then infers multiple time [20] step DBN by minimizing MDL score using genetic algorithm. Banjo [21] 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 [22] 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 [23], 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 [24] to remove indirect effects and estimate the coefficients, therefore the edges are labelled with the delays as well as the regulatory effects. CLINDE [25] uses a similar model, but uses conditional independence of the shifted time series to estimate the delays and eliminate indirect effects.

Some other algorithms use perturbation data, or use a combination of perturbation data and time series expression is a [26] parallel implementation of the Network Identification by multiple Regression (NIR) algorithm utilizing perturbation data [27] needs promoter sequence and TF binding site information in addition to (non-time series) expression data [28] is an IC [29,30] based method, which uses steady state data, with partial prior knowledge of ordering of regulatory relationship, and uses entropy to test conditional independence, giving an acyclic network where some edges may remain undirected [31] uses convex programming on an ODE model using perturbation data. TSNI [32] solves a discretized linear ODE model using time series expression data after each gene is perturbed (Figure 1).

Methods

In this section, the proposed algorithm is described. We assume the true GRN consists of a number of subnetworks, where most connections are within subnetworks, but each subnetwork is not densely connected, and there are some connections across subnetworks. Figure 1 illustrates the assumed structure of the GRN.

These assumptions may suggest a clustering based method, but there are some issues to address in using plain clustering method. Firstly, the similarity measure needs to take into ac- count the time delays. This could be solved by considering the correlation between shifted time series, and try 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 ! b and b ! c, then a high correlation between a and c will be observed. Since there are some connections across subnetworks, indirect effects may cause more genes in different subnetworks to appear dependent than they actually are, which may make the clustering more difficult. This is illustrated in Figure 2. Thirdly, since the subnetworks are not disconnected, if clustering is used, overlapping" clusters are preferred to disjoint ones, but most clustering methods give disjoint clusters as output. If only disjoint clusters are found, either further processing is needed to find the cross edges between subnet- works or these edges have to be ignored. Therefore, either overlapping clusters are found, or disjoint clusters

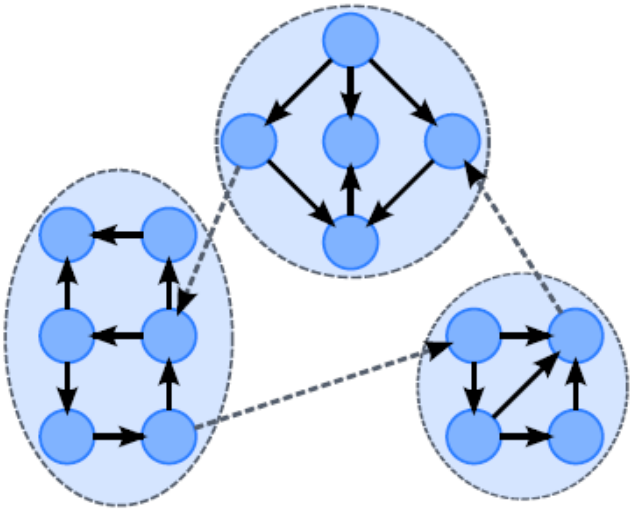

Figure 1: Structure of subnetworks the overall GRN consists of some subnetworks connected by a few links.

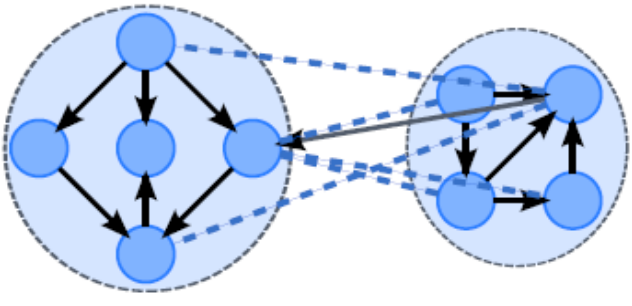

Figure 2: Indirect effect across subnetworks the dotted lines indicate the indirect effects across the two subnetworks.

have to be expanded" a little to allow the cross edges to be inferred in the process of inferring each subnetworks, instead of needing further inference. Because of the above, we propose to first infer an initial GRN using CLINDE, which can handle time delays and help eliminate some indirect effects, then to decompose the initial GRN into overlapping subnetworks, to get the subsets of genes. The over allow is given in Figure 3. The steps are

1) Initial GRN learning using CLINDE

2) Decomposition using edge betweenness to get grouped subsets of genes

3) Each gene subset is used to infer a subnetwork using either CLINDE or DD-lasso

4) The subnetworks are combined to obtain the final GRN. In the following, the input data, GRN model, and the steps of the proposed algorithm are described.

Data and model

The given data is $\{x_i(t)g$, for i=1,…, n, t=1,…,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. If the raw input data does 5 not have equidistant time points, interpolation (e.g. spline interpolation) could be performed as pre-processing before using this algorithm.

The GRN model assumed here is the same as that for DD-lasso [23] and CLINDE [25]:

$$x_j(t) = \sum a_{ij} x_i(T_{ij}) + \in_j(t)$$

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 T_{ij} is the positive time delay of the edge i→ j (if a_{ij} /= 0); and $\epsilon_j(t)$ is the error term for gene j at time t. We assume that the error terms of each gene and at each time point 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 does not preclude self-regulation or cycles in the GRN, though any cycles must have positive delays.

Initial GRN

The first step is to obtain an initial GRN. There are not many GRN learning algorithms that handles multiple time delays, CLINDE [25] and DD-lasso [23] are two of them. And from the comparison in [25], CLINDE outperforms DD-lasso when the number of time points is small relative to the number of genes, therefore we choose CLINDE to infer an initial GRN. CLINDE is based on the PC algorithm [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 allowed delay, to determine if $x \rightarrow y$ 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 correlation test and partial correlation test below, the score is -log10 (p-value), and in the mutual information test and conditional mutual information test below, the score is the (conditional) mutual information. So a higher score threshold means a more stringent test. And for correlation test, the regulatory effect (positive or negative) is also estimated if the edge is significant. 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 neighbour, then h=2 neighbours, and so on up to h=N0, for a given parameter N0. In each iteration, each remaining edge is tested by conditioning on h neighbours, shifted properly using the delays estimated in stage 1. If the conditional test is not significant, the edge is pruned. After stage 2, the remaining edges are output as the GRN. For the purpose of finding the decomposition of subnetworks, we \condense" the multiple edges x→y with different delays between the same pair of genes so that only the one with the most significant p-value in stage 1 remains (Figures 4 and 5).

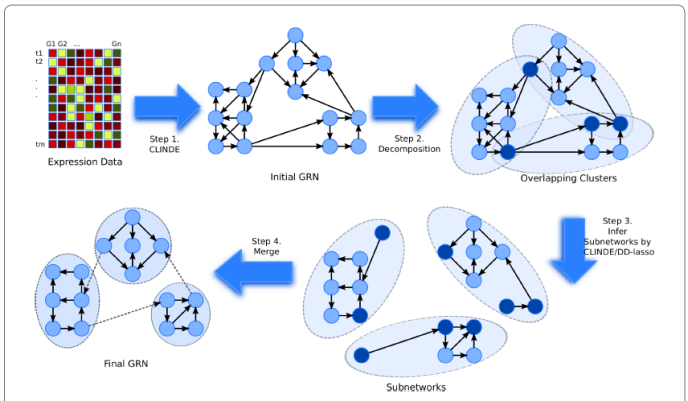

Figure 3: Overall Flow of the Algorithm The steps are 1) Initial GRN learning using CLINDE, 2) Decomposition into overlapping clusters, 3) Infer each subnetworks by CLINDE/DD-lasso, 4) Merge subnetworks.

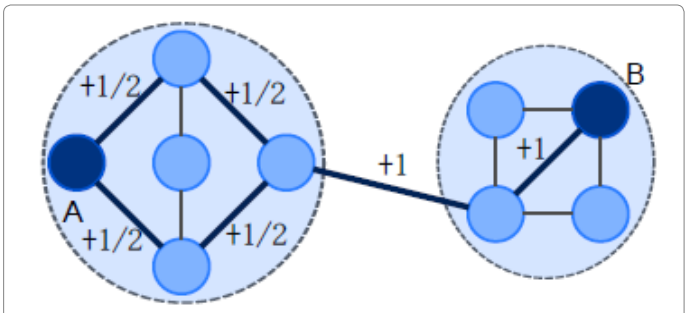

Figure 4: Contribution of Vertices Pairs to Edge Betweenness For each pair of vertices (A and B in the example), the shortest path(s) between them contributes to the edge betweenness. Multiple paths will share the weight equally.

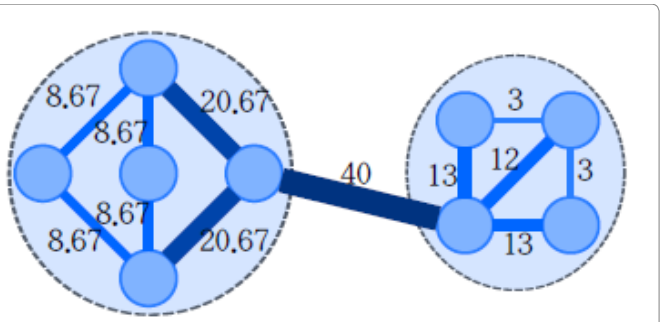

Figure 5: Example of Edge Betweenness An example of edge betweenness calculated for each edge. The edge across subnetworks will get higher edge betweenness because the shortest path(s) of vertices in different subnetworks have to go through that edge.

Decomposition

Given an initial GRN, the proposed strategy of decomposing it into subnetworks is to identify which edges are likely to be across subnetworks. This is accomplished based on "edge between-ness" [33], as used in community structure discovery algorithms [33]. In a graph, for every two distinct vertices (genes), consider the undirected shortest path(s) between them. If there are k shortest paths, for each shortest path, its constituent edges each receives a weight of $1=k$ from the path. The edge betweenness" for an edge is the sum of weights received after considering all vertex pairs. This is illustrated in Figure 4. Intuitively, a higher edge betweenness means the edge is more likely to be across subnetworks, this is illustrated in Figure 5. Consider subnetworks connected by only a few edges, the shortest paths of gene pairs in the same subnetwork likely stay within the subnetwork and would not overlap too much. On the other hand, for genes not in the same subnetwork, their shortest paths would need to go through one of the few edges across subnetworks. Therefore, the edges across subnetworks would likely have higher edge betweenness [34] gives a fast method to calculate edge betweenness of a graph with m edges and n vertices in $O(mn)$ time.

The steps for decomposition are as follows:

1. Starting from the initial GRN, first identify all the components (the maximally connected sub graphs, considering the edges as undirected).

2. For each component, if it is larger than or equal to a size threshold S_0, calculate the edge betweenness, and remove the one with the highest edge betweenness.

3. Go back to step 2 until all components are small than S_0.

For the choice of size threshold S_0, it should not be set too small, because this decomposition procedure is only a heuristic, and removing more edges means higher chance that the genes are grouped wrongly. Also, even if S_0 is small, the resulting overlapping subnetworks may still be larger than desired after expansion below. On the other hand, S_0 should not be too large, as the subsequent subnetwork learning algorithm may have difficulty dealing with large network, given the limited data. As a balance, the choice should be a large value that the subnetwork learning algorithm can still comfortably handle, taking into account the possible expansion of the gene subset discussed below. The default threshold S_0 used is 60, based on the previous performance of CLINDE and DD-lasso. Having identified the components, 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 of the genes, so is used mainly for comparison purpose, i.e. the performance of the algorithm if the predicted cross edges are ignored.

