CN: A Consensus Algorithm for Inferring Gene Regulatory Networks Using SORDER Algorithm and Conditional Mutual Information Test

Rosa Aghdam[1], Mojtaba Ganjali[1,6], Xiujun Zhang [2,3,4,*] Changiz Eslahchi[5,6,*]

1 Faculty of Mathematical Sciences, Department of Statistics, Shahid Beheshti University, G.C., Tehran, Iran.

2 Department of Mathematics, Xinyang Normal University, Xinyang, 464000, China.

3 Key Laboratory of Systems Biology, Shanghai Institutes for Biological Sciences, Chinese Academy of Sciences, Shanghai, 200031, China.

4 Department of Microbiology, Immunology and Molecular Genetics, University of California Los Angeles, 90095, Los Angeles, USA.

5 Faculty of Mathematical Sciences, Department of Computer Science, Shahid Beheshti University, G.C., Tehran, Iran.

6 Institute for Research in Fundamental Sciences (IPM), Iran.

∗ E-mail: zhang-xiujun@163.com, eslahchi.ch@gmail.com.

Contents

1 Supplementary Materials

In this section, the definitions of Bayesian network and Conditional Mutual information are introduced. In addition more details of proposed algorithms are explained.

1.1 Preliminaries

1.1.1 Bayesian Network

Bayesian Networks (BNs), known also as belief networks, are the family of probabilistic graphical models [1]. A BN is a DAG which consists of a set of vertices and a set of directed edges between vertices. If there is a directed edge from A to B, we say that B is a child of A and A is a parent of B. Learning BNs from datasets consists of two stages. In the first stage the skeleton of the BN is determined. This is called structural learning. In the second stage the parameters of the BN are determined. This is called parameter learning. A BN, indicated by $M = (G, \theta_G)$, consists of a network structure, G, and a set of parameters, θ_G, where parameters determine the conditional probabilities of the model. The structure of BN consists of a DAG, $G = (\mathbf{X}, E(G))$, where $\mathbf{X}$ denotes the set of vertices and $E(G)$ indicates the set of edges which represents dependency relationships between vertices.
The vector of parameters of BN, θ_G, contains the parameter $\theta_{x_i|Pa_M(x_i)} = P_M(x_i|Pa_M(x_i))$, where x_i denotes some value of the X_i and $Pa_M(x_i)$ indicates some set of values for X_i's parents. If X_i has no parent, then $P(X_i|Pa_M(X_i))$ is equal to $P(X_i)$. By using these conditional distributions, the joint distribution over $\mathbf{X}$ can be obtained as follows:

$$P(X_1, X_2, ..., X_n) = \prod_{X_i \in X} P(X_i|Pa_M(X_i)).$$

If $P(X, Y|\mathbf{Z}) = P(X|\mathbf{Z})P(Y|\mathbf{Z})$, then two vertices X and Y are conditionally independent given $\mathbf{Z}$.
Basically, there are three methods for learning the structure of BNs from data; constraint-based methods [2–8], score-based searching methods [9–11] and Hybrid methods [12,13]. Constraint-based methods apply information about Conditional Independent (CI) test to determine dependency between vertices. The score-based searching methods produce a series of candidate networks, calculate a score for each candidate and return a candidate with the highest score. The hybrid method is a combination of this two methods. We refer to [1] for the basic notions on BNs.

1.1.2 Conditional Mutual information

Mutual Information (MI) is a suitable tool for detecting nonlinear dependencies between genes and has been widely used to infer GRNs [14]. Furthermore, MI is able to deal with thousands of genes and a limited number of samples [15]. Gene expression data are typically modeled as continuous variables. With the widely adopted hypothesis of Gaussian distribution for gene expression data. The measure of MI for two continuous variables X

and Y can be simply calculated using the following formula[8,16].

$$MI(X,Y) = \frac{1}{2} log \frac{\sigma_X^2 \sigma_Y^2}{\sigma_{XY}}, \tag{1}$$

where σ_X^2, σ_Y^2 and σ_{XY} indicate the variance of X, the variance of Y and the covariance between X and Y, respectively. Similarly, CMI for continuous variables X and Y given the row vector of vertices $\mathbf{Z}$ can be determined by formula[8]

$$CMI(X,Y|\mathbf{Z}) = \frac{1}{2} log \frac{det(C(X,\mathbf{Z}))det(C(Y,\mathbf{Z}))}{det(C(\mathbf{Z}))det(C(X,Y,Z))}, \tag{2}$$

where $C(A)$ and $det(C(A))$ represent the covariance matrix of vector $\mathbf{A}$ and determinant of $C(A)$, respectively. If X and Y are conditionally independent given $\mathbf{Z}$, then $CMI(X,Y|\mathbf{Z}) = 0$. The theory on MI and CMI has been deeply studied and widely used for GRN inference[8,14,16–19]. PCA-CMI[8] is a famous method which applied these measure to detect the dependency between genes. The PCA-CMI is computationally feasible and runs fast to infer GRNs. It is notable that result of PCA-CMI is dependent on the sequential ordering of vertices. It means that by changing the sequential ordering of vertices, the result of PCA-CMI may be different.

