2012

INFORMATION THEORETIC APPROACHES TOWARDS REGULATORY NETWORK INFERENCE

Vijender Chaitankar
Virginia Commonwealth University

INFORMATION THEORETIC APPROACHES TOWARDS REGULATORY NETWORK INFERENCE

Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy at Virginia Commonwealth University

by

VIJENDER CHAITANKAR

B.E., Dr. B.A.M.U., India - September 2000 to April 2004

M.S., The University of Southern Mississippi, USA - Fall 2005 to Fall 2006

Director: Dr. Preetam Ghosh, Assitant Professor

Department of Computer Science

Virginia Commonwewalth University

Richmond, Virginia

December 2012

ACKNOWLEDGEMENTS

First and foremost I would like to thank my advisor Dr. Preetam Ghosh for his thoughtful guidance, encouragement and constant support throughout my tenure as a Ph.D. student. It has been an honor to be his first Ph.D. student. I appreciate his contribution of time, ideas, and funding to make my Ph.D. dissertation productive. I would also like to thank the other members of my dissertation committee Dr. Krzysztof Cios, Dr. Vojislav Kecman, Dr. Ramana Pidaparti, and Dr. Michael Mayo. Their thoughtful suggestions and inputs are greatly appreciated.

I am also thankful for the members of Environmental Lab, Engineer Research and Development Center, Vicksburg, Mississippi for their continuous help and support. The group has been a source of good advice and collaboration. I would like to especially thank Dr. Kurt A. Gust and Dr. Edward J. Perkins for their suggestions and inputs in my research work.

I am appreciative of my fellow lab members, Ahmed Abdelzaher, Bhanu Kamapantula, and Joseph Nalluri for their patience, constant source of support, and team work on various projects.

Lastly, I would like to extend my deepest gratitude to my family for all their unconditional love and encouragement. My parents, brothers, and my sisters have always provided me with love and encouragement.

TABLE OF CONTENTS

LIST OF ALGORITHMS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

INFORMATION THEORETIC APPROACHES TOWARDS REGULATORY NETWORK INFERENCE

by Vijender Chaitankar

Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy at Virginia Commonwealth University

Virginia Commonwealth University, 2012

Major Director: Dr. Preetam Ghosh, Assistant Professor, Department of Computer Science

In spite of many efforts in the past, inference or reverse engineering of regulatory networks from microarray data remains an unsolved problem in the area of systems biology. Such regulatory networks play a critical role in cellular function and organization and are of interest in the study of a variety of disease areas and ecotoxicology to name a few. This dissertation proposes information theoretic methods/algorithms for inferring regulatory networks from microarray data. Most of the algorithms proposed in this dissertation can be implemented both on time series and multifactorial microarray data sets. The work proposed here infers regulatory networks considering the following six factors: (i) computational efficiency to infer genome-scale networks, (ii) incorporation of prior biological knowledge, (iii) choosing the optimal network that minimizes the joint network entropy, (iv) impact of higher order structures (specif-

ically 3-node structures) on network inference (v) effects of the time sensitivity of regulatory interactions and (vi) exploiting the benefits of existing/proposed metrics and algorithms for reverse engineering using the concept of consensus of consensus networks.

Specifically, this dissertation presents an approach towards incorporating knock-out data sets. The proposed method for incorporating knock-out data sets is flexible so that it can be easily adapted in existing/new approaches.

While most of the information theoretic approaches infer networks based on pair-wise interactions this dissertation discusses inference methods that consider scoring edges from complex structures. A new inference method for building consensus networks based on networks inferred by multiple popular information theoretic approaches is also proposed here. For time-series datasets, new information theoretic metrics were proposed considering the time-lags of regulatory interactions estimated from microarray datasets. Finally, based on the scores predicted for each possible edge in the network, a probabilistic minimum description length based approach was proposed to identify the optimal network (minimizing the joint network entropy).

Comparison analysis on in-silico and/or real time data sets have shown that the proposed algorithms achieve better inference accuracy and/or higher computational efficiency as compared with other state-of-the-art schemes such as ARACNE, CLR and Relevance Networks. Most of the methods proposed in this dissertation are generalized and can be easily incorporated into new methods/algorithms for network inference.