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 learning, the edges between the parents (not in the component) are removed. The rationale is that the parents of the parents may not be included in the subnetwork, so some indirect effects between the parents may not be removed properly. Also, if there is genuine edge between two parents, presumably these two parents would have been present in some component.

Infer subnetworks

After the possibly overlapping subsets of genes for each subnetwork (either one of Component, Parents or XParents) have been estimated from the decomposition of the initial GRN, 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, as both could handle the now small subnetworks.

DD-lasso is based on lasso [24], 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 $x \rightarrow y$ 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, by using the expression data of g as target and the shifted expression data of all the genes as predictors. In stage 3, backward elimination is done for each gene to further remove parents that are likely only indirect effects. Finally the output GRN is obtained. 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 CLINDE again to infer the subnetworks may not lead to great improvement, since the learning method is not changed much, unless the learning parameters are adjusted based on the network sizes. In contrast, DD-lasso is based on different approach from CLINDE, and has different performance characteristics, so it may lead to greater improvement on the subnetworks, even though DD-lasso may not perform well on the large network, given the limited data.

Merge the subnetworks

After the subnetworks are re-learnt using either CLINDE or DD-lasso, the subnetworks 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.

Experiment Results

This section evaluates the effectiveness of our proposed algorithm. Since it is difficult to find known large gene networks and sufficiently long time series data of the involved genes, we mainly rely on synthetic data, where we know the underlying gene network, and there is no lack of sufficient expression data. On the other hand, it is useful to evaluate the proposed algorithm on real data, even though the time series data may not be long enough or have high enough quality. For this purpose we use the human transcription factor regulatory networks from [35] as the known GRN and use GDS4238 obtained from Gene Expression Omnibus as expression data.

Performance metrics

We first describe the performance metrics used in evaluating the proposed algorithm. It is more appropriate to focus on correctly predicting the presence of links rather than its absence, due to the sparse nature of GRNs. Same as in [25], we assess the performance of the learning algorithm on three aspects, namely *Links* (which is considered correct if and only if both the gene pair and the direction are correct), *Delays* (which is considered correct if and only if both the link and the time delay *_ij* are correct) and *Effects* (which is considered correct if and only if both the link and the sign of the effect *aij* are correct). For each aspect, we look at three metrics respectively, namely

Recall=TP

TP+FN, *Precision=TP*

TP+FP and *F-score=2_Recall_Precision*

Recall+Precision, where

TP is the number of *true positives*, *FP* is the number of *false positives*, and *FN* is the number of *false negatives*.

Synthetic data generation

Generation of network: We first describe how to generate a small network of size *k*, where each gene has a maximum of *M*0 parents, and the maximum delay is _0. We represent a network of size *k* as a pair of *k* by *k* matrices ($[a_{ij}]$; $[T_{ij}]$), where a_{ij}, if non-zero, is the coefficient of the link $i \rightarrow j$, with the associated time delay T_{ij} and a_{ij} is 0 if there is no link from *i* to *j*. For each gene *i*, 1 _ *i* _ *k*, we

1. Uniformly choose the number of parents s_i from $f_1,\ldots,M_{0g}$

2. Randomly choose its parents P_i _$f_1,\ldots,kg$ where $/P_i/=s_i$

3. For $u \in P_i$, set $a_{iu}=\rho_{iu}z_{iu}$, where ρ_{iu} is uniformly chosen from {-1,1} and z_{iu} is uniformly chosen from (0.5, 1.5)

4. For $u \in P_i$, uniformly choose T_{iu} from $\{1,\ldots,T_O\}$

5. For $u \in P_i$, set $a_{iu}=0$ and $T_{iu}=0$.

6. Lastly, scale the coefficients to make the network stable.

We generate a large network of *n* genes with the structure illustrated in Figure 1, where each subnetwork has size n_0, as follows:

Generate $K=\left[\frac{n}{n_0}\right]$ subnetworks $\left\{\left(\left[a_{ij}^{(u)}\right],\left[T_{ij}^{(u)}\right]\right)\right\}_{1\le u\le K}$ each with size n_0 as above

Generate a subnetwork $\left(\left[a_{ij}^{(0)}\right],\left[T_{ij}^{(0)}\right]\right)$ of size *K*

Set $A'=\left[a'_{ij}\right] and\ T'=\left[T'_{ij}\right]$ as:

For $1\le u\le K$ set $a'_{pq}=a_{ij}^{(u)}$ and $T'_{pq}=T_{ij}^{(u)}$ where

$P=(u-1)n_0+i, q=(u-1)n_0+j$

For each $a_{ij}^{(0)}\neq 0$ set $a'_{pq}=a_{ij}^{(0)}$ and $T'_{pq}=T_{ij}^{(0)}$ where *p* is uniformly chosen from $\{(i-1)n_0+1,\ldots,(i-1)n_0+n_0\}$ and *q* is uniformly chosen from $\{(j-1)n_0+1,\ldots,(j-1)n_0+n_0\}$

$a'_{ij}=T_{ij}=0$ otherwise

1. Generate a random permutation π of {1,…,n}

Output the final network as $\left(\left[a'_{\pi(i),\pi(j)}\right]\le i,j\le n,\left[T'_{\pi(i),\pi(j)}\right]1\le i,j\le n\right)$

Essentially we first generate $\left[\frac{n}{n_0}\right]$ subnetworks, each with size n_0 and then generate a network of size $\left[\frac{n}{n_0}\right]$ to connect them. And the final permutation is to prevent the subnetworks from being easily identified from the indices.

Generation of expression data: Having generated the network, we obtain a network as ($a_{ij,ij}$). Given the parameters 2 (which controls the gaussianity of error terms), we then generate the expression data with *m* time points as follows:

1. For $-T_0<t\le 0, 1\le j\le n$, set $x_j(t)=e_j(t)$ where each $e_j(t)$ is generated from *N*(0, 1)

2. For $1\le t\le m,\ 1\le j\le n$ set $x_j(t)=\sum_{i=1}^{n}a_{ij}x_i(t-T_{ij})+e_j(t)$ where $e_j(t)=sign(z_{jt})|z_{jt}|^{\alpha}$

3. Output $\{x_j(t)\}1\le t\le m, 1\le j\le n$

Settings: There are a number of parameters controlling the generation of the synthetic networks: network size (*n*), subnetwork size (*n*0), σ affecting the variance of error terms, α controlling the gaussianity of the error terms, maximum number of parents M_0, maximum delay T_0, number of time points (*m*). For some parameters, we test different values listed in Table 1. For each setting, we generate 20 random replicates; and for each replicate, we first generate time series of 200 points and then take prefix of different lengths to give different number of time points. So there are a total of 480 networks, and 1920 time series.

Also, there are parameters for CLINDE for learning initial GRN and subnetworks. We use the default PC or method, and tried the score thresholds (use the same one for both stage 1 and stage 2) 2, 3, 4. For other parameters, we use the default. And after obtaining the initial GRN, we tried three types of decompositions into subsets of genes: *Component*, *Parents* and *XParents*.

Synthetic Data Results

Results of CLINDE alone on the large networks: Tables 2 and 3 show the median F-score of CLINDE alone on *Links*, *Delays* and *Effects* on the *n*=500 and *n*=1000 networks respectively, where the medians are taken over 20 replicates. The performance of using score thresholds of 2, 3 and 4 are not too different for different settings, but for small number of time points (*m*=20 and *m*=50), using a score threshold of 2 is usually slightly better than 3 and 4; and for more time points (*m*=100 and *m*=200), using a score threshold of 3 is usually better, though for *m*=200 and α=3 it is sometimes better to use

4. Therefore, in the following, if space is limited, we show the results of score threshold 3 for CLINDE.

For a given *m* and α, the F-score usually decreases gradually as σ increases, but the drop is less prominent for larger *m*. Also, the difference in F-score for different α is relatively large only for very small number of time points (*m*=20).

Furthermore, the performance for *Links*, *Delays* and *Effects* are similar for each setting. This suggests that it is more usual to predict both the delay and effect for a link correctly, rather than only getting one correct but not the other. Therefore, in the following, we focus mainly on the performance of *Effects*.

Results of CLINDE and DD-lasso alone on the large networks: Table 4 and 5 show the median effect performance of using CLINDE and DD-lasso alone on the large network of size *n*=500 and *n*=1000

Parameter	Value(s)
Network size (*n*)	500, 1000
Subnetwork size (n_0)	50
σ	0.5, 2, 8
α	0.5, 1, 2, 3
Maximum parents (M_0)	4
Maximum delay (τ_0)	4
Time points (*m*)	20, 50, 100, 200
Replicates	20

Table 1: Parameter settings of synthetic data generation.

n	m	α	σ	Links			Delays			Effects	
				st2	st3	st4	st2	st3	st4	st2	st3	st4
500	20	0.5	0.5	0.032	0.029	0.018	0.031	0.027	0.017	0.031	0.028	0.018
			2	0.026	0.021	0.011	0.022	0.019	0.010	0.024	0.020	0.011
			8	0.024	0.020	0.011	0.021	0.018	0.011	0.022	0.019	0.011
		1	0.5	0.043	0.040	0.025	0.041	0.039	0.025	0.042	0.039	0.025
			2	0.030	0.025	0.017	0.027	0.023	0.015	0.028	0.024	0.016
			8	0.028	0.027	0.015	0.027	0.025	0.015	0.027	0.025	0.015
		2	0.5	0.067	0.063	0.052	0.066	0.061	0.051	0.066	0.062	0.051
			2	0.042	0.038	0.029	0.039	0.035	0.028	0.040	0.036	0.029
			8	0.045	0.042	0.034	0.043	0.042	0.033	0.044	0.042	0.034
		3	0.5	0.088	0.082	0.075	0.087	0.082	0.072	0.088	0.082	0.073
			2	0.063	0.061	0.055	0.062	0.060	0.054	0.062	0.060	0.054
			8	0.066	0.061	0.054	0.064	0.060	0.052	0.065	0.060	0.053
	50	0.5	0.5	0.243	0.233	0.204	0.238	0.232	0.204	0.239	0.233	0.204
			2	0.228	0.226	0.196	0.225	0.223	0.194	0.226	0.223	0.194
			8	0.221	0.217	0.183	0.218	0.215	0.183	0.220	0.215	0.183
		1	0.5	0.252	0.250	0.221	0.250	0.248	0.220	0.251	0.250	0.220
			2	0.234	0.229	0.202	0.232	0.226	0.200	0.232	0.228	0.201
			8	0.223	0.221	0.193	0.221	0.219	0.193	0.222	0.220	0.193
		2	0.5	0.267	0.262	0.238	0.265	0.261	0.236	0.266	0.261	0.237
			2	0.244	0.238	0.214	0.240	0.236	0.212	0.242	0.237	0.213
			8	0.240	0.239	0.216	0.238	0.237	0.216	0.239	0.238	0.216
		3	0.5	0.275	0.263	0.245	0.271	0.262	0.243	0.272	0.263	0.244
			2	0.243	0.234	0.215	0.241	0.233	0.214	0.242	0.234	0.215
			8	0.257	0.244	0.218	0.254	0.242	0.217	0.255	0.243	0.218
	100	0.5	0.5	0.511	0.530	0.495	0.509	0.530	0.494	0.511	0.530	0.494
			2	0.507	0.532	0.496	0.505	0.529	0.495	0.506	0.530	0.495
			8	0.502	0.522	0.485	0.500	0.521	0.485	0.501	0.522	0.485
		1	0.5	0.525	0.546	0.504	0.524	0.545	0.503	0.525	0.546	0.504
			2	0.507	0.537	0.495	0.505	0.536	0.495	0.506	0.537	0.495
			8	0.505	0.534	0.491	0.503	0.532	0.491	0.504	0.533	0.491
		2	0.5	0.514	0.529	0.500	0.512	0.527	0.500	0.514	0.529	0.500
			2	0.490	0.509	0.483	0.489	0.509	0.482	0.489	0.509	0.482
			8	0.489	0.512	0.485	0.488	0.511	0.484	0.488	0.511	0.485
		3	0.5	0.464	0.462	0.438	0.461	0.460	0.437	0.462	0.461	0.437
			2	0.429	0.436	0.417	0.428	0.435	0.417	0.429	0.435	0.417
			8	0.441	0.446	0.428	0.439	0.444	0.426	0.440	0.445	0.427
	200	0.5	0.5	0.705	0.773	0.743	0.704	0.773	0.743	0.704	0.773	0.743
			2	0.696	0.769	0.747	0.696	0.769	0.747	0.696	0.769	0.747
			8	0.700	0.768	0.742	0.700	0.768	0.742	0.700	0.768	0.742
		1	0.5	0.710	0.772	0.746	0.709	0.772	0.746	0.709	0.772	0.746
			2	0.701	0.776	0.746	0.701	0.775	0.746	0.701	0.775	0.746
			8	0.701	0.768	0.743	0.700	0.768	0.743	0.701	0.768	0.743
		2	0.5	0.693	0.759	0.737	0.691	0.759	0.737	0.692	0.759	0.737
			2	0.682	0.748	0.734	0.682	0.747	0.734	0.682	0.747	0.734
			8	0.679	0.746	0.730	0.679	0.745	0.730	0.679	0.746	0.730
		3	0.5	0.594	0.644	0.644	0.592	0.643	0.643	0.592	0.643	0.643
			2	0.577	0.629	0.630	0.577	0.628	0.630	0.577	0.629	0.630
			8	0.582	0.635	0.636	0.581	0.634	0.636	0.582	0.635	0.636