1.1.3 PC Algorithm based on Conditional Mutual Information (PCA-CMI)

The PCA-CMI is computationally feasible and runs fast to infer GRNs.[8]. PCA-CMI is an iterative algorithm and starts with a complete graph. The following describes the process of PCA-CMI which reconstructs GRNs by extracting skeleton S_i from S_{i-1}.

Let S_i ($i = -1$) be a complete undirected graph with vertex set X. This algorithm starts by setting $i = i+1$. Let X and Y are adjacent in S_{i-1}, and $V_{XY} = ADJ(X) \bigcap ADJ(Y)$, where $ADJ(X)$ denotes the set of adjacent vertices of $X \in \mathbf{X}$. Suppose that the size of V_{XY}, $|V_{XY}|$, is j. If $i \leq j$, for each i-subset of V_{XY} such as $\mathbf{M} = \{m_1, ..., m_i\}$, the i-order $CMI(X,Y|M)$ is computed using Equation 2. Let $CMI_{max}(X,Y|i) = \max_M CMI(X,Y|M)$. If $CMI_{max}(X,Y|i) < \theta$ (θ denotes the threshold for CMI test), the edge between X and Y is removed from S_{i-1}. Let S_i be the skeleton of the constructed graph, the above process are repeated until $S_{i-1} = S_i$ (i denotes the order of the algorithm). It is notable that result of PCA-CMI is dependent on the sequential ordering of vertices. It means that by changing the sequential ordering of vertices, the result of PCA-CMI may be different. Details of PCA-CMI are given in Table S1. PCA-CMI is an iterative algorithm and starts with a complete graph. The process of PCA-CMI is indicated in Figure S1.

According to the microarray data set for the expression values of genes $G1, G2, G3, G4$ and $G5$ in different samples, the process of PCA-CMI which reconstructs GRNs is indicated in Figure S1. In step 1, the complete undirected graph S_i ($i = -1$) is generated according to the number of genes. In step 2, MI values are computed for each pairs of

Table S1: The PC Algorithm based on CMI test (PCA-CMI)[8]

Start with a complete undirected graph S_{-1}.
$i = 0$.
  Repeat
   For each $X \in \mathbf{X}$,
    For each $Y \in ADJ(X)$,
     Determine if there is $\mathbf{M} \subseteq V_{XY}$ with $|\mathbf{M}| = i$, such that X and Y given $\mathbf{M}$ are independent.
      If this set exists,
        remove the edge between X and Y from S_{i-1}.
$i = i + 1$.
Until $i \leq |V_{XY}|$.

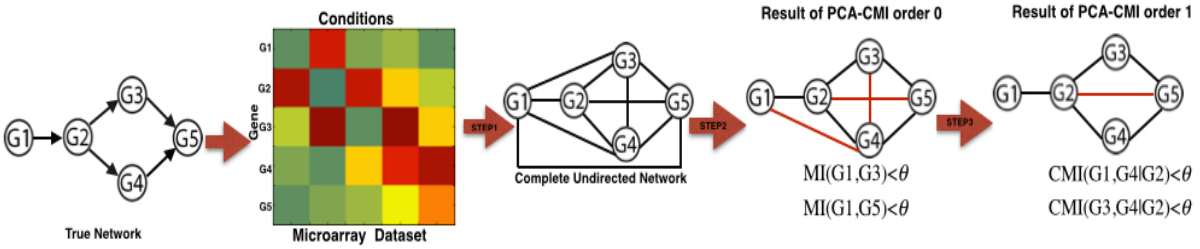

Figure S1: Diagram of the PCA-CMI. Microarray data set indicates gene expression data sets under different conditions (samples). MI denotes the mutual information value between two genes. CMI indicates the conditional mutual information value. The resulted PCA-CMI is compared to true network to evaluate.

genes and compared to the threshold θ. If the value of MI is smaller than a threshold, the edge between two genes is deleted. Based on these MI values and threshold θ, edges $G1-G3$ and $G1-G5$ are removed from complete network. In step 3, the first-order CMI between genes are calculated. By using θ, we have $CMI(G1,G4|G2) < \theta$ and $CMI(G3,G4|G2) < \theta$, so edges $G1-G4$ and $G3-G4$ are removed. In comparison with the true network, edges $G1-G2$, $G2-G4$, $G2-G3$, $G4-G5$ and $G3-G5$ are correctly identified and edge $G2-G5$ is incorrectly identified by PCA-CMI.