CHAPTER 1

INTRODUCTION

1.1 Background

A genetic regulatory network at an abstract level can be conceptualized as a network of interconnected genes/transcription factors which respond to internal and external conditions by altering the relevant connections within the network; here a connection represents the regulation of a gene by another gene/protein. Understanding these regulatory interactions is of great importance as it can lead to a better understanding of an organisms health, which could further lead to diagnosis of diseases and its drug development [1].

Babu et al.[2] organized the various approaches towards inferring the gene regulatory networks into three fundamental categories:

1. Template based methods
2. Inferring networks by predicting cis-regulatory elements
3. Reverse engineering using gene expression data

In the *template based methods*, the known regulatory interactions from a model organism are transferred to the target genome of interest [3, 4, 5]. However, in the *inference of networks by predicting cis-regulatory elements*, relevant methods use the knowledge of experimentally well characterized transcription factor binding sites to infer the regulatory networks [6, 7, 8, 2]. While the first two approaches are more biologically oriented, the third one relies primarily on mathematical and computational

based approaches. *Reverse engineering using gene expression data* based methods scan for patterns in the time series microarray data sets [2] generally known as the gene expression matrix. This matrix is created after a series of steps performed over the microarray chip which includes image processing, transformation and normalization. In a gene expression matrix, generally a row represents a gene and a column represents a microarray experiment.

The advantage of the *reverse engineering approach using gene expression data* is that it does not require any prior knowledge to infer the regulatory networks, while incorporating such prior knowledge would yield even better inference accuracy [9].

Understanding the global properties of regulatory networks has also resulted in improved inference results. One example of such a global property based on empirical studies [1, 9] on the yeast regulatory network is that, a gene is generally regulated by a small number of transcription factors. This property greatly reduces the computational complexity of popular methods such as Dynamic Bayesian Networks (DBN) [10, 11, 12] and Probabilistic Boolean Networks (PBN) [13, 14] to name a few.

Figure 1 outlines the requirements of an inference algorithm and the process of validating any newly developed algorithm. The main input that a reverse engineering/inference algorithm requires in order to infer a network is the gene expression matrix, while other inputs such as prior knowledge (e.g., known interactions between genes) and other kinds of experimental data (e.g., gene knock out data) can also be utilized to obtain better results. The algorithm is validated by comparing inferred networks with known networks (the two well studied organisms whose transcriptional networks are well known are *Escheria-coli* and *Saccharomyces cerevisiae*). Thus, data sets from known networks are used as input to the algorithm and the inferred

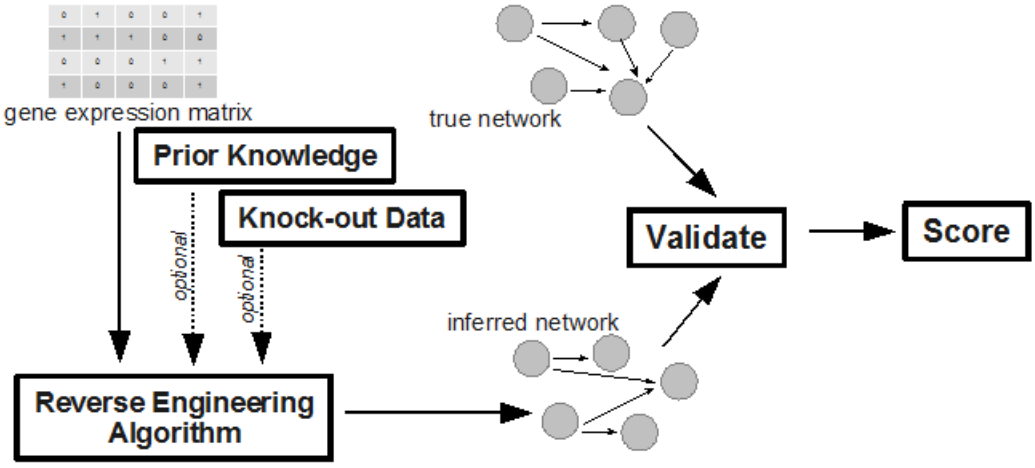


Fig. 1. Network Inference/Reverse Engineering paradigm

networks are compared with the known networks (also referred to as the true networks).