Table 2: Median F-score of CLINDE on the n=500 Networks. The medians are taken over the 20 replicates. st2, st3 and st4 are for score thresholds of 2, 3 and 4 respectively.

respectively, where the medians are taken over 20 replicates. In Tables 4 and 5, the score threshold for CLINDE is 3. S1 and S2 Table show more detailed median performances.

Firstly, when the number of time points is small (m=20), both CLINDE and DD-lasso per- form badly, where DD-lasso is slightly better, but the performance is too poor to be conclusive.

As m increases, CLINDE performs increasingly well, but for DD-lasso, while the recall increases quickly, the precision does not improve much or even decrease, leading to poor F-score. And for $m \geq 50$, the F-score of CLINDE is better than DD-lasso.

n	*m*	*α*	*σ*	Links st2	Links st3	Links st4	Delays st2	Delays st3	Delays st4	Effects st2	Effects st3	Effects st4
1000	20	0.5	0.5	0.021	0.019	0.015	0.020	0.018	0.015	0.020	0.019	0.015
			2	0.016	0.014	0.010	0.015	0.013	0.009	0.015	0.013	0.009
			8	0.015	0.013	0.010	0.014	0.012	0.009	0.014	0.013	0.009
		1	0.5	0.027	0.025	0.019	0.025	0.023	0.019	0.026	0.023	0.019
			2	0.020	0.017	0.013	0.018	0.016	0.013	0.019	0.016	0.013
			8	0.019	0.016	0.012	0.018	0.016	0.012	0.018	0.016	0.012
		2	0.5	0.050	0.045	0.042	0.049	0.044	0.041	0.049	0.045	0.041
			2	0.030	0.028	0.025	0.029	0.027	0.024	0.030	0.027	0.025
			8	0.032	0.030	0.026	0.031	0.029	0.025	0.031	0.029	0.026
		3	0.5	0.072	0.067	0.062	0.070	0.066	0.061	0.071	0.067	0.062
			2	0.049	0.047	0.043	0.048	0.047	0.042	0.048	0.047	0.043
			8	0.048	0.046	0.042	0.047	0.045	0.042	0.047	0.046	0.042
	50	0.5	0.5	0.206	0.202	0.189	0.204	0.201	0.188	0.205	0.201	0.189
			2	0.189	0.185	0.170	0.188	0.184	0.170	0.188	0.185	0.170
			8	0.183	0.183	0.173	0.182	0.182	0.173	0.183	0.182	0.173
		1	0.5	0.212	0.212	0.194	0.212	0.211	0.193	0.212	0.212	0.194
			2	0.192	0.193	0.179	0.192	0.192	0.179	0.192	0.193	0.179
			8	0.185	0.190	0.176	0.183	0.189	0.174	0.184	0.189	0.175
		2	0.5	0.231	0.225	0.211	0.230	0.225	0.211	0.230	0.225	0.211
			2	0.196	0.193	0.180	0.195	0.193	0.179	0.195	0.193	0.179
			8	0.205	0.200	0.186	0.204	0.198	0.185	0.205	0.199	0.185
		3	0.5	0.235	0.227	0.212	0.233	0.225	0.210	0.234	0.226	0.211
			2	0.215	0.205	0.191	0.214	0.204	0.191	0.214	0.205	0.191
			8	0.214	0.206	0.190	0.213	0.205	0.190	0.213	0.205	0.190
	100	0.5	0.5	0.467	0.500	0.486	0.466	0.499	0.485	0.466	0.499	0.485
			2	0.452	0.493	0.471	0.451	0.492	0.471	0.452	0.492	0.471
			8	0.458	0.489	0.471	0.457	0.488	0.471	0.458	0.489	0.471
		1	0.5	0.469	0.497	0.482	0.469	0.497	0.482	0.469	0.497	0.482
			2	0.462	0.493	0.483	0.461	0.492	0.482	0.461	0.492	0.482
			8	0.452	0.487	0.474	0.452	0.487	0.474	0.452	0.487	0.474
		2	0.5	0.457	0.484	0.469	0.456	0.483	0.468	0.457	0.484	0.469
			2	0.433	0.456	0.445	0.433	0.455	0.444	0.433	0.456	0.444
			8	0.441	0.469	0.455	0.441	0.468	0.455	0.441	0.468	0.455
		3	0.5	0.410	0.420	0.401	0.409	0.419	0.400	0.410	0.420	0.401
			2	0.400	0.407	0.389	0.399	0.407	0.388	0.400	0.407	0.388
			8	0.401	0.407	0.385	0.400	0.406	0.385	0.400	0.407	0.385
	200	0.5	0.5	0.663	0.750	0.739	0.662	0.750	0.739	0.663	0.750	0.739
			2	0.650	0.737	0.732	0.649	0.737	0.732	0.649	0.737	0.732
			8	0.651	0.740	0.738	0.651	0.740	0.738	0.651	0.740	0.738
		1	0.5	0.655	0.742	0.737	0.654	0.742	0.737	0.654	0.742	0.737
			2	0.657	0.743	0.739	0.657	0.743	0.739	0.657	0.743	0.739
			8	0.650	0.741	0.735	0.650	0.741	0.735	0.650	0.741	0.735
		2	0.5	0.640	0.720	0.722	0.640	0.719	0.721	0.640	0.719	0.721
			2	0.629	0.714	0.716	0.629	0.713	0.716	0.629	0.714	0.716
			8	0.629	0.712	0.714	0.629	0.711	0.714	0.629	0.712	0.714
		3	0.5	0.550	0.603	0.612	0.550	0.602	0.612	0.550	0.603	0.612
			2	0.539	0.595	0.607	0.538	0.594	0.607	0.539	0.595	0.607
			8	0.541	0.596	0.601	0.540	0.596	0.601	0.541	0.596	0.601

Table 3: Median F-score of CLINDE on the n = 1000 Networks. The medians are taken over the 20 replicates. st2, st3 and st4 are for score thresholds of 2, 3 and 4 respectively.

Secondly, for both CLINDE and DD-lasso, for a given m, there is only slight difference in F-score for different σ and α, except for m=200 where the drop in F-score is more prominent for CLINDE for α=3. This shows that both CLINDE and DD-lasso are quite robust to slight deviation from gaussian error terms, and quite robust to different variance of error terms.

Results of CLINDE and DD-lasso by first decomposing the large networks: Tables 6 and 7 show the median effects F-score of first inferring an initial GRN by CLINDE (referred to as CL-init below), then decompose the initial GRN (*Component*, *Parents* and *XParents*), then use CLINDE and DD-lasso to infer the subnetworks (referred to as CL-sub and DD-sub respectively), and finally merge the subnetworks. The score threshold for

					CL-init			DD	
n	*m*	*α*	*σ*	Recall	Precision	F-score	Recall	Precision	F-score
500	20	0.5	0.5	0.018	0.058	0.028	0.047	0.027	0.034
			2	0.013	0.040	0.020	0.037	0.022	0.027
			8	0.013	0.040	0.019	0.036	0.022	0.027
		1	0.5	0.026	0.080	0.039	0.052	0.030	0.038
			2	0.016	0.051	0.024	0.043	0.024	0.031
			8	0.016	0.053	0.025	0.041	0.024	0.031
		2	0.5	0.041	0.127	0.062	0.087	0.047	0.061
			2	0.024	0.073	0.036	0.062	0.035	0.045
			8	0.028	0.085	0.042	0.067	0.037	0.048
		3	0.5	0.055	0.161	0.082	0.115	0.062	0.080
			2	0.041	0.119	0.060	0.086	0.051	0.064
			8	0.040	0.117	0.060	0.092	0.056	0.070
	50	0.5	0.5	0.164	0.402	0.233	0.357	0.069	0.117
			2	0.158	0.381	0.223	0.345	0.068	0.113
			8	0.152	0.369	0.215	0.331	0.067	0.112
		1	0.5	0.176	0.426	0.250	0.377	0.070	0.118
			2	0.163	0.387	0.228	0.341	0.066	0.111
			8	0.156	0.372	0.220	0.333	0.067	0.111
		2	0.5	0.187	0.434	0.261	0.439	0.075	0.128
			2	0.171	0.386	0.237	0.394	0.070	0.119
			8	0.171	0.395	0.238	0.400	0.071	0.121
		3	0.5	0.195	0.402	0.263	0.511	0.081	0.140
			2	0.174	0.357	0.234	0.481	0.078	0.133
			8	0.179	0.372	0.243	0.485	0.081	0.139
	100	0.5	0.5	0.418	0.733	0.530	0.753	0.063	0.117
			2	0.417	0.734	0.530	0.746	0.062	0.114
			8	0.409	0.724	0.522	0.741	0.062	0.114
		1	0.5	0.427	0.754	0.546	0.751	0.062	0.114
			2	0.420	0.735	0.537	0.752	0.062	0.114
			8	0.419	0.732	0.533	0.749	0.062	0.114
		2	0.5	0.417	0.714	0.529	0.775	0.061	0.114
			2	0.410	0.686	0.509	0.768	0.060	0.111
			8	0.406	0.685	0.511	0.761	0.060	0.111
		3	0.5	0.386	0.572	0.461	0.808	0.060	0.111
			2	0.367	0.529	0.435	0.788	0.058	0.108
			8	0.378	0.543	0.445	0.792	0.058	0.108
	200	0.5	0.5	0.687	0.886	0.773	0.927	0.038	0.073
			2	0.683	0.882	0.769	0.935	0.038	0.073
			8	0.679	0.879	0.768	0.926	0.038	0.073
		1	0.5	0.683	0.885	0.772	0.930	0.039	0.075
			2	0.687	0.880	0.775	0.933	0.038	0.072
			8	0.683	0.878	0.768	0.932	0.037	0.072
		2	0.5	0.680	0.859	0.759	0.930	0.041	0.078
			2	0.668	0.844	0.747	0.933	0.038	0.073
			8	0.670	0.841	0.746	0.933	0.038	0.073
		3	0.5	0.627	0.663	0.643	0.930	0.043	0.082
			2	0.618	0.644	0.629	0.928	0.042	0.080
			8	0.621	0.646	0.635	0.931	0.042	0.080

Table 4: Median Effect Performance of CLINDE and DD-lasso on the n = 500 Net-Works. Score threshold for CLINDE is 3. The medians are taken over the 20 replicates. CL-init is CLINDE, and DD is DD-lasso.