1.2 Methods

1.2.1 SORDER Algorithm

To clearly describe the SORDER Algorithm, some notations need to be introduced. The degree of a vertex $X \in \mathbf{X}$ in graph G is represented by $Deg_G X$ which equals to the number of vertices of G adjacent to X. A regular graph is a graph in which all vertices have the same degree. Let R be a subset of $\mathbf{X}$, the induced subgraph of G generated by R, $G[R]$, is a graph with vertex set R and edges of G with both ends in R. Edges of $E(G[R])$ represent the links

between vertices of R in G. For set $\mathbf{M}$, $|\mathbf{M}|$ indicates the number of elements of $\mathbf{M}$. Suppose that, $\mathbf{O}_1$ and $\mathbf{O}_2$ denote the sequential ordering of $(X_1,...,X_k)$ and $(Y_1,...,Y_m)$, respectively. The concatenation of $\mathbf{O}_1$ and $\mathbf{O}_2$, represented by $\mathbf{O}_1 \bigoplus \mathbf{O}_2$, is considered to be $\mathbf{O}_1 \bigoplus \mathbf{O}_2 = (X_1,...,X_k,Y_1,...,Y_m)$.

To overcome the disadvantage of PC algorithm i.e. the dependence on the sequential ordering of vertices, the SORDER algorithm is proposed to select a suitable sequential ordering of vertices (genes) based on the degrees of vertices in S_0 (skeleton of order 0 resulted by PCA-CMI).

Node selection in SORDER algorithm is based on the maximum degree of nodes. In each step of SORDER algorithm, node with maximum degree is selected, and the selection process are repeated until all nodes of graph are chosen. When some nodes have the same degree in one step, an induced subgraph of these nodes are constructed. In a similar manner, nodes in the induced subgraph are selected according to the maximum degree.Table S2 gives the detail of SORDER algorithm.

Table S2: **SORDER** Algorithm

1: Set $\mathbf{O} = \emptyset$.
2: SORDER ($G[\mathbf{X}]$).
3: { If $G[\mathbf{X}]$ is a regular graph,
$\mathbf{O} \leftarrow \mathbf{O} \bigoplus$ random sequential ordering of $\mathbf{X}$.
else
For each $0 \leq u \leq n$,
R_u=$\{X|Deg_G X$=$u\}$,
SORDER ($G[R_u]$) }.

1.2.2 CN Algorithm

The PCA-CMI depends not only on the sequential ordering of vertices, but also on the threshold value which is used for CMI test. In the previous section, a suitable sequential ordering of vertices is determined using SORDER algorithm. Different networks can be resulted by using different threshold values for CMI tests.

Suppose that $[a \;\; b]$ is an interval of threshold values for CMI tests. For each $\theta_i = a + i\frac{b-a}{100}, \quad 0 \leq i < 100$, network G_i is constructed based on PCA-CMI using the sequential ordering generated by SORDER algorithm (SPCA-CMI) with threshold θ_i and its sequential ordering. For two vertices X and Y, we define $W(XY) = \frac{|\{i|XY \in E(G_i)\}|}{100}$. $W(XY)$ denotes the reliability value of dependency between X and Y. In the CN algorithm, edge XY is removed from the complete graph when, $W(XY) < \varphi$, for threshold value φ. In order to assign a suitable value for φ, the Receiver Operating Characteristic (ROC) curve is drawn by the TPR and FPR obtained by the CN algorithm. In this work we consider $\varphi = 0.6$ a value in which TPR and FPR have simultaneously appropriate values. Table S3 gives this process of the CN algorithm.

Table S3: Details of the CN algorithm. SPCA-CMI: PCA-CMI based on sequential ordering of genes which obtained by SORDER algorithm.

1: For each $\theta_i = a + i\frac{b-a}{100},\ 0 \leq i < 100$, Run SPCA-CMI using threshold θ_i to reconstruct graph $G_i = (\mathbf{X}, E(G_i))$.
2: For two vertices $X \in \mathbf{X}$ and $Y \in \mathbf{X}$ define $W(XY) = \frac{\lvert\{i \lvert XY \in E(G_i)\}\rvert}{100}$.
3: Reconstruct graph G=(**X**,E(G)) where **X** denotes the set of vertices and $E(G) = \{XY \lvert W(XY) \geq 0.6\}$.

1.2.3 CNMIT Algorithm

The resulted CN is an undirected network. We introduce an algorithm, namely CNMIT algorithm, to give direction to edges of skeleton that are determined by CN algorithm. In order to describe the CNMIT algorithm, some definitions need to be presented.

The score-based searching methods assign a score to each network structure. Then, the structure with the highest score is selected by the search algorithm. Given a scoring function, the task is to find the highest-scoring network structure among the set of all possible network structures.

3.3.1 Search Algorithm

A Hill climbing (HC) algorithm is particularly popular in search algorithms. A HC algorithm traverses the search space by starting from an initial DAG, then an iterative procedure is repeated. At each procedure, only local changes such as edge addition which inserts a single edge between two nonadjacent vertices, edge deletion which removes a single edge between two vertices and edge reversal which reverses the direction of a single edge between two vertices are considered. The algorithm stops when there is no local change improvement of the scoring function value. Obviously, when we work with heuristic search algorithms we are not guaranteed to find a global optimal structure but only a local optimal structure. In order to scape local maximum, the algorithm can restart with different initial DAGs[20].