1.2 Microarray Experiments

Microarrays help biologists to record the expressions levels of genes in an organism at a genome level. A microarray is a glass plate which has thousands of spots; each of these spots contains similar DNA molecules that uniquely identify a gene. In any microarray experiment, initially the RNAs from the cells are extracted; these extracted RNAs are then reverse transcribed in to cDNAs. The cDNAs are then labeled with colored dyes and are then allowed to hybridize on the microarray glass plate. At this stage, the labeled cDNAs will hybridize to the complimentary sequences on the spots. If the concentration of a particular gene is high, then the corresponding spot on the microarray plate will show the dye color. There are a number of ways in which a microarray experiment can record expression levels. The most popular one is to compare the expression levels of a cell exposed to a particular external condition to that of a reference cell in normal condition. In this approach, cells under the two

different conditions are labeled with two different dye colors. If a particular gene is expressed in only one condition then the corresponding spot on the microarray plate shows the color for that particular condition. If it is not expressed in any condition, then the actual color of spot (usually black) shows up. It is also possible that the genes may be expressed in both conditions; in such cases a spot shows a variant color which is the combination of the two chosen colors. After hybridization the colors on the microarray plate are scanned and recorded by a machine. This microarray data is then analyzed which includes image processing, transformation and normalization [15]. Different Normalization and quantization methods might yield different networks; hence, one must be careful while selecting the proper normalization and quantization methods. The data after the analysis is used as input by the reverse engineering algorithms.

Some mathematical concepts (Information Theoretic Metrics) that will be required in the design of methods/algorithms in this dissertation are discussed in the following section.

1.3 Information Theoretic Metrics

1.3.1 Entropy

Entropy (H) is the measure of average uncertainty in a random variable. Entropy of a random variable X with probability mass function *p(x)* is defined [16] by Equation 1.1.

$$H(X) = -\sum_{x \in X} p(x).\log(x) \tag{1.1}$$

1.3.2 Mutual Information (MI)

Mutual information measures the amount of information that can be obtained about one random variable by observing another one. MI is defined [16] as shown in Equation 1.2.

$$I(X;Y) = \sum_{x,y}[p(x,y).log\frac{p(x,y)}{p(x).p(y)}] \tag{1.2}$$

MI can also be defined in terms of entropies [16] as shown in Equation 1.3.

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \tag{1.3}$$

1.3.3 Conditional Mutual Information (CMI)

Conditional Mutual Information is the reduction in the uncertainty of X due to knowledge of Y when Z is given [9]. The CMI of random variables X and Y given is defined [17] as shown in Equation 1.4.

$$I(X;Y|Z) = \sum_{x,y,z} p(x,y,z).log\frac{p(x,y|z)}{p(x|z).p(y|z)} \tag{1.4}$$

CMI can also be expressed in terms of entropies as shown in Equation 1.5.

$$I(X;Y|Z) = H(X,Z) + H(Y,Z) - H(Z) - H(X,Y,Z) \tag{1.5}$$

Figure 2 gives example calculations of the MI and CMI values.

1.4 Performance Metrics

In order to compare the accuracy of approaches, the benchmark precision (P) and recall (R) metrics are considered. While different definitions exist [18], R is

X	0	1	1	1	1	1	1	0	0	0
Y	0	0	0	1	1	0	0	1	1	1

$$H(X) = -0.4 * \log_2(0.4) - 0.6 * \log_2(0.6)$$

$$H(Y) = -0.5 * \log_2(0.5) - 0.5 * \log_2(0.5)$$

$$H(X,Y) = -0.1 * \log_2(0.1) - 0.3 * \log_2(0.3) - 0.4 * \log_2(0.4) - 0.2 * \log_2(0.2)$$

$$I(X,Y) = H(X) + H(Y) - H(X,Y) = 0.9710 + 1 - 1.8464 = 0.1246$$

(A)