CLINDE is 3 for both the initial GRN and subnetworks. Table 6 shows the results for n=500 networks, while Table 7 shows the results for n=1000 networks. We also perform one-sided Wilcoxon signed-rank test to test whether the median F-score of CL-sub and DD-sub are better than CL-init for m=100 and m=200. The p-values are shown in Table 8, where the entries larger than 0.1 are omitted.

For small number of time points (m=20 and m=50), CL-init is better than CL-sub. As m increases, the performance of CL-sub gets closer to but is still worse than CL-init for m <100. For m=100 and m=200, the performance of CL-sub is comparable to or sometimes slightly better than CL-init.

n	*m*	*α*	*σ*	CL-init Recall	CL-init Precision	CL-init F-score	DD Recall	DD Precision	DD F-score
1000	20	0.5	0.5	0.012	0.038	0.019	0.029	0.018	0.022
			2	0.009	0.027	0.013	0.024	0.015	0.018
			8	0.008	0.025	0.013	0.021	0.014	0.017
		1	0.5	0.015	0.047	0.023	0.035	0.020	0.026
			2	0.011	0.032	0.016	0.027	0.016	0.020
			8	0.011	0.032	0.016	0.024	0.015	0.019
		2	0.5	0.030	0.090	0.045	0.063	0.035	0.045
			2	0.018	0.054	0.027	0.043	0.025	0.032
			8	0.020	0.057	0.029	0.044	0.025	0.032
		3	0.5	0.045	0.131	0.067	0.088	0.050	0.063
			2	0.032	0.089	0.047	0.067	0.041	0.051
			8	0.031	0.089	0.046	0.066	0.040	0.050
	50	0.5	0.5	0.146	0.330	0.201	0.291	0.063	0.104
			2	0.132	0.301	0.185	0.267	0.059	0.097
			8	0.133	0.294	0.182	0.262	0.060	0.098
		1	0.5	0.153	0.341	0.212	0.298	0.062	0.103
			2	0.141	0.308	0.193	0.276	0.062	0.101
			8	0.136	0.310	0.189	0.273	0.061	0.100
		2	0.5	0.164	0.357	0.225	0.357	0.067	0.113
			2	0.142	0.303	0.193	0.316	0.063	0.105
			8	0.147	0.308	0.199	0.322	0.064	0.107
		3	0.5	0.169	0.343	0.226	0.439	0.075	0.128
			2	0.154	0.303	0.205	0.417	0.073	0.124
			8	0.154	0.307	0.205	0.414	0.072	0.124
	100	0.5	0.5	0.401	0.663	0.499	0.704	0.067	0.122
			2	0.394	0.651	0.492	0.678	0.066	0.120
			8	0.393	0.645	0.489	0.687	0.066	0.121
		1	0.5	0.398	0.663	0.497	0.699	0.065	0.118
			2	0.398	0.651	0.492	0.692	0.066	0.120
			8	0.392	0.642	0.487	0.686	0.066	0.120
		2	0.5	0.393	0.628	0.484	0.723	0.061	0.112
			2	0.374	0.586	0.456	0.702	0.061	0.112
			8	0.384	0.600	0.468	0.708	0.061	0.112
		3	0.5	0.360	0.505	0.420	0.768	0.056	0.105
			2	0.352	0.479	0.407	0.765	0.056	0.104
			8	0.353	0.477	0.407	0.762	0.056	0.104
	200	0.5	0.5	0.680	0.839	0.750	0.927	0.043	0.081
			2	0.669	0.822	0.737	0.921	0.043	0.082
			8	0.673	0.826	0.740	0.924	0.043	0.082
		1	0.5	0.670	0.834	0.742	0.925	0.043	0.081
			2	0.673	0.832	0.743	0.927	0.043	0.081
			8	0.670	0.823	0.741	0.921	0.043	0.082
		2	0.5	0.661	0.793	0.719	0.927	0.040	0.077
			2	0.656	0.780	0.714	0.919	0.040	0.077
			8	0.655	0.781	0.712	0.926	0.040	0.077
		3	0.5	0.605	0.601	0.603	0.928	0.037	0.070
			2	0.603	0.588	0.595	0.926	0.036	0.069
			8	0.605	0.588	0.596	0.928	0.036	0.069

Table 5: Median Effect Performance of CLINDE and DD-lasso on the n = 1000 Networks. Score threshold for CLINDE is 3. The medians are taken over the 20 replicates. CL-init is CLINDE, and DD is DD-lasso.

For DD-sub, for small m, the performance is poor. Presumably this is the result of combined effects of poor decomposition because of poor initial GRN, and that even after decomposing the initial GRN into (possibly overlapping) subnetworks, the subnetworks are still too large for DD-lasso to handle when the number of time points is small. Although the decomposition stops only when a component size is smaller than 60, *Parents* and *XParents* include also the parents of the component, the subnetwork size for *Parents* and *XParents* may be larger than 60. From our preliminary analysis, the subnetwork sizes are around 100. And from our experience and results in [25], for a network of size k, DD-lasso needs at least k to $2k$ time points to perform decently. Therefore, when m=100, DD-sub becomes comparable to CL-sub and

						Component		Parents		XParents	
n	*m*	*α*	*σ*	CL-init	DD	CL-sub	DD-sub	CL-sub	DD-sub	CL-sub	DD-sub
500	20	0.5	0.5	0.028	0.034	0.015	0.012	0.015	0.012	0.015	0.013
			2	0.020	0.027	0.010	0.011	0.010	0.011	0.010	0.011
			8	0.019	0.027	0.009	0.011	0.009	0.011	0.009	0.012
		1	0.5	0.039	0.038	0.021	0.019	0.021	0.018	0.021	0.019
			2	0.024	0.031	0.013	0.013	0.013	0.013	0.013	0.012
			8	0.025	0.031	0.014	0.013	0.014	0.013	0.014	0.013
		2	0.5	0.062	0.061	0.036	0.030	0.036	0.029	0.036	0.029
			2	0.036	0.045	0.021	0.018	0.021	0.018	0.021	0.018
			8	0.042	0.048	0.024	0.020	0.024	0.020	0.024	0.020
		3	0.5	0.082	0.080	0.046	0.039	0.046	0.039	0.046	0.041
			2	0.060	0.064	0.037	0.029	0.037	0.029	0.037	0.029
			8	0.060	0.070	0.038	0.031	0.038	0.031	0.038	0.030
	50	0.5	0.5	0.233	0.117	0.210	0.055	0.222	0.057	0.221	0.059
			2	0.223	0.113	0.204	0.051	0.215	0.056	0.214	0.056
			8	0.215	0.112	0.195	0.049	0.203	0.047	0.204	0.049
		1	0.5	0.250	0.118	0.220	0.057	0.236	0.057	0.234	0.060
			2	0.228	0.111	0.197	0.050	0.210	0.053	0.211	0.054
			8	0.220	0.111	0.194	0.048	0.205	0.051	0.203	0.053
		2	0.5	0.261	0.128	0.234	0.053	0.253	0.062	0.252	0.063
			2	0.237	0.119	0.210	0.054	0.225	0.056	0.225	0.060
			8	0.238	0.121	0.218	0.050	0.230	0.057	0.229	0.059
		3	0.5	0.263	0.140	0.238	0.070	0.258	0.068	0.259	0.073
			2	0.234	0.133	0.209	0.066	0.228	0.072	0.230	0.078
			8	0.243	0.139	0.217	0.065	0.235	0.071	0.236	0.073
	100	0.5	0.5	0.530	0.117	0.488	0.439	0.534	0.500	0.534	0.496
			2	0.530	0.114	0.479	0.426	0.532	0.478	0.532	0.477
			8	0.522	0.114	0.469	0.416	0.531	0.485	0.528	0.478
		1	0.5	0.546	0.114	0.501	0.457	0.549	0.534	0.548	0.530
			2	0.537	0.114	0.494	0.452	0.542	0.487	0.540	0.487
			8	0.533	0.114	0.480	0.425	0.539	0.498	0.536	0.491
		2	0.5	0.529	0.114	0.471	0.449	0.527	0.467	0.525	0.466
			2	0.509	0.111	0.449	0.420	0.511	0.396	0.508	0.404
			8	0.511	0.111	0.457	0.433	0.509	0.411	0.509	0.420
		3	0.5	0.461	0.111	0.386	0.366	0.449	0.128	0.456	0.164
			2	0.435	0.108	0.363	0.336	0.417	0.154	0.431	0.188
			8	0.445	0.108	0.375	0.355	0.425	0.151	0.441	0.184
	200	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

Table 6: Median Effects F-score of CLINDE and DD-lasso by Decomposition on the n = 500 Networks. CL-init is CLINDE on the initial GRN with score threshold 3. DD is DD-lasso alone. CL-sub is CLINDE on the subnetworks with score threshold 3. DD-sub is DD-lasso on the subnetworks. The medians are taken over the 20 replicates.

slightly worse than CL-init. When m=200, the performance of DD-sub becomes better than CL-init and CL-sub, although DD-lasso performs poorly when applied to the large network directly (Tables 4 and 5).