3.3.2 Scoring Function

There are many scoring functions to measure the degree of fitness of a DAG, G, to a data set D[10,21–27]. One of the famous scoring functions is a MIT score which is defined as follows[10]:

$$g_{MIT}(G,D) = \sum_{i=1,\ Pa_M(X_i)\neq\emptyset}^{n} (2NMI_D(X_i, Pa_M(X_i)) - \max_{\sigma_i} \sum_{j=1}^{s_i} \chi_{\alpha, l_{i,\sigma_i(j)}}), \tag{3}$$

where N denotes the total number of measurements in D and $MI_D(X_i, Pa_M(X_i))$ is the mutual information between X_i and its parents as estimated from D. Given a graph G, let s_i be the number of parents of X_i, i.e $s_i = |Pa_M(X_i)|$. $\chi_{\alpha, l_{i,\sigma_i(j)}}$ is the value that $P(\chi^2(l_{i,\sigma_i(j)}) \leq \chi_{\alpha, l_{i,\sigma_i(j)}}) = \alpha$ (the Chi-squared distribution at significance level $1-\alpha$), the term $l_{i,\sigma_i(j)}$ denotes the degrees of freedom and determined by:

$$l_{i,\sigma_i(j)} = \begin{cases} (r_i - 1)(r_{i\sigma_i(j)} - 1)\prod_{k=1}^{j-1} r_{i\sigma_i(k)} & j = 2, ..., s_i \\ (r_i - 1)(r_{i\sigma_i(1)-1}) & j = 1 \end{cases} \tag{4}$$

where $\sigma_i = [\sigma_i(1), ..., \sigma_i(s_i)]$ indicates any permutation of the index set $(1, ..., s_i)$ of the variables in $Pa_M(X_i) = \{X_{i1}, ..., X_{is_i}\}$. For more details of MIT score see [10].

In this work, we apply the search algorithm (HC algorithm) based on the scoring function (MIT score) to give direction to the resulted CN algorithm. The details of CNMIT algorithm are given in Table S4.

Table S4: Details of the CNMIT algorithm. CN: Consensus Network Algorithm; HC algorithm: Hill Climbing algorithm.

1:	Run CN algorithm to obtain skeleton S
2:	Apply the HC algorithm based on MIT score to direct the edges of S

1.3 Results

Two real data set are applied in this work. Figure S2 illustrates the structure of Gene-gene interaction between the nine genes of SOS network in E.coil[28]. The time course profiles for a set of 8 genes, part of the SOS pathway

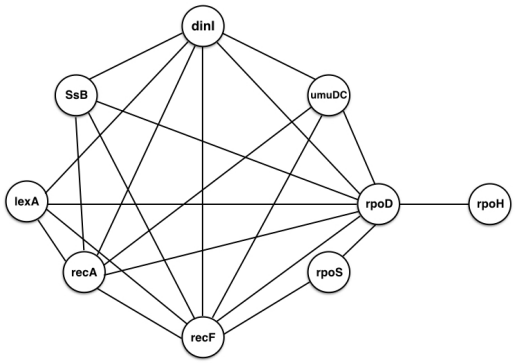

Figure S2: E. coli SOS pathway for 9 genes.

of *E. coli*[29] are selected[29]. Figure S3 illustrates the structure of E.coil network which contains 8 genes and 7 edges. Table S5 indicates results of SPCA-CMI and CN algorithms and methods studied by Zhang[8] for DREAM3

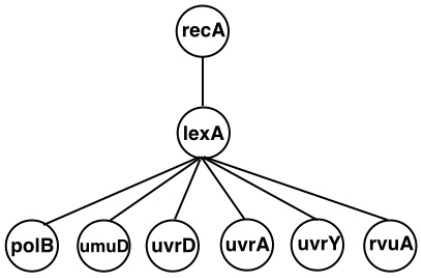

Figure S3: E. coli SOS pathway of 8 genes.

with 10 nodes and 10 edges. Results of this table are related to the threshold value of 0.05 for MI and CMI tests in PCA-CMI. As shown by Table S5, TP, ACC, PPV, F-measure, MCC and TPR under other algorithms are less than those of the CN algorithm. Therefore, it can be deduced that the CN algorithm is more powerful in structure learning than other methods studied by Zhang. Results of employing proposed algorithms on DREAM3 Challenge

Table S5: Comparison of different methods for Learning DREAM3 Challenge with 10 genes and 10 edges. LASSO: regression based method; LP: linear programming based method; PCA-PCC: PC-algorithm based on partial correlation coefficient; MI: MI method; PCA-CMI: Path Consistency Algorithm based on Conditional Mutual Information with threshold 0.05 for CMI; SPCA-CMI: PCA-CMI based on sequential ordering of genes which obtained by SORDER algorithm; CN: Consensus Network Algorithm. The best performer for the relative item is noted in bold.