X	0	1	1	1	1	1	1	0	0	0
Y	0	0	0	1	1	0	0	1	1	1
Z	1	1	0	0	0	1	1	0	0	0

$$H(Z) = -0.4 * \log_2(0.4) - 0.6 * \log_2(0.6)$$

$$H(X,Z) = -0.3 * \log_2(0.3) - 0.1 * \log_2(0.1) - 0.3 * \log_2(0.3) - 0.3 * \log_2(0.3)$$

$$H(Y,Z) = -0.1 * \log_2(0.1) - 0.4 * \log_2(0.4) - 0.5 * \log_2(0.5) - 0.0 * \log_2(0.0)$$

$$H(X,Y,Z) = -0.0 * \log_2(0.0) - 0.1 * \log_2(0.1) - 0.3 * \log_2(0.3) - 0.0 * \log_2(0.0) - 0.1 * \log_2(0.1) - 0.3 * \log_2(0.3) - 0.2 * \log_2(0.2) - 0.0 * \log_2(0.0)$$

$$I(X;Y|Z) = H(X,Z) + H(Y,Z) - H(Z) - H(X,Y,Z) = 0.5999$$

(B)

Fig. 2. Example calculations for (a) MI and (b) CMI

herein defined as $C_e/(C_e + M_e)$ and P as $C_e/(C_e + F_e)$, where C_e denotes the edges that exist in both the true and the inferred network, M_e are the edges that exist in the true network but not in the inferred network and F_e are the edges that do not exist in the true network but do exist in the inferred network.

1.5 Contributions of this Dissertation

The major contributions in this dissertation are highlighted in the following points:

1. Designed new information theoretic algorithm (PMDL algorithm) based on predictive minimum description length principle to identify the network having minimum joint entropy and hence having the highest confidence; conditional mutual information metric was used to further prune false edges from the inferred network.

2. Performed sensitivity analysis on different microarray data sizes for time series data-sets to identify the limitations of current information theoretic metrics and corresponding algorithms.

3. Proposed time lagged mutual information (TLMI) and time lagged conditional mutual information (TLCMI) metrics to improve the inference accuracy of the relevance network class of algorithms.

4. Proposed a method to incorporate knock-out data sets and biological prior knowledge for inferring regulatory networks.

5. Proposed structure based inference approaches (sCoIn, aCoIn) and their extensions to infer networks with higher accuracy.

6. Proposed a new method for building consensus networks to exploit the benefits of existing algorithms and metrics along with the proposed ones.

7. Developed a web-based tool for building consensus networks from micro-array datasets for easy access and use by biologists.

CHAPTER 2

EXISTING APPROACHES TOWARDS REGULATORY NETWORKS INFERENCE USING EXPRESSION DATA

A wide range of regulatory network inference methods have been developed so far. Some of the popular methods are Boolean Networks (BN) [19, 20, 21], Probabilistic Boolean Networks (PBN), Bayesian Networks (BN) [22, 23], Dynamic Bayesian Networks (DBN), Differential Equations Methods (DEM) [24, 25], Linear Methods (LM) [26, 27], Neural Network Methods (NNM) [28, 29] and Information Theoretic Methods [18, 30, 31, 32, 33, 34, 35]. This dissertation is more focused on the inference of regulatory networks using Information Theoretic Methods. This chapter briefly introduces some popular information theory based inference algorithms. While existing approaches such as BN, PBN, DEM, DBN etc. do infer quality networks, their capability is limited to inferring small networks. As the microarray experiments provide data for thousands of genes, these approaches cannot utilize the full potential of the available data. Usually a subset of the complete data set such as differentially expressed genes is filtered and then is used as input to these approaches. This step brings in a new problem of clustering the data set. There is no single best clustering approach that can be implemented on microarray data and different clustering techniques produce different results [36]. As information theoretic approaches are capable of inferring networks at the genome scale, the data clustering issue is eliminated. In the following sections of this chapter some of the widely used information theory based algorithms are briefly introduced.

2.1 Relevance Network

Relevance Network [30, 31] is the simplest and computationally most effective information theory based algorithm. This algorithm computes pair-wise MI between every pair of gene from the given expression data. An edge between a pair of genes exists if the MI between the two genes is greater than a user selected threshold. The quality of inferred networks heavily depends on the threshold selected which lies in the range of the minimum and the maximum pair-wise MI's computed. A small threshold gives high recall but very low precision whereas a high threshold gives better precision and very low recall. Thus, threshold selection is a major issue with Relevance Network based approaches.