Comparing *Component*, *Parents* and *XParents*, there is not much difference for both CL-sub and DD-sub when m is small. When m is larger, 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;

						Component		Parents		XParents	
n	*m*	*α*	*σ*	CL-init	DD	CL-sub	DD-sub	CL-sub	DD-sub	CL-sub	DD-sub
1000	20	0.5	0.5	0.019	0.022	0.011	0.010	0.011	0.010	0.011	0.010
			2	0.013	0.018	0.009	0.008	0.009	0.008	0.009	0.008
			8	0.013	0.017	0.008	0.008	0.008	0.008	0.008	0.008
		1	0.5	0.023	0.026	0.014	0.012	0.014	0.012	0.014	0.012
			2	0.016	0.020	0.010	0.010	0.010	0.011	0.010	0.011
			8	0.016	0.019	0.009	0.010	0.009	0.010	0.009	0.009
		2	0.5	0.045	0.045	0.027	0.020	0.027	0.020	0.027	0.021
			2	0.027	0.032	0.017	0.016	0.017	0.015	0.017	0.015
			8	0.029	0.032	0.018	0.016	0.018	0.016	0.018	0.016
		3	0.5	0.067	0.063	0.040	0.031	0.040	0.030	0.040	0.031
			2	0.047	0.051	0.028	0.022	0.028	0.022	0.028	0.022
			8	0.046	0.050	0.030	0.023	0.030	0.023	0.030	0.023
	50	0.5	0.5	0.201	0.104	0.180	0.042	0.196	0.042	0.195	0.045
			2	0.185	0.097	0.166	0.037	0.178	0.037	0.178	0.039
			8	0.182	0.098	0.158	0.037	0.176	0.038	0.176	0.040
		1	0.5	0.212	0.103	0.186	0.045	0.204	0.045	0.204	0.047
			2	0.193	0.101	0.170	0.037	0.186	0.040	0.185	0.042
			8	0.189	0.100	0.167	0.037	0.183	0.037	0.182	0.040
		2	0.5	0.225	0.113	0.201	0.049	0.219	0.050	0.219	0.055
			2	0.193	0.105	0.171	0.041	0.190	0.041	0.190	0.045
			8	0.199	0.107	0.176	0.045	0.194	0.044	0.194	0.047
		3	0.5	0.226	0.128	0.197	0.054	0.218	0.058	0.219	0.064
			2	0.205	0.124	0.179	0.049	0.197	0.050	0.199	0.055
			8	0.205	0.124	0.180	0.048	0.197	0.051	0.199	0.056
	100	0.5	0.5	0.499	0.122	0.429	0.402	0.501	0.297	0.499	0.326
			2	0.492	0.120	0.423	0.377	0.492	0.278	0.491	0.313
			8	0.489	0.121	0.419	0.377	0.488	0.233	0.487	0.261
		1	0.5	0.497	0.118	0.432	0.390	0.502	0.296	0.502	0.316
			2	0.492	0.120	0.424	0.389	0.497	0.278	0.496	0.302
			8	0.487	0.120	0.418	0.375	0.491	0.260	0.490	0.289
		2	0.5	0.484	0.112	0.410	0.391	0.485	0.254	0.483	0.298
			2	0.456	0.112	0.384	0.358	0.455	0.143	0.457	0.179
			8	0.468	0.112	0.394	0.368	0.467	0.195	0.468	0.233
		3	0.5	0.420	0.105	0.346	0.340	0.396	0.074	0.411	0.104
			2	0.407	0.104	0.329	0.313	0.381	0.075	0.400	0.102
			8	0.407	0.104	0.330	0.317	0.381	0.059	0.402	0.085
	200	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

Table 7: Median Effects F-score of CLINDE and DD-lasso by Decomposition on the n = 1000 Networks. CL-init is CLINDE on the initial GRN with score threshold 3. DD is DD-lasso alone. CL-sub is CLINDE on the subnetworks with score threshold 3. DD-sub is DD-lasso on the subnetworks. The medians are taken over the 20 replicates.

presumably it is because DD-lasso can already effectively remove indirect effects when given sufficient time points.

These results show that the strategy of using CLINDE to infer initial GRN and then decompose into subnetworks, than using DD-lasso (and sometimes CLINDE) to infer the subnetworks can be better than using CLINDE or DD-lasso alone on large network.

Real data pre-processing

Human transcription factors regulatory network: It is rare to find large known gene regulatory networks. Fortunately [35] has compiled the regulatory networks of 475 sequence-specific human

				Component		Parents		XParents	
n	*m*	*α*	*σ*	CL-sub	DD-sub	CL-sub	DD-sub	CL-sub	DD-sub
500	100	0.5	0.5	–	–	2.658E-02	–	–	–
			2	–	–	8.450E-04	–	1.812E-02	–
			8	–	–	9.537E-07	–	1.907E-06	–
		1	0.5	–	–	6.808E-03	–	5.270E-02	–
			2	–	–	6.046E-04	–	1.827E-03	–
			8	–	–	1.812E-05	–	–	–
		2	0.5	–	–	–	–	–	–
			2	–	–	7.682E-02	–	–	–
			8	–	–	–	–	–	–
		3	0.5	–	–	–	–	–	–
			2	–	–	–	–	–	–
			8	–	–	–	–	–	–
	200	0.5	0.5	–	4.154E-03	–	9.537E-07	–	9.537E-07
			2	5.344E-03	5.083E-04	–	9.537E-07	–	9.537E-07
			8	4.253E-04	7.656E-03	–	9.537E-07	3.186E-02	9.537E-07
		1	0.5	7.682E-02	3.648E-03	–	9.537E-07	3.793E-02	9.537E-07
			2	–	7.145E-02	–	9.537E-07	8.591E-03	9.537E-07
			8	5.699E-02	6.676E-05		9.537E-07	5.083E-04	9.537E-07
		2	0.5	–	3.147E-05	–	9.537E-07	–	9.537E-07
			2	1.638E-02	9.537E-07	–	9.537E-07	–	9.537E-07
			8	–	8.392E-05		9.537E-07		9.537E-07
		3	0.5	–	–	–	9.537E-07	–	9.537E-07
			2	–	–	–	9.537E-06	–	6.676E-06
			8	–	–	–	9.537E-07	–	9.537E-07
1000	100	0.5	0.5	–	–	7.162E-04	–	8.843E-02	–
			2	–	–	3.538E-04	–	3.195E-03	–
			8	–	–	8.450E-04	–	1.812E-02	–
		1	0.5	–	–	2.430E-03	–	1.827E-03	–
			2	–	–	4.101E-05	–	6.040E-03	–
			8	–	–	6.676E-06	–	1.049E-04	–
		2	0.5	–	–	–	–	8.248E-02	–
			2	–	–	–	–	–	–
			8	–	–	–	–	–	–
		3	0.5	–	–	–	–	–	–
			2	–	–	–	–	–	–
			8	–	–	–	–	–	–
	200	0.5	0.5	1.335E-05	9.537E-07	–	9.537E-07	1.163E-03	9.537E-07
			2	3.147E-05	9.537E-07	–	9.537E-07	4.718E-03	9.537E-07
			8	6.676E-06	1.907E-06	–	9.537E-07	8.591E-03	9.537E-07
		1	0.5	3.195E-03	1.335E-05	–	9.537E-07	2.110E-03	9.537E-07
			2	6.046E-04	1.612E-04	–	9.537E-07	1.907E-06	9.537E-07
			8	1.999E-02	1.307E-04	–	9.537E-07	1.074E-02	9.537E-07
		2	0.5	3.147E-05	9.537E-07	–	9.537E-07	–	9.537E-07
			2	–	1.612E-04	–	9.537E-07	–	9.537E-07
			8	1.163E-03	9.537E-07	–	9.537E-07	3.479E-02	9.537E-07
		3	0.5	–	–	–	4.768E-06	–	9.537E-07
			2	–	–	–	9.617E-03	–	3.147E-05
			8	–	–	–	5.245E-05	–	1.907E-06

Table 8: P-values of one-sided Wilcoxon signed-rank test on whether subnetwork F-score is better than initial GRN F-score on the n = 500 and n = 1000 Networks. CL-sub is CLINDE on the subnetworks with score threshold 3. DD-sub is DD-lasso on the subnetworks. The tests are based on the 20 replicates. The entries larger than 0.1 are omitted.

transcription factors across 41 diverse cell and tissue types, using genome-wide maps of *in vivo* DNaseI footprints. These networks should serve a useful role in evaluating GRN learning algorithms. But we cannot use the whole GRN as some constituent genes lack expression data used below. Table 9 shows the sizes of the TF GRNs before and after retaining only the genes with expression data used below. Also, since the TF GRNs give only regulatory links with direction, but not the regulatory effect (excitatory or repressive) nor the time delays, in the following we evaluate only the links performance.

TF GRN	Original		Retained	
	Genes	Links	Genes	Links
AG10803	521	12,482	458	10,878
AoAF	529	14,795	463	12,830
CD20+	515	16,723	456	14,695
CD34+ Mobilized	526	16,461	463	14,349
fBrain	519	11,698	458	10,274
fHeart	516	14,295	456	12,571
fLung	532	17,823	466	15,622
GM06990	501	12,994	445	11,567
GM12865	513	15,202	453	13,400
H7-hESC	533	16,424	468	13,815
HA-h	531	16,391	465	14,160
HAEpiC	526	13,286	461	11,804
HCF	522	14,492	461	12,713
HCM	527	15,115	462	13,265
HCPEpiC	527	13,903	464	12,001
HEEpiC	528	14,577	464	12,820
HepG2	493	12,863	438	11,277
HFF	513	12,126	455	10,619
HIPEpiC	527	12,511	462	11,048
HMF	526	13,961	463	12,156
HMVEC LLy	520	15,435	461	13,505
HMVEC-dBl-Ad	520	13,510	461	11,976
HMVEC-dBl-Neo	526	16,761	464	14,637
HMVEC-dLy-Neo	526	15,582	465	13,709
HPAF	531	13,501	468	11,914
HPdLF	521	12,822	460	11,064
HPF	527	14,588	465	12,818
HRCEpiC	525	9,597	462	8,516
HSMM	523	13,806	460	11,901
HVMF	526	15,135	461	13,226
IMR90	519	11,274	459	9,997
K562	493	9,099	437	7,976
NB4	525	18,906	462	16,258
NH-A	516	9,296	454	8,149
NHDF-Ad	529	13,644	465	11,870
NHDF-neo	521	15,565	460	13,582
NHLF	527	14,700	464	12,902
SAEC	522	9,886	458	8,831
SK-N-SH RA	508	12,761	447	11,303
SKMC	529	17,320	467	14,948
Th1	518	12,812	459	11,318

Table 9: Sizes of original TF GRNs and sizes of TF GRNs retained after matching to expression data.

Expression data: Even with high throughput technologies, it is not common to find long time series with small equidistant time steps. We have found GDS4238, which is the time series expression data of human bronchial epithelial cell response to inuenza virus, viral RNA and interferon-beta. Unfortunately these experiments are not performed specifically for the cell/tissue types of the GRNs in Tables 6-9. This time series is relatively long, with samples at 0.25, 0.5, 1, 1.5, 2, 4, 6, 8, 12 and 18 hours after treatment, and has multiple treatments with up to 2 replicates. We have filtered out some treatments with less time points, resulting in 14 (heterogeneous) series.

The pre-processing steps are as follows:

1. For each TF GRN above, take subset corresponding to the genes.

2. For each series, the given expression data is labelled as *raw*, and we take log10 of the expression data and label it as *log10*.

3. Since these time series have non-equidistant time steps, we perform spline interpolation to both *raw* and *log10* to obtain series with time step of 0.25 hour.

4. We take the prefix of each series up to 8 hours because the original time steps after 8 hours are much larger, and the interpolated data may not provide very useful information. This step gives the final pre-processed time series used. Since now the time series data are multiple heterogeneous replicates instead of one long time series, we rely on the feature of CLINDE and DD-lasso that can handle this type of expression data.