Method	TP	FP	TPR	FDR	PPV	ACC	MCC	F-measure
LASSO	7	34	0.7	0.83	0.17	0.17	-0.4	0.27
LP	2	21	0.2	0.91	0.08	0.36	-0.33	0.12
PCA-PCC	8	**1**	0.8	**0.1**	0.89	0.93	0.8	0.84
MI	**9**	3	**0.9**	0.25	0.75	0.91	0.77	0.82
PCA-CMI	7	**1**	0.7	0.13	0.87	0.91	0.73	0.78
SPCA-CMI	7	**1**	0.7	0.13	0.87	0.91	0.73	0.78
CN	**9**	**1**	**0.9**	**0.1**	**0.9**	**0.96**	**0.87**	**0.9**

database with 50 genes and 77 edges have been represented in Table S6. The value of 0.05 and 0.1 are chosen as the

Table S6: Comparison of different methods for learning DREAM3 Challenge with 50 genes and 77 edges. LASSO: regression based method; LP: linear programming based method; PCA-PCC: PC-algorithm based on partial correlation coefficient; MI: MI method; PCA-CMIZ: Path Consistency Algorithm based on Conditional Mutual Information with threshold 0.1 for CMI which reported by Zhang, 2012; PCA-CMI: Path Consistency Algorithm based on Conditional Mutual Information with threshold 0.05 for CMI; SPCA-CMI: PCA-CMI based on sequential ordering of genes which obtained by SORDER algorithm; CN: Consensus Network Algorithm. The best performer for the relative item is noted in bold.

Method	TP	FP	TPR	FDR	PPV	ACC	MCC	F-measure
LASSO	30	32	0.39	0.52	0.49	0.94	0.4	0.43
LP	8	**7**	0.1	0.45	0.55	0.94	0.22	0.17
PCA-PCC	35	93	0.45	0.73	0.27	0.89	0.3	0.34
MI	38	55	0.5	0.59	0.41	0.92	0.41	0.45
PCA-CMIZ	22	21	0.29	0.49	0.51	0.93	0.35	0.36
PCA-CMI	30	34	0.39	0.53	0.47	0.93	0.4	0.43
SPCA-CMI	32	32	0.42	0.5	0.5	0.94	0.42	0.46
CN	**39**	29	**0.51**	**0.42**	**0.58**	**0.95**	**0.51**	**0.54**

thresholds for CMI tests to determine the presence of dependency between genes. The TP, ACC, PPV, F-measure, MCC and TPR measures receive larger values applying SPCA-CMI and CN algorithms to infer GRNs. This shows that the CN algorithm has superior performance. Results of DREAM3 with 100 nodes and 166 edges have been illustrated in Table S7. To determine the dependency among genes, both 0.05 and 0.1 thresholds for CMI tests are considered. As shown by Table S7, the ACC, PPV, F and MCC measures received larger values by CN algorithm for inferring about DREAM3 with 100 nodes.

Table S7: Comparison of different methods for Learning DREAM3 Challenge with 100 genes and 166 edge. LASSO: regression based method; LP: linear programming based method; PCA-PCC: PC-algorithm based on partial correlation coefficient; MI: MI method; PCA-CMIZ: Path Consistency Algorithm based on Conditional Mutual Information with threshold 0.1 for CMI which reported by Zhang, 2012; PCA-CMI: Path Consistency Algorithm based on Conditional Mutual Information with threshold 0.05 for CMI; SPCA-CMI: PCA-CMI based on sequential ordering of genes which obtained by SORDER algorithm; CN: Consensus Network Algorithm. The best performer for the relative item is noted in bold.

Method	TP	FP	TPR	FDR	PPV	ACC	MCC	F-measure
LASSO	**132**	1549	**0.79**	0.92	0.08	0.68	0.18	0.14
LP	36	327	0.22	0.9	0.1	0.91	0.1	0.1
PCA-PCC	97	204	0.58	0.68	0.32	0.95	0.41	0.41
MI	84	119	0.51	0.59	0.41	0.96	0.44	0.45
PCA-CMIZ	46	**25**	0.27	0.35	0.64	0.97	0.41	0.38
PCA-CMI	57	48	0.34	0.46	0.54	0.97	0.42	0.42
SPCA-CMI	68	36	0.41	0.35	0.65	0.97	0.51	0.51
CN	72	35	0.43	**0.33**	**0.67**	**0.98**	**0.53**	**0.53**

Table S8: Best result for SPCA-CMI algorithm. TH: The threshold in which SPCA-CMI algorithm achieve the best result; D10: DREAM10; D50: DREAM50; D100: DREAM100.

Data	TH	TP	FP	TPR	FDR	PPV	ACC	MCC	F-measure
D10	0.1	8	0	0.8	0	1	0.96	0.87	0.89
D50	0.037	38	32	0.5	0.02	0.54	0.94	0.49	0.52
D100	0.037	79	44	0.48	0.01	0.62	0.97	0.53	0.54

Tables S5 to S7 show that CN algorithm not only can remarkably reduce the number of FP but also it can successfully find more true edges. Results of using different algorithms for the real data set with 9 genes have been represented in Table S9. The threshold value for CMI test is determined to be 0.01. Table S9 indicates that TP, ACC, PPV, F-measure, MCC and TPR measures receive larger values when applying the CN algorithm to infer GRNs.