2.2 ARACNE

Relevance Networks assumed that if MI is low then the genes are not connected and if MI is high then genes are connected [17]. Based on the study of chemical kinetics, it has been found that the second assumption is not true [17]. If there are two genes being regulated by a third gene, then the MI between the two genes could be high resulting in a false edge in the network. ARACNE [32, 33] is the first inference algorithm to implement a method to identify and prune such false edges. ARACNE states that if the MI between two genes X and Y is less than or equal to that between genes X and Z or between Y and Z, i.e. $I(X,Y)[I(X,Z), I(Y,Z)]$, then there is no connectivity between X and Y.

2.3 REVEAL

REVEAL [34] aims at inferring rules for interaction between genes from the Boolean states of the genes given by the gene expression matrix and by considering the

mutual information between genes. The algorithm starts with computing pair-wise mutual information between all pairs of genes. If the ratio of the mutual information between the regulating gene and the entropy of the gene being regulated equals one, REVEAL states that a regulatory interaction exists between these two genes. If not all genes are being regulated with another gene, REVEAL next proceeds to see if a combination of two genes regulate the remaining genes. If, the ratio of the MI and entropy (as explained above) equals one, then the set of two genes does regulate the other gene under consideration. If there are still other genes where the ratio did not equal one, REVEAL will next consider gene sets of three to interact with the given gene of interest and so on. Thus, the algorithm iteratively considers more complex genetic interactions thereby having exponential time and space complexities [37].

2.4 CLR

Relevance network algorithm first proposed the use of mutual information metric for reverse engineering of regulatory networks. REVEAL was computationally more complex as it aimed at inferring connections using an exponential search space which required the computation of MI over several genes. Relevance networks, on the other hand were computationally less complex as it considered only pair-wise interactions. Various improvements over the relevance networks approach were achieved over the last decade with CLR [35] being the most popular amongst them today.

CLR applies an adaptive background correction step to eliminate false connections and indirect influences. If the gene expression matrix has N rows (i.e. N genes) a $N*N$ pair-wise MI matrix M is maintained; an entry $M(i,j)$ in this matrix designates the MI value between the i^{th} and j^{th} genes. Every entry in this MI matrix is now compared with the background distribution of MI of every possible pair. This is achieved by computing the z-scores over the rows and columns and then finding the

square root of the sum of the squares of these z-scores. Using a threshold over the z-scores, the network is finally obtained.

2.5 Network MDL

The network MDL [38] approach is based on the minimum description length (MDL) principle [39, 40, 41, 42] which states that if multiple theories exist, then the theory with the smallest model length is the best. This approach solves the problem of threshold selection that exists in relevance networks. For every MI value the algorithm generates a network. In order to select the best network, the algorithm computes the description length for every network and chooses the network with minimum description length as the best network. The description length involves the computation of model length and data length. The model length is the amount of memory consumed by the algorithm and the data length is the sum of the joint entropies of every gene and its parents (as entropy is a measure of uncertainty, a network with lower entropy is more robust). Thus, the robustness principle of biological networks is exploited in this approach. A fine tuning parameter is used to control the effects of the model parameter in this method. Unlike in the earlier discussed methods (RN, ARACNE and CLR), in which a network generated with higher threshold is a subset of the network generated using a lower threshold, a network generated using a high fine tuning parameter might not necessarily be a subset of network generated using a low fine tuning in the network MDL algorithm. The model length computation here is not standardized and as a fine tuning parameter is used to control the effect of the model length parameter, this method can be arbitrary.

2.6 Conclusions

This chapter briefly introduced the most popular information theory based algorithms along with their advantages and disadvantages.

CHAPTER 3

PREDICTIVE MINIMUM DESCRIPTION LENGTH PRINCIPLE-BASED APPROACH

The robustness property of biological networks can be best exploited by using the minimum description length principle. The network MDL principle relied on a fine tuning parameter as discussed in section 2.5. Another issue with this algorithm is that it is purely based on pair wise mutual information and does not employ a method for pruning false edges. This chapter proposes predictive minimum description length based algorithm (PMDL Algorithm) which also uses the conditional mutual information metric to further improve the inference accuracy by pruning the false edges.