TF GRN	st	CL-init			DD			DD-Component			DD-Parents			DD-XParents		
		R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
AG10803	2	0.015	0.059	0.023	0.200	0.048	0.078	0.004	0.109	0.008	0.390	0.046	0.082	0.390	0.046	0.082
	2.3	0.014	0.061	0.023				0.005	0.113	0.009	0.268	0.049	0.083	0.268	0.049	0.083
	3	0.013	0.062	0.022				0.005	0.110	0.010	0.329	0.047	0.083	0.329	0.047	0.083
	3.3	0.013	0.063	0.021				0.006	0.119	0.011	0.282	0.053	0.089	0.282	0.053	0.089
	4	0.012	0.065	0.020				0.006	0.111	0.011	0.230	0.049	0.081	0.230	0.049	0.081
	4.3	0.012	0.065	0.020				0.005	0.108	0.009	0.266	0.056	0.092	0.266	0.056	0.092
	5	0.011	0.065	0.018				0.006	0.117	0.012	0.230	0.051	0.084	0.230	0.051	0.084
AoAF	2	0.016	0.072	0.026	0.196	0.055	0.086	0.004	0.102	0.007	0.333	0.050	0.087	0.333	0.050	0.087
	2.3	0.015	0.073	0.025				0.005	0.121	0.010	0.518	0.056	0.100	0.518	0.056	0.100
	3	0.014	0.076	0.023				0.005	0.123	0.010	0.370	0.060	0.103	0.370	0.060	0.103
	3.3	0.013	0.076	0.023				0.004	0.112	0.008	0.298	0.059	0.099	0.298	0.059	0.099
	4	0.012	0.078	0.021				0.005	0.123	0.010	0.222	0.057	0.091	0.222	0.057	0.091
	4.3	0.012	0.079	0.021				0.005	0.117	0.009	0.199	0.070	0.103	0.199	0.070	0.103
	5	0.012	0.081	0.020				0.006	0.130	0.012	0.350	0.063	0.107	0.350	0.063	0.107
CD20+	2	0.014	0.078	0.024	0.207	0.068	0.102	0.004	0.138	0.008	0.418	0.064	0.111	0.418	0.064	0.111
	2.3	0.014	0.079	0.023				0.005	0.152	0.009	0.379	0.073	0.123	0.379	0.073	0.123
	3	0.013	0.084	0.023				0.005	0.146	0.010	0.344	0.065	0.109	0.344	0.065	0.109
	3.3	0.013	0.086	0.022				0.005	0.151	0.010	0.241	0.071	0.110	0.241	0.071	0.110
	4	0.012	0.088	0.021				0.006	0.152	0.011	0.282	0.072	0.115	0.282	0.072	0.115
	4.3	0.012	0.087	0.020				0.006	0.141	0.011	0.296	0.070	0.113	0.296	0.070	0.113
	5	0.011	0.088	0.019				0.006	0.155	0.012	0.234	0.074	0.113	0.234	0.074	0.113
CD34+ Mobilized	2	0.014	0.074	0.024	0.200	0.063	0.096	0.004	0.133	0.009	0.454	0.060	0.106	0.454	0.060	0.106
	2.3	0.014	0.078	0.024				0.004	0.134	0.008	0.266	0.065	0.105	0.266	0.065	0.105
	3	0.013	0.082	0.023				0.005	0.131	0.009	0.263	0.071	0.112	0.263	0.071	0.112
	3.3	0.013	0.084	0.022				0.005	0.121	0.009	0.327	0.073	0.120	0.327	0.073	0.120
	4	0.012	0.086	0.021				0.005	0.134	0.009	0.257	0.070	0.111	0.257	0.070	0.111
	4.3	0.012	0.088	0.021				0.005	0.138	0.009	0.154	0.064	0.091	0.154	0.064	0.091
	5	0.011	0.089	0.020				0.006	0.136	0.011	0.165	0.069	0.098	0.165	0.069	0.098

Table 10: Links Peformances of *raw* for first 4 Real Data for CLINDE for initial GRN, DD-lasso for whole network and for subnetworks. CL-init is CLINDE on the initial network with score threshold st. DD is DD-lasso alone on the network, so is the same for different st for a given TF GRN. DD-component, DD-parents and DD-XParents are DD-lasso on the subnetworks with different decompositions. R is Recall, P is Precision, F is F-score.

Real data results

The results of synthetic data suggest that using DD-lasso for inferring subnetworks gives better results with moderate amount of data, so here we focus on DD-lasso for subnetworks. The *links* performances of CLINDE on the initial GRN, DD-lasso alone on the network, and using DD-lasso on the subnetworks (with *Component*, *Parents* and *XParents*) are shown in Tables 10-13. In Tables 10 and 11 we show the performances of the first 4 TF GRNs with different score thresholds for *raw* and *log10* respectively, and in Tables 12 and 13 we show the performances on all TF GRNs with score threshold 3 for *raw* and *log10* respectively, because the performances are quite similar with different score thresholds. Tables 3-6 show more detailed performances for the real data. Firstly, the F-scores are quite poor, which may be due to noise in the data, or that the data are not specifically for the genes in the TF GRNs. Secondly, comparing *raw* and *log10*, both CLINDE on the initial GRN, DD-lasso alone and DD-lasso on the subnetworks perform slightly better using *raw*. Thirdly, even DD-lasso alone performs better than CLINDE on the initial GRN, with much better *recall*, but slightly worse *precision*. DD-lasso on the subnetworks with *Parents* or *XParents* is better than DD-lasso alone, though DD-lasso with *Component* performs worse than CLINDE. The *Precision* of DD-lasso with *Component* is better than DD-lasso alone and CLINDE alone, but the *Recall* is much worse, leading to poorer *F-score*. Decomposing into subnetworks by *Parents* and *XParents* improves mainly the *Recall* and a little bit of the *Precision* for DD-lasso. This suggests that the decomposition is not done very well because the initial GRN is not very good, but including the parents of the components allows DD-lasso to recover more links, which alleviates the problem of bad decomposition. The performance of DD-lasso on subnetworks varies with the score threshold of CLINDE on the initial GRN, but the trend is not very clear, though all are better than DD-lasso alone. We have also performed one-sided Wilcoxon signed rank test comparing the median F-scores of CLINDE on initial GRN, DD-lasso on initial GRN, and DD-lasso with decomposition using *Parents* or *XParents*, with different score thresholds, using the 41 TF GRNs as the cases. The p-values are shown in Table 14. We can see that except for *raw* with score threshold of 2, in all other cases, the p-values are very small, showing that the improvements of decomposing into subnetworks are statistically significant. The results on real data show that there is good potential of our proposed strategy of decomposing an initial GRN into subnetworks and then inferring the subnetworks with a possibly different algorithm, even when the initial GRN is not estimated very accurately, and thus the decomposition is not accurate either.

Discussions and Conclusion

Ideally, there would be prior knowledge on the decomposition of a large network into subnetworks so that GRN learning can work on smaller networks using the limited expression data. Lacking this prior knowledge, expression data has to be utilized in the decomposition.

TF GRN	st	CL-init			DD			DD-Component			DD-Parents			DD-XParents		
		R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
AG10803	2	0.015	0.061	0.024	0.199	0.048	0.078	0.005	0.125	0.009	0.266	0.046	0.078	0.266	0.046	0.078
	2.3	0.014	0.059	0.022				0.005	0.112	0.009	0.389	0.050	0.089	0.389	0.050	0.089
	3	0.013	0.061	0.021				0.005	0.112	0.010	0.314	0.051	0.088	0.314	0.051	0.088
	3.3	0.012	0.059	0.020				0.006	0.114	0.011	0.390	0.051	0.091	0.390	0.051	0.091
	4	0.012	0.063	0.020				0.005	0.113	0.010	0.199	0.055	0.087	0.199	0.055	0.087
	4.3	0.012	0.067	0.020				0.005	0.095	0.010	0.296	0.053	0.090	0.296	0.053	0.090
	5	0.011	0.065	0.018				0.005	0.107	0.010	0.160	0.050	0.077	0.160	0.050	0.077
AoAF	2	0.015	0.070	0.025	0.192	0.055	0.085	0.004	0.122	0.007	0.378	0.053	0.093	0.378	0.053	0.093
	2.3	0.014	0.070	0.023				0.004	0.107	0.008	0.339	0.058	0.099	0.339	0.058	0.099
	3	0.013	0.071	0.022				0.004	0.116	0.008	0.351	0.058	0.100	0.351	0.058	0.100
	3.3	0.012	0.072	0.021				0.005	0.118	0.009	0.310	0.056	0.095	0.310	0.056	0.095
	4	0.012	0.077	0.021				0.005	0.121	0.010	0.167	0.065	0.094	0.167	0.065	0.094
	4.3	0.012	0.082	0.022				0.005	0.116	0.010	0.256	0.057	0.094	0.256	0.057	0.094
	5	0.011	0.077	0.019				0.005	0.125	0.010	0.243	0.056	0.091	0.243	0.056	0.091
CD20+	2	0.015	0.080	0.025	0.201	0.066	0.100	0.005	0.143	0.009	0.308	0.073	0.117	0.308	0.073	0.117
	2.3	0.014	0.079	0.023				0.005	0.157	0.009	0.426	0.069	0.118	0.426	0.069	0.118
	3	0.012	0.078	0.021				0.005	0.160	0.010	0.292	0.073	0.117	0.292	0.073	0.117
	3.3	0.012	0.080	0.021				0.005	0.150	0.010	0.213	0.074	0.110	0.213	0.074	0.110
	4	0.012	0.086	0.021				0.005	0.144	0.010	0.297	0.076	0.121	0.297	0.076	0.121
	4.3	0.012	0.090	0.021				0.006	0.146	0.011	0.297	0.072	0.116	0.297	0.072	0.116
	5	0.011	0.087	0.019				0.006	0.163	0.011	0.166	0.086	0.114	0.166	0.086	0.114
CD34+ Mobilized	2	0.015	0.076	0.025	0.199	0.063	0.095	0.004	0.136	0.008	0.267	0.068	0.108	0.267	0.068	0.108
	2.3	0.014	0.076	0.023				0.005	0.136	0.009	0.424	0.064	0.111	0.424	0.064	0.111
	3	0.013	0.078	0.022				0.005	0.140	0.009	0.290	0.065	0.106	0.290	0.065	0.106
	3.3	0.012	0.079	0.021				0.005	0.143	0.010	0.241	0.064	0.101	0.241	0.064	0.101
	4	0.012	0.086	0.021				0.005	0.136	0.010	0.193	0.070	0.102	0.193	0.070	0.102
	4.3	0.012	0.091	0.022				0.005	0.140	0.010	0.267	0.071	0.112	0.267	0.071	0.112
	5	0.011	0.086	0.019				0.005	0.138	0.010	0.170	0.063	0.091	0.170	0.063	0.091

Table 11: Links Performances of *log10* for first 4 Real Data for CLINDE for initial GRN, DD-lasso for whole network and for subnetworks. CL-init is CLINDE on the initial network with score threshold st. DD is DD-lasso alone on the network, so is the same for different st for a given TF GRN. DD-component, DD-parents and DD-XParents are DD-lasso on the subnetworks with different decompositions. R is Recall, P is Precision, F is F-score.

In this paper, we use CLINDE to infer the initial GRN because it can handle delays and indirect effects, and perform better than DD-lasso on small number of time points relative to the network size. It is natural to ask whether using other algorithm for the initial GRN would give better results. This should be investigated in future work. The goal of inferring the initial GRN is to decompose the large network into subnetworks. In this paper, we have used the measure of "edge betweenness" used in community structure discovery algorithms, and the simple strategy of successively removing the edge with the highest edge betweenness" until the components are below a certain size. Different algorithms used in community structure discovery could be investigated and compared with this simple strategy. Furthermore, clustering based algorithms should be considered to approximate the subnetworks, though most probably customized algorithm may need to be developed to handle delays and indirect effects, as mentioned before.