Table S8 represents the best results for SPCA-CMI algorithm. First column corresponds to different networks that are studied in this work. Second column considers threshold values for CMI test. From this table, it is demonstrated that SPCA-CMI algorithm dominate the method of PCA-CMI. Tables S10 and S11 indicate results of CNMIT algorithm and methods studied by Zhang[16] for DREAM3 with 50 and 100 nodes. According to these tables LASSO2012, LPMethod and RO have large value for FP. ARACNE contains best TP value in comparison other methods but it has FP strictly higher than GENIE3, NARROMI and CNMIT methods. Results of CN algorithm contain small value for FP, so FP values for CNMIT algorithm are strictly less than those of other methods for directed networks.

Table S9: Comparison of different methods for learning SOS network with 9 genes. LASSO: regression based method; LP: linear programming based method; PCA-PCC: PC-algorithm based on partial correlation coefficient; MI: MI method; PCA-CMIZ: Path Consistency Algorithm based on Conditional Mutual Information with threshold 0.01 for CMI which reported by Zhang, 2012; SPCA-CMI: PCA-CMI based on sequential ordering of nodes which obtained by SORDER algorithm; CN: Consensus Network Algorithm. The best performer for the relative item is noted in bold.

Method	TP	FP	TPR	FDR	PPV	ACC	MCC	F-measure	AUC
LASSO	9	4	0.37	0.31	0.69	0.47	0.04	0.48	0.61
LP	5	**1**	0.21	0.16	0.83	0.44	0.16	0.33	0.59
PCA-PCC	12	3	0.5	0.2	0.8	0.58	0.24	0.61	0.67
MI	16	3	0.66	0.16	0.84	0.69	0.39	0.74	0.73
PCA-CMIZ	16	4	0.67	0.2	0.8	0.67	0.32	0.73	0.74
SPCA-CMI	16	4	0.67	0.2	0.8	0.67	0.32	0.73	0.74
CN	**19**	3	**0.8**	**0.14**	**0.86**	**0.78**	**0.52**	**0.83**	**0.75**

Table S10: Results of directed networks. Comparison of different methods for Learning DREAM3 Challenge with 50 genes and 77 edges. LASSO2012: regression based method; LPMethod: linear programming based method; RO: recursive optimization based method; ARACNE: MI-based method; GENIE3-FR-sqrt0: GENIE3 method with parameter 'sqrt for one case; GENIE3-FR-all: GENIE3 method with parameter 'all' that is time consuming in this case; NARROMI: method based on RO and MI; CNMIT: Consensus Network algorithm based on MIT score. The best performer for the relative item is noted in bold.

Method	TP	FP	TPR	FDR	PPV	ACC	MCC	F-measure
LASSO2012	27	148	0.35	0.84	0.15	0.91	0.19	0.21
LPMethod	30	97	0.39	0.76	0.23	0.94	0.27	0.29
RO	38	150	0.5	0.8	0.2	0.92	0.28	0.28
ARACNE	**45**	94	**0.6**	0.7	0.33	0.95	0.4	0.42
GENIE3-FR-sqrt	37	89	0.48	0.71	0.29	0.94	0.34	0.36
GENIE3-FR-all	34	83	0.44	0.71	0.28	0.94	0.3	0.34
NARROMI	41	71	0.53	0.63	0.36	0.95	**0.41**	**0.43**
CNMIT	30	**38**	0.39	**0.55**	**0.44**	**0.97**	0.4	0.42

Table S11: Results of directed networks. Comparison of different methods for Learning DREAM3 Challenge with 100 genes and 166 edges. LASSO2012: regression based method; LPMethod: linear programming based method; RO: recursive optimization based method; ARACNE: MI-based method; GENIE3-FR-sqrt0: GENIE3 method with parameter 'sqrt' for one case; GENIE3-FR-all: GENIE3 method with parameter 'all' that is time consuming in this case; NARROMI: method based on RO and MI; CNMIT: Consensus Network algorithm based on MIT score. The best performer for the relative item is noted in bold.

Method	TP	FP	TPR	FDR	PPV	ACC	MCC	F-measure
LASSO2012	30	497	0.18	0.94	0.05	0.93	0.04	0.07
LPMethod	67	450	0.4	0.87	0.12	0.94	0.2	0.19
RO	62	832	0.37	0.93	0.06	0.9	0.12	0.11
ARACNE	**84**	322	**0.5**	0.79	0.2	0.95	0.3	0.29
GENIE3-FR-sqrt	70	198	0.42	0.73	0.26	0.97	0.31	0.31
GENIE3-FR-all	74	255	0.44	0.77	0.22	0.96	0.3	0.29
NARROMI	77	237	0.46	0.75	0.24	0.96	0.31	0.31
CNMIT	58	**59**	0.35	**0.5**	**0.5**	**0.98**	**0.41**	**0.41**

2 User Manual

2.1 Introduction

CN algorithm is a Matlab code for learning Gene Regulatory Network (GRN) structure. It implements algorithm for learning the structure using the Mutual Information Test (MIT) criterion, as presented in[8]. In CNMIT algorithm, HC algorithm based on MIT score is applied to direct the edges of resulted structure by CN algorithm. Softwares are in the form of MATLAB and JAVA codes. The source of data sets and codes are available at $http://bs.ipm.ir/softwares/CN$. You can download all functions and data set in this site.