3.1 Genetic Network

The network formulation is similar to the one used in [38]. A graph $G(V, E)$ represents a network where V denotes a set of genes and E denotes a set of regulatory relationships between genes. If gene x shares a regulatory relationship with gene, then there exists an edge between x and y (x -> y). Genes can have more than one regulator. The notation $P(x)$ is used to represent a set of genes that share regulatory relationships with gene x. For example, if gene x shares a regulatory relationship with y and z then $P(x) = y, z$. Also every gene is associated with a function $f(x) : P(x)$ which denotes the expression value of gene x determined by the values of genes in $P(x)$.

The gene expression is affected by many environmental factors. Since it is not possible to incorporate all factors, the regulatory functions are assumed to be probabilistic.

Table 1. Conditional Probability

yz:x	**0**	**1**
00	0.6	0.4
01	0.3	0.7
10	0.5	0.5
11	0.8	0.2

Also, the gene expression values are assumed discrete-valued [38], as in all Boolean and PBN based methods and the probabilistic regulation functions are represented as look-up tables. If the expression levels are quantized to q levels and a gene x has n predecessors then the look up table has q^n rows and q columns and every entry in the table corresponds to a conditional probability.

Say we have a gene x which shares regulatory relationship with two other genes y, z and the data is quantized to 2 levels, the look up table is as in Table 1. In this example the entry 0.6 can be inferred as, if genes y and z are lowly expressed then the probability that is also lowly expressed is 0.6. The entries in this look-up table are computed based on the microarray data.

3.2 Predictive Minimum Description Length Principle

The description length of the network MDL algorithm involves calculation of model length and data length. As the length can vary for various models, the network MDL algorithm is in danger of being biased towards the length of the model [41]. The Normalized Maximum Likelihood Model has been implemented in [43] to overcome this issue. Another such model based on universal code length is the PMDL principle [26] which was chosen here as it suits time series data [44, 45].

The predictive minimum description length for a model L_D is given as [40]:

$$L_D = -\sum_{t=0}^{m-1} log(p(X_{t+1}|X_t)) \tag{3.1}$$

Where, $p(X_{t+1}|X_t)$ is the conditional probability or density function and m is the number of time points in the data set . We calculate the description length as data length given in [38].

3.2.1 Data Length Computation

The description length of the PMDL algorithm is equivalent to the data length proposed in [40]. A gene can take any value when transformed from one time point to another due to the probabilistic nature of the network. The network is associated with Markov chain which is used to model state transitions. These states are represented as n-gene expression vectors $X_t = (x_{1,t}, ..., x_{n,t})^T$ and the transition probability $p(X_{t+1}|X_t)$ can be derived as follows:

$$p(X_{t+1}|X_t) = \prod_{i=1}^{n} p(x_{i,t+1}|P_t(x_i)) \tag{3.2}$$

The probability $p(x_{i,t+1}|P_t(x_i))$ can be obtained from the look-up table associated with the vertex x_i and is assumed to be time invariant. It is estimated as follows:

$$p(x_{i,t+1} = j|P_t(x_i) = \frac{1}{m-1} \sum_{t=1}^{m-1} 1_{\{j\}} x_{i,t+1}|P_t(x_i)) \tag{3.3}$$

Each state transition brings new information that is measured by the conditional entropy:

$$H(X_{t+1}|X_t) = -\log(p(X_{t+1}|X_t)) \tag{3.4}$$

The total entropy for given m time-series sample points $(X_1, ..., X_m)$, is given by

$$L_D = H(X_1) + \sum_{j=1}^{m-1} H(X_{j+1}|X_j) \tag{3.5}$$

As $H(X_1)$ is same for all models it is removed thus the description length is

$$L_D = \sum_{j=1}^{m-1} H(X_{j+1}|X_j) \tag{3.6}$$

As the MDL principle selects the model with smallest length as best model, in this case the best model is the one with lowest uncertainty (entropy). Thus, this approach exploits the robustness property of the biological networks.