The synthetic data results show that while DD-lasso alone performs poorly on large network, the performance dramatically improves after decomposing the initial GRN into subnetworks. The real data results show that even the initial GRN is not very good; DD-lasso on the sub- networks nevertheless performs better than DD-lasso alone. The results raise the possibility of combining CLINDE and DD-lasso more tightly, for example by using CLINDE to estimate the parents and the associated delays for each gene, then using lasso as in DD-lasso to further remove indirect effects, without explicitly estimating the (possibly overlapping) subnetworks.

To conclude, we have proposed an algorithm to first infer an initial GRN using CLINDE, then decompose it into possibly overlapping subnetworks, then infer each subnetwork by either CLINDE or DD-lasso and finally merge the subnetworks, to try to alleviate the problem of insufficient expression data relative to network size. We have tested this algorithm on synthetic data of networks with 500 and 1000 genes. We have also tested on real data on 41 human TF regulatory networks. Results show that our proposed algorithm does improve the GRN learning performance of using either CLINDE or DD-lasso alone on the large networks.

Acknowledgments

This research is supported by General Research Funds LU310111 and 414413 from the Research Grant Council of the Hong Kong Special Administrative Region.

TF GRN	CL-init			DD			DD-Component			DD-Parents			DD-XParents		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
AG10803	0.013	0.062	0.022	0.200	0.048	0.078	0.005	0.110	0.010	0.329	0.047	0.083	0.329	0.047	0.083
AoAF	0.014	0.076	0.023	0.196	0.055	0.086	0.005	0.123	0.010	0.370	0.060	0.103	0.370	0.060	0.103
CD20+	0.013	0.084	0.023	0.207	0.068	0.102	0.005	0.146	0.010	0.344	0.065	0.109	0.344	0.065	0.109
CD34+ Mobilized	0.013	0.082	0.023	0.200	0.063	0.096	0.005	0.131	0.009	0.263	0.071	0.112	0.263	0.071	0.112
fBrain	0.013	0.058	0.021	0.199	0.045	0.074	0.005	0.113	0.010	0.236	0.043	0.073	0.236	0.043	0.073
fHeart	0.013	0.071	0.022	0.202	0.057	0.089	0.006	0.136	0.011	0.308	0.059	0.099	0.308	0.059	0.099
fLung	0.013	0.084	0.022	0.199	0.068	0.101	0.005	0.138	0.009	0.314	0.069	0.113	0.314	0.069	0.113
GM06990	0.013	0.069	0.022	0.215	0.057	0.090	0.006	0.134	0.011	0.269	0.053	0.089	0.269	0.053	0.089
GM12865	0.014	0.082	0.024	0.207	0.063	0.096	0.006	0.139	0.011	0.388	0.063	0.108	0.388	0.063	0.108
H7-hESC	0.012	0.071	0.021	0.196	0.059	0.091	0.004	0.120	0.008	0.416	0.061	0.107	0.416	0.061	0.107
HA-h	0.012	0.072	0.020	0.200	0.062	0.095	0.005	0.138	0.010	0.308	0.068	0.112	0.308	0.068	0.112
HAEpiC	0.013	0.065	0.021	0.206	0.054	0.085	0.005	0.122	0.010	0.335	0.055	0.095	0.335	0.055	0.095
HCF	0.014	0.077	0.024	0.200	0.056	0.088	0.006	0.129	0.011	0.345	0.054	0.093	0.345	0.054	0.093
HCM	0.014	0.081	0.024	0.202	0.059	0.091	0.005	0.146	0.010	0.221	0.060	0.094	0.221	0.060	0.094
HCPEpiC	0.012	0.061	0.020	0.195	0.052	0.082	0.005	0.121	0.009	0.239	0.054	0.088	0.239	0.054	0.088
HEEpiC	0.012	0.068	0.021	0.200	0.056	0.088	0.005	0.116	0.009	0.296	0.056	0.094	0.296	0.056	0.094
HepG2	0.013	0.069	0.022	0.222	0.058	0.092	0.006	0.144	0.012	0.272	0.062	0.101	0.272	0.062	0.101
HFF	0.013	0.063	0.022	0.201	0.048	0.077	0.006	0.119	0.011	0.339	0.055	0.095	0.339	0.055	0.095
HIPEpiC	0.013	0.063	0.022	0.200	0.048	0.078	0.005	0.113	0.010	0.313	0.045	0.079	0.313	0.045	0.079
HMF	0.012	0.064	0.020	0.195	0.052	0.082	0.005	0.130	0.010	0.275	0.055	0.092	0.275	0.055	0.092
HMVEC LLy	0.014	0.080	0.023	0.203	0.061	0.093	0.005	0.135	0.010	0.393	0.063	0.108	0.393	0.063	0.108
HMVEC-dBl-Ad	0.014	0.072	0.023	0.200	0.053	0.084	0.005	0.118	0.009	0.308	0.049	0.085	0.308	0.049	0.085
HMVEC-dBl-Neo	0.014	0.085	0.023	0.200	0.064	0.097	0.004	0.132	0.008	0.313	0.067	0.111	0.313	0.067	0.111
HMVEC-dLy-Neo	0.013	0.079	0.023	0.202	0.061	0.094	0.005	0.138	0.010	0.151	0.051	0.077	0.151	0.051	0.077
HPAF	0.013	0.065	0.021	0.196	0.051	0.081	0.005	0.103	0.009	0.379	0.055	0.096	0.379	0.055	0.096
HPdLF	0.013	0.063	0.022	0.200	0.049	0.078	0.006	0.124	0.012	0.390	0.057	0.099	0.390	0.057	0.099
HPF	0.013	0.071	0.022	0.200	0.056	0.088	0.005	0.126	0.009	0.267	0.057	0.094	0.267	0.057	0.094
HRCEpiC	0.014	0.053	0.023	0.201	0.038	0.064	0.005	0.078	0.009	0.418	0.043	0.078	0.418	0.043	0.078
HSMM	0.012	0.063	0.020	0.202	0.053	0.084	0.006	0.129	0.011	0.251	0.049	0.082	0.251	0.049	0.082
HVMF	0.012	0.072	0.021	0.200	0.058	0.090	0.005	0.140	0.010	0.355	0.066	0.111	0.355	0.066	0.111
IMR90	0.014	0.059	0.022	0.203	0.045	0.073	0.005	0.094	0.009	0.304	0.039	0.069	0.304	0.039	0.069
K562	0.014	0.052	0.022	0.218	0.040	0.068	0.006	0.096	0.011	0.355	0.043	0.077	0.355	0.043	0.077
NB4	0.013	0.092	0.023	0.200	0.071	0.105	0.005	0.143	0.009	0.413	0.073	0.124	0.413	0.073	0.124
NH-A	0.013	0.047	0.021	0.203	0.037	0.063	0.005	0.082	0.009	0.228	0.034	0.059	0.228	0.034	0.059
NHDF-Ad	0.013	0.064	0.021	0.193	0.050	0.080	0.005	0.126	0.010	0.261	0.062	0.100	0.261	0.062	0.100
NHDF-neo	0.013	0.075	0.022	0.200	0.060	0.092	0.005	0.140	0.010	0.367	0.061	0.104	0.367	0.061	0.104
NHLF	0.013	0.071	0.022	0.201	0.057	0.089	0.004	0.130	0.009	0.224	0.060	0.095	0.224	0.060	0.095
SAEC	0.014	0.053	0.022	0.202	0.040	0.067	0.004	0.081	0.008	0.280	0.042	0.073	0.280	0.042	0.073
SK-N-SH RA	0.013	0.064	0.021	0.207	0.053	0.085	0.005	0.119	0.009	0.208	0.047	0.077	0.208	0.047	0.077
SKMC	0.013	0.081	0.022	0.195	0.064	0.096	0.004	0.130	0.008	0.236	0.065	0.102	0.236	0.065	0.102
Th1	0.014	0.069	0.023	0.200	0.050	0.080	0.005	0.108	0.010	0.446	0.055	0.098	0.446	0.055	0.098

Table 12: Links Performances of *raw* Real Data for CLINDE for initial GRN, DD- lasso for whole network and for subnetworks. The score threshold is 3. CL-init is CLINDE on the initial network. DD is DD-lasso alone on the network. DD-component, DD-parents and DD-XParents are DD-lasso on the subnetworks with different decompositions. R is Recall, P is Precision, F is F-score.