2.2 Installation

The Matlab codes are ready for use. The main files are listed in Tables S12 which contains all the main functionas.

Table S12: The Main functions for SORDER, CN and CNMIT algorithms

File	Description
pca-cmi	An algorithm for inferring gene regulatory networks from gene expression data.
sorder.m	The SORDER algorithm to select a suitable sequential ordering of vertices.
MBluePen.m	Is a java path for RedPen2.jar.
RedPen2.jar	Is a java code for selecting a network with maximum MIT score.
direct.m	Direct the result of CN algorithm.
cn.m	A heuristic algoritm to infer gene regulatory networks from gene expression data.
cnmit.m	A heuristic algoritm to infer direct gene regulatory networks from gene expression data.
compareTrue.m	Compare a network with a true directed network.
compare.m	Compare a network with a true network.
demo.m	A walk-through demonstration

2.3 Usage and Examples

In this section, we demonstrate the use of our algorithms for structure learning via a set of walk-through examples. The code for these examples can be found in the demo.m file. We first load the Dream3 data set and its true network by the following codes:

data=load('Dream10.txt');

load('TT.txt');

'data file' is expression of variable, in which row is sample and column is the variable. 'True Network.txt' is an undirected true network for the data set.

DREAM3 Challenge data sets with n variables and n samples ($n = 10, 50, 100$) contain true networks and data

generated from these networks as described in [30–32].

PCA-CMI

We first get the structure of network set by PCA-CMI. For example:

lamda=0.05;

order0=2;

$[G, Gval, order] = pca - cmi(data, lamda, order0)$;

This function takes three parameters:

- data: Gene expression data.
- lamda: The significance level for the mutual information test of independence.
- order0: Is the order of the algorithm.

This function returns three outputs:

- G: The 0-1 symmetric network or graph after PCA-CMI. The value of 1 indicates the existence of edges between two mentioned nodes.
- Gval: The network which shows the strength dependence. It contain values of MI or CMI.
- order: Is the order of the PCA-CMI.

SORDER algorithm

The SORDER algorithm is proposed to select a suitable sequential ordering of vertices (genes) based on the degrees of vertices in G.

$[sor] = sorder(G)$;

This function takes one parameter:

- G: The 0-1 adjacency matrix for resulted network by PCA-CMI.

This function returns one output:

- sor: A suitable sequential ordering of vertices (genes).

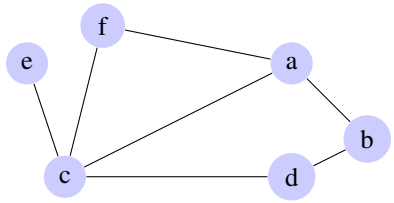

Figure S4: Network G with vertex set $X = \{a,b,c,d,e,f\}$.

For example for a structure of network given in Figure S4, adjacency matrix G is determined by:

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array} \begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

The result of SORDER algorithm is a sequential order : $O = \{c,a,b,d,f,e\}$.

CN algorithm

In CN algorithm we applied an interval threshold for dependency determination to construct consensus network. We also apply SORDER algorithm to select the sequential ordering of vertices.

This algorithm takes an interval threshold for MI and CMI value such as:

lamdaa=0.03:0.001:0.05;

It also gets a threshold for construct consensus network.

th=0.6;

$[GGTT] = cn(data, lamdaa, order0, th)$;

This function takes four parameters:

- data: Gene expression data.
- lamdaa: The interval threshold for the mutual information test of independence.
- order0: Is the order of the algorithm.
- th: a threshold for construct consensus network.

This function returns one output:

- GGTT: The 0-1 symmetric network or graph after CN algorithm.

The MIT score is implemented within the Elvira system (a JAVA package for learning the structure of BN[10]). The Elvira package can be downloaded from http://leo.ugr.es/elvira/.
The MIT score is available at /bayelvira2/elvira/learning/MITMetrics.java. In this study, we rewrite the MIT score program (Red.Pen2) which, in comparison to the Elvira system, reduces running time and memory occupied by the algorithm[33]. MBluePen.m is a java path for RedPen2.jar. For example when you save the RedPen2.jar in drive C in folder program, you have: $'/C:/Users/program/RedPen2.jar'$.

CNMIT algorithm

The result of CN algorithm is an undirected graph, HC algorithm based on MIT score is applied to direct edges of resulted network by CN algorithm.
For example we have:
$[finalgraphCon] = cnmit(network, data, chi2States, retryCount, hillClimbing)$;. This function takes five parameters:

- network:The resulted network by CN algorithm.
- data:Gene expression data.
- chi2States:The number of states which considered for variables.
- retryCount:The number of repetition for Hill Climbing (HC) algorithm.
- hillClimbing:The default is True, it means that HC algorithm is used for search method.