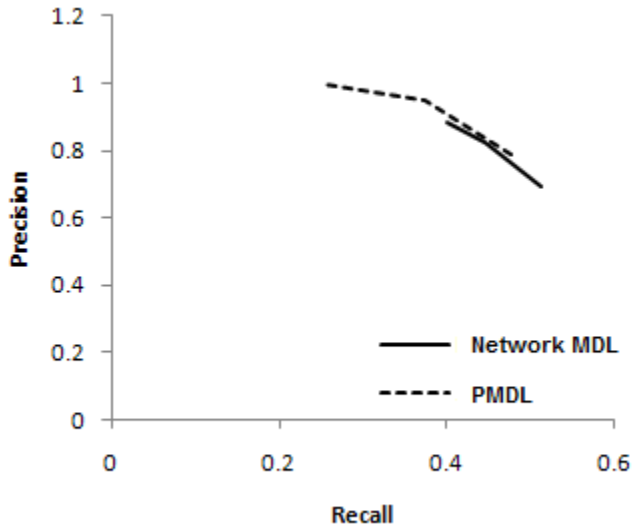
3.3 PMDL Algorithm

Initially the time series data is pre-processed (filling missing values and quantizing the data). An example of pre-processing methodologies implemented on a benchmark data set is discussed in Section 3.6. After pre-processing, the MI matrix $M_{n\times n}$ is evaluated. A connectivity matrix $C_{n\times n}$ is maintained which has two entries: 0 and 1. An entry of 0 indicates that no regulatory relationship exists between genes, but an entry of 1 at $C_{i\times j}$ indicates that gene i regulates j.

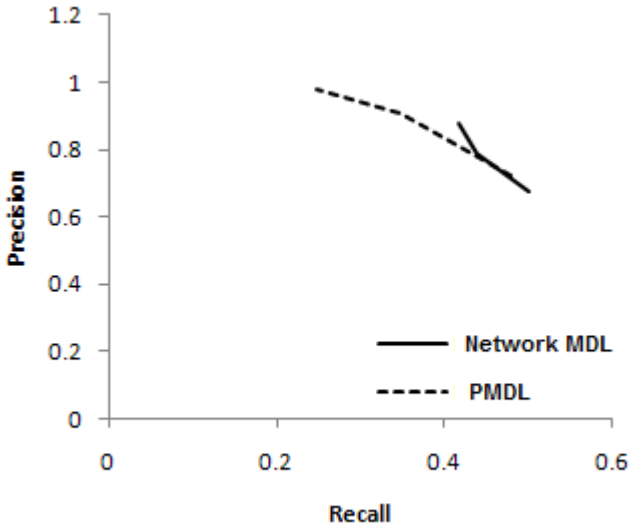
The pseudo-code of PMDL algorithm is shown in Algorithm 1. From lines 5 to 18 of PMDL algorithm, every value of the MI matrix is used as a threshold and a network is obtained. The conditional probabilities and the description lengths for each of these models are evaluated using equations 3.3 and 3.5 respectively. Then at line 19, the MI which was used to obtain the network with the shortest description length is then used as the threshold to obtain the initial network. Lines 20 to 31 implements false edge pruning technique using CMI metric. For every pair of genes with valid regulatory connection in the initial network, CMI of these genes with every other

```
Input time series data
Preprocess data
Initialize M_{n×n}, C_{n×n} and P(x_j) ⇐ φ
Calculate the cross-time mutual info between genes and fill M_{nxn}
for i = 1 to n do
    for j = 1 to n do
        δ ← M_{i×j}
        for k = 1 to n do
            for l = 1 to n do
                if M_{k×l} ≥ δ then
                    C_{k×l} = 1, P(x_l) ⇐ P(x_l) ∪ x_k
                end if
            end for
        end for
        Compute probabilities using Equation 3.3
        Compute the description length L_D using Equation 3.6
    end for
end for
Select the MI of model having least L_D as the MI threshold δ
for i = 1 to n do
    for j = 1 to n do
        if C_{i×j==1} then
            for k = 1 to n and k != i, j do
                if CMI_{i,j,k} < Th then
                    C_{i×j} ⇐ 0
                    break
                end if
            end for
        end if
    end for
end for
```