TF GRN	CL-init			DD			DD-Component			DD-Parents			DD-XParents		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
AG10803	0.013	0.061	0.021	0.199	0.048	0.078	0.005	0.112	0.010	0.314	0.051	0.088	0.314	0.051	0.088
AoAF	0.013	0.071	0.022	0.192	0.055	0.085	0.004	0.116	0.008	0.351	0.058	0.100	0.351	0.058	0.100
CD20+	0.012	0.078	0.021	0.201	0.066	0.100	0.005	0.160	0.010	0.292	0.073	0.117	0.292	0.073	0.117
CD34+ Mobilized	0.013	0.078	0.022	0.199	0.063	0.095	0.005	0.140	0.009	0.290	0.065	0.106	0.290	0.065	0.106
fBrain	0.014	0.061	0.022	0.205	0.047	0.076	0.005	0.113	0.010	0.192	0.054	0.084	0.192	0.054	0.084
fHeart	0.013	0.071	0.022	0.206	0.058	0.090	0.005	0.130	0.009	0.285	0.064	0.105	0.285	0.064	0.105
fLung	0.012	0.079	0.021	0.197	0.067	0.100	0.005	0.149	0.010	0.372	0.074	0.123	0.372	0.074	0.123
GM06990	0.013	0.067	0.021	0.213	0.056	0.089	0.006	0.143	0.011	0.251	0.049	0.081	0.251	0.049	0.081
GM12865	0.013	0.075	0.022	0.208	0.062	0.096	0.005	0.145	0.010	0.403	0.064	0.110	0.403	0.064	0.110
H7-hESC	0.012	0.071	0.020	0.198	0.059	0.091	0.005	0.131	0.009	0.335	0.062	0.105	0.335	0.062	0.105
HA-h	0.012	0.071	0.020	0.198	0.061	0.093	0.004	0.134	0.009	0.263	0.065	0.104	0.263	0.065	0.104
HAEpiC	0.012	0.060	0.020	0.200	0.052	0.083	0.005	0.121	0.010	0.346	0.054	0.094	0.346	0.054	0.094
HCF	0.013	0.072	0.022	0.202	0.056	0.088	0.005	0.140	0.010	0.166	0.053	0.080	0.166	0.053	0.080
HCM	0.013	0.075	0.022	0.200	0.058	0.090	0.005	0.141	0.010	0.341	0.061	0.103	0.341	0.061	0.103
HCPEpiC	0.012	0.061	0.020	0.194	0.051	0.081	0.005	0.122	0.009	0.294	0.051	0.087	0.294	0.051	0.087
HEEpiC	0.012	0.068	0.021	0.195	0.055	0.086	0.006	0.133	0.011	0.337	0.058	0.098	0.337	0.058	0.098
HepG2	0.013	0.065	0.021	0.217	0.056	0.090	0.006	0.143	0.011	0.179	0.055	0.084	0.179	0.055	0.084
HFF	0.012	0.058	0.020	0.206	0.049	0.079	0.006	0.137	0.012	0.291	0.052	0.088	0.291	0.052	0.088
HIPEpiC	0.013	0.063	0.022	0.203	0.049	0.079	0.005	0.122	0.010	0.345	0.053	0.092	0.345	0.053	0.092
HMF	0.012	0.061	0.019	0.201	0.054	0.085	0.006	0.129	0.011	0.276	0.058	0.096	0.276	0.058	0.096
HMVEC LLy	0.012	0.072	0.021	0.199	0.060	0.092	0.005	0.145	0.010	0.278	0.064	0.104	0.278	0.064	0.104
HMVEC-dBl-Ad	0.012	0.064	0.021	0.201	0.053	0.084	0.005	0.126	0.010	0.349	0.060	0.103	0.349	0.060	0.103
HMVEC-dBl-Neo	0.013	0.080	0.022	0.197	0.064	0.096	0.005	0.145	0.010	0.361	0.072	0.120	0.361	0.072	0.120
HMVEC-dLy-Neo	0.012	0.072	0.021	0.197	0.059	0.091	0.005	0.144	0.010	0.384	0.065	0.110	0.384	0.065	0.110
HPAF	0.012	0.061	0.020	0.198	0.051	0.081	0.005	0.109	0.009	0.338	0.056	0.096	0.338	0.056	0.096
HPdLF	0.013	0.062	0.021	0.196	0.048	0.077	0.005	0.122	0.010	0.189	0.045	0.073	0.189	0.045	0.073
HPF	0.012	0.067	0.020	0.200	0.056	0.087	0.005	0.137	0.009	0.247	0.059	0.096	0.247	0.059	0.096
HRCEpiC	0.015	0.054	0.023	0.197	0.037	0.062	0.005	0.083	0.009	0.264	0.036	0.064	0.264	0.036	0.064
HSMM	0.012	0.061	0.020	0.202	0.053	0.084	0.005	0.125	0.010	0.318	0.056	0.095	0.318	0.056	0.095
HVMF	0.011	0.065	0.019	0.202	0.059	0.091	0.006	0.150	0.011	0.251	0.056	0.092	0.251	0.056	0.092
IMR90	0.014	0.059	0.022	0.201	0.044	0.073	0.005	0.108	0.010	0.263	0.043	0.074	0.263	0.043	0.074
K562	0.013	0.047	0.020	0.215	0.040	0.067	0.006	0.102	0.011	0.346	0.044	0.077	0.346	0.044	0.077
NB4	0.012	0.085	0.021	0.205	0.073	0.108	0.005	0.156	0.009	0.241	0.083	0.124	0.241	0.083	0.124
NH-A	0.012	0.044	0.019	0.202	0.037	0.062	0.005	0.075	0.009	0.352	0.038	0.069	0.352	0.038	0.069
NHDF-Ad	0.012	0.059	0.019	0.196	0.051	0.081	0.005	0.116	0.009	0.316	0.048	0.083	0.316	0.048	0.083
NHDF-neo	0.012	0.070	0.020	0.199	0.060	0.092	0.005	0.145	0.010	0.270	0.057	0.094	0.270	0.057	0.094
NHLF	0.013	0.072	0.022	0.199	0.056	0.088	0.005	0.127	0.009	0.365	0.058	0.100	0.365	0.058	0.100
SAEC	0.013	0.051	0.021	0.201	0.040	0.066	0.006	0.108	0.011	0.176	0.038	0.063	0.176	0.038	0.063
SK-N-SH RA	0.013	0.066	0.022	0.207	0.053	0.085	0.005	0.108	0.010	0.435	0.055	0.098	0.435	0.055	0.098
SKMC	0.012	0.075	0.020	0.197	0.064	0.097	0.004	0.134	0.008	0.258	0.075	0.116	0.258	0.075	0.116
Th1	0.013	0.062	0.021	0.200	0.051	0.081	0.006	0.120	0.011	0.416	0.052	0.093	0.416	0.052	0.093

Table 13: Links Performances of *log10* Real Data for CLINDE for initial GRN, DD- lasso for whole network and for subnetworks. The score threshold is 3. CL-init is CLINDE on the initial network. DD is DD-lasso alone on the network. DD-component, DD-parents and DD-XParents are DD-lasso on the subnetworks with different decompositions. R is Recall, P is Precision, F is F-score.

Type	st	DD > CL-init	DD-Parents		DD-XParents	
			> CL-init	> DD	> CL-init	> DD
raw	2	4.547474e-13	1.240421e-08	0.1104267	1.240421e-08	0.1104267
	2.3	4.547474e-13	1.252151e-08	5.115953e-08	1.252151e-08	5.115953e-08
	3	4.547474e-13	1.256364e-08	1.984658e-07	1.256364e-08	1.984658e-07
	3.3	4.547474e-13	1.253835e-08	2.273737e-12	1.253835e-08	2.273737e-12
	4	4.547474e-13	1.246274e-08	1.500666e-11	1.246274e-08	1.500666e-11
	4.3	4.547474e-13	1.24879e-08	2.658122e-06	1.24879e-08	2.658122e-06
	5	4.547474e-13	1.253835e-08	1.396074e-10	1.253835e-08	1.396074e-10
log10	2	4.547474e-13	1.252993e-08	6.794688e-06	1.252993e-08	6.794688e-06
	2.3	4.547474e-13	1.252151e-08	5.734364e-10	1.252151e-08	5.734364e-10
	3	4.547474e-13	1.255099e-08	7.660674e-09	1.255099e-08	7.660674e-09
	3.3	4.547474e-13	1.24879e-08	1.136868e-11	1.24879e-08	1.136868e-11
	4	4.547474e-13	1.25047e-08	1.364242e-12	1.25047e-08	1.364242e-12
	4.3	4.547474e-13	1.252993e-08	6.366463e-12	1.252993e-08	6.366463e-12
	5	4.547474e-13	1.246693e-08	3.268769e-08	1.246693e-08	3.268769e-08

CL-init is CLINDE on the initial network. DD is DD-lasso alone on the network. DD-parents and DD-XParents are DD-lasso on the subnetworks with different decompositions. The tests are based on the 41 TF GRNs.

Table 14: P-values of one-sided Wilcoxon signed-rank test on the medians of the Real Data F-scores with different alternative hypotheses.

References

1. Pavlov MY, Ehrenberg M (1996) Rate of translation of natural mrnas in an optimized in vitro system. Arch Biochem Biophys 328: 9-16.
2. Romero L, Silber M, Hatzimanikatis V (2010) The Origins of Time-Delay in Template Biopolymerization Processes. PLoS Comput Bio l 6: 1000726.
3. Mcadams HH, Shapiro L (1995) Circuit simulation of genetic networks. Science 269: 650-656.
4. Swinburne IA, Silver PA (2008) Intron delays and transcriptional timing during development. Developmental cell 14: 324-330.
5. Chen L, Aihara K (2002) Stability of genetic regulatory networks with time delay. IEEE Transactions on 49: 602 -608.
6. Lewis J (2003) Autoinhibition with transcriptional delay: a simple mechanism for the zebra_sh somitogenesis oscillator. Curr Biol 13: 1398-1408.
7. Mather W, Bennett MR, Hasty J, Tsimring LS (2009) Delay-induced degrade-and-reos- cillations in small genetic circuits. Phys Rev Lett 102: 068105.
8. Tigges M, Marquez-Lago TT, Stelling J, Fussenegger M (2009) A tunable synthetic mam- malian oscillator. Nature 457: 309-312.
9. Karlebach G, Shamir R (2008) Modelling and analysis of gene regulatory networks. Nature Reviews Molecular Cell Biology 9: 770-780.
10. Schlitt T, Brazma A (2007) Current approaches to gene regulatory network modelling. BMC Bioinformatics 8: S9.
11. Bansal M, Belcastro V, Ambesi-Impiombato A, di Bernardo D (2007) How to infer gene networks from expression profiles. Molecular Systems Biology 3: 78.
12. Margolin A, Nemenman I, Basso K, Wiggins C, Stolovitzky G, et al. (2006) Aracne: An al- gorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. BMC Bioinformatics 7: S7.
13. Altay G, Emmert-Streib F (2010) Inferring the conservative causal core of gene regulatorynetworks. BMC Systems Biology 4: 132.
14. Ram R, Chetty M, Dix T (2006) Causal modeling of gene regulatory network. IEEE Symposium on. pp. 1 -8.
15. Kuo HC, Tsai PC, Huang JP (2009) Finding time-delayed gene regulation patterns from microarray data. Dimension 1: 117 -122.
16. Maucher M, Kracher B, Kuhl M, Kestler HA (2011) Inferring Boolean network structure via correlation. Bioinformatics 27: 1529-1536.
17. Aijo T, Lahdesmaki H (2009) Learning gene regulatory networks from gene expression measurements using non-parametric molecular kinetics. Bioinformatics 25: 2937-2944.
18. Li Z, Li P, Krishnan A, Liu J (2011) Large-scale dynamic gene regulatory network infer- ence combining differential equation models with local dynamic Bayesian network analysis. Bioinformatics 27: 2686-2691.
19. Tienda-Luna IM, Huang Y, Yin Y, Padillo DPR, Perez MCC (2007) Uncovering gene regulatory networks from time-series microarray data with variational bayesian structural expectation maximization. EURASIP J Bioinformatics and Systems Biology 2007.
20. Xing Z, Wu D (2006) Modeling multiple time units delayed gene regulatory network using dynamic bayesian network. IEEE 6: 190-195.
21. Yu J, Smith VA, Wang PP, Hartemink AJ, Jarvis ED (2004) Advances to bayesian network inference for generating causal networks from observational biological data. Bioinformatics 20: 3594-3603.
22. Zoppoli P, Morganella S, CeccarelliM (2010) Timedelay-aracne: Reverse engineering of gene networks from time-course data by an information theoretic approach. BMC Bioinformatics 11: 154.
23. ElBakry O, Ahmad M, Swamy M (2013) Inference of gene regulatory networks with variable time delay from time-series microarray data. IEEE/ACM Trans Comput Biol Bioinform 10: 671-687.
24. Tibshirani R (1994) Regression shrinkage and selection via the lasso. J Royal Stat Soc Ser 58: 267-288.
25. Lo LY, Leung KS, Lee KH Inferring time-delayed causal gene network using time-series expression data. IEEE/ACM Trans Comput Biol Bioinform 12: 1169-1182.
26. Gregoretti F, Belcastro V, di Bernardo D, Oliva G (2010) A parallel implementation of the network identi_cation by multiple regression (nir) algorithm to reverse-engineer regulatory gene networks. PLoS ONE 5: 10179.
27. Xing B, Van Der Laan MJ (2005) A causal inference approach for constructing transcrip- tional regulatory networks. Bioinformatics 21: 4007-4013.
28. Zhang X, Baral C, Kim S (2005) An Algorithm to Learn Causal Relations Between Genes from Steady State Data: Simulation and Its Application to Melanoma Dataset. Arti_cial Intelligence in Medicine 1: 524-534.
29. Pearl J (2000) Causality: Models, Reasoning, and Inference. Cambridge University Press.
30. Spirtes P, Glymour C, Scheines R (2001) Causation, Prediction, and Search.
31. Julius A, Zavlanos M, Boyd S, Pappas GJ (2009) Genetic network identification using convex programming. IET Syst Biol 3: 155-166.
32. Bansal M, Gatta GD, di Bernardo D (2006) Inference of gene regulatory networks and compound mode of action from time course gene expression profiles. Bioinformatics 22: 815-822.
33. Newman ME (2001) Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. Physical review E 64: 016132.
34. Girvan M, Newman ME (2002) Community structure in social and biological networks. Proceedings of the National Academy of Sciences 99: 7821-7826.
35. Neph S, Stergachis AB, Reynolds A, Sandstrom R, Borenstein E, et al. (2012) Circuitry and dynamics of human transcription factor regulatory networks. Cell 150: 1274-1286.