This function has one output:

- finalgraphCon: The 0-1 non symmetric direct network. 1 indicates the existence of direct edge from parent to child.

References

1. F. V. Jensen, *An introduction to Bayesian networks*, UCL Press London, 1996, vol. 210.
2. P. Spirtes, C. Meek and T. Richardson, Proceedings of the Eleventh conference on Uncertainty in artificial intelligence, 1995, pp. 499–506.
3. P. Spirtes, C. N. Glymour and R. Scheines, *Causation, prediction, and search*, MIT press, 2000, vol. 81.

4. P. Spirtes, Proc. of the Eighth International Workshop on Artificial Intelligence and Statistics, 2001, pp. 213–221.
5. J. Zhang, *Artificial Intelligence*, 2008, **172**, 1873–1896.
6. D. Colombo, M. H. Maathuis, M. Kalisch, T. S. Richardson *et al.*, *The Annals of Statistics*, 2012, **40**, 294–321.
7. T. Claassen, J. Mooij and T. Heskes, *arXiv preprint arXiv:1309.6824*, 2013.
8. X. Zhang, X.-M. Zhao, K. He, L. Lu, Y. Cao, J. Liu, J.-K. Hao, Z.-P. Liu and L. Chen, *Bioinformatics*, 2012, **28**, 98–104.
9. S. Imoto, T. Goto, S. Miyano *et al.*, Pacific Symposium on Biocomputing, 2002, pp. 175–186.
10. L. M. De Campos, *The Journal of Machine Learning Research*, 2006, **7**, 2149–2187.
11. E. Faulkner, Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on, 2007, pp. 18–25.
12. S. Acid and L. M. de Campos, *International Journal of Approximate Reasoning*, 2001, **27**, 235–262.
13. I. Tsamardinos, L. E. Brown and C. F. Aliferis, *Machine Learning*, 2006, **65**, 31–78.
14. H. Brunel, J.-J. Gallardo-Chacón, A. Buil, M. Vallverdú, J. M. Soria, P. Caminal and A. Perera, *Bioinformatics*, 2010, **26**, 1811–1818.
15. P. E. Meyer, F. Lafitte and G. Bontempi, *BMC Bioinformatics*, 2008, **9**, 461.
16. X. Zhang, K. Liu, Z.-P. Liu, B. Duval, J.-M. Richer, X.-M. Zhao, J.-K. Hao and L. Chen, *Bioinformatics*, 2013, **29**, 106–113.
17. N. A. Ahmed and D. Gokhale, *Information Theory, IEEE Transactions on*, 1989, **35**, 688–692.
18. M. Kalisch and P. Bühlmann, *The Journal of Machine Learning Research*, 2007, **8**, 613–636.
19. S. Saito, X. Zhou, T. Bae, S. Kim and K. Horimoto, Bioinformatics and Biomedicine (BIBM), 2010 IEEE International Conference on, 2010, pp. 296–301.
20. J. A. Gámez, J. L. Mateo and J. M. Puerta, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, Springer, 2007, pp. 585–597.
21. W. Buntine, Proceedings of the Seventh conference on Uncertainty in Artificial Intelligence, 1991, pp. 52–60.
22. D. Heckerman, D. Geiger and D. M. Chickering, *Machine Learning*, 1995, **20**, 197–243.
23. M. Kayaalp and G. F. Cooper, Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence, 2002, pp. 251–258.
24. C. Chow and C. Liu, *Information Theory, IEEE Transactions on*, 1968, **14**, 462–467.
25. W. Lam and F. Bacchus, *Computational Intelligence*, 1994, **10**, 269–293.
26. R. R. Bouckaert, *Bayesian belief networks: from construction to inference*, Universiteit Utrecht, Faculteit Wiskunde en Informatica, 1995.
27. N. Friedman and M. Goldszmidt, *Learning in graphical models*, Springer, 1998, pp. 421–459.
28. A. Baralla, M. Cavaliere and A. de la Fuente, 2008.

29. M. Ronen, R. Rosenberg, B. I. Shraiman and U. Alon, *Proceedings of the national academy of sciences*, 2002, **99**, 10555–10560.

30. D. Marbach, R. J. Prill, T. Schaffter, C. Mattiussi, D. Floreano and G. Stolovitzky, *Proceedings of the National Academy of Sciences*, 2010, **107**, 6286–6291.

31. D. Marbach, T. Schaffter, C. Mattiussi and D. Floreano, *Journal of computational biology*, 2009, **16**, 229–239.

32. R. J. Prill, D. Marbach, J. Saez-Rodriguez, P. K. Sorger, L. G. Alexopoulos, X. Xue, N. D. Clarke, G. Altan-Bonnet and G. Stolovitzky, *PloS one*, 2010, **5**, e9202.

33. R. Aghdam, M. Ganjali and C. Eslahchi, *PloS one*, 2014, **9**, e92600.