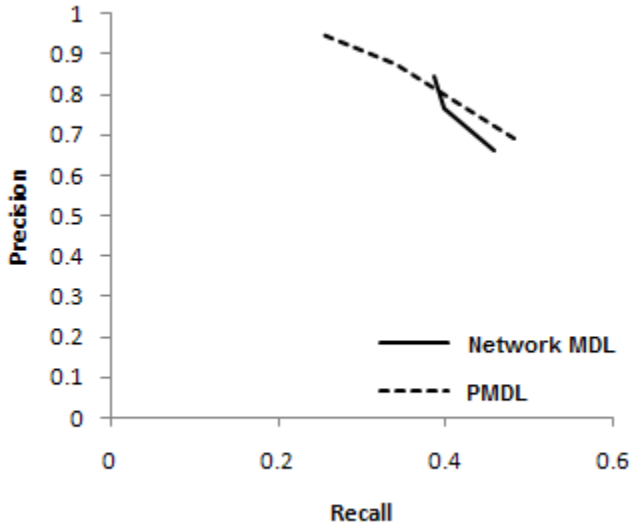
Algorithm 1: Pseudo-code for PMDL Algorithm



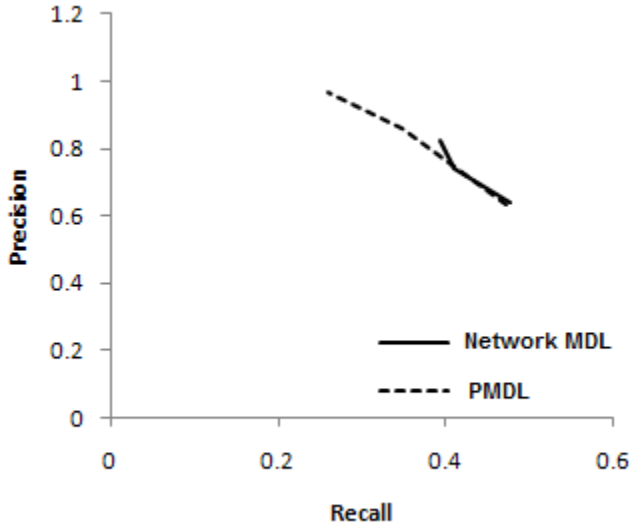
(a) 20 gene network



(b) 30 gene network



(c) 40 gene network



(d) 50 gene network

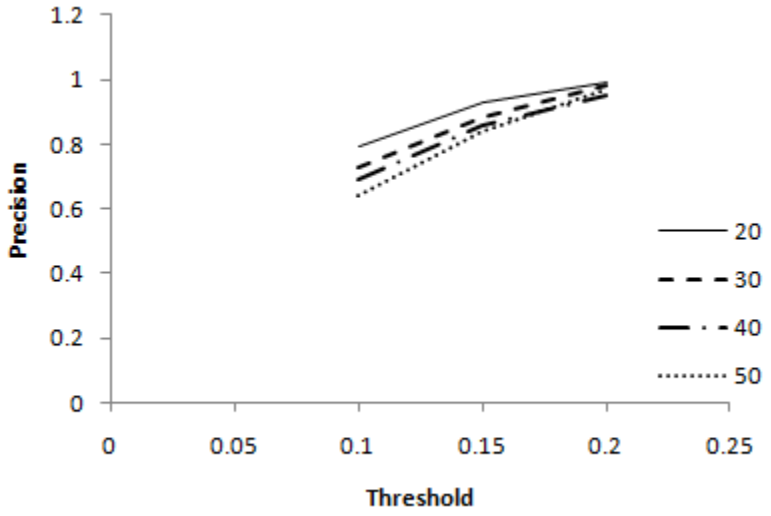
Fig. 3. Precision recall sensitivity analysis of threshold selection for PMDL and network MDL algorithms

gene is evaluated and if the value is below a user specified threshold, the connection is deleted. The resulting network after the false edge pruning technique is the final network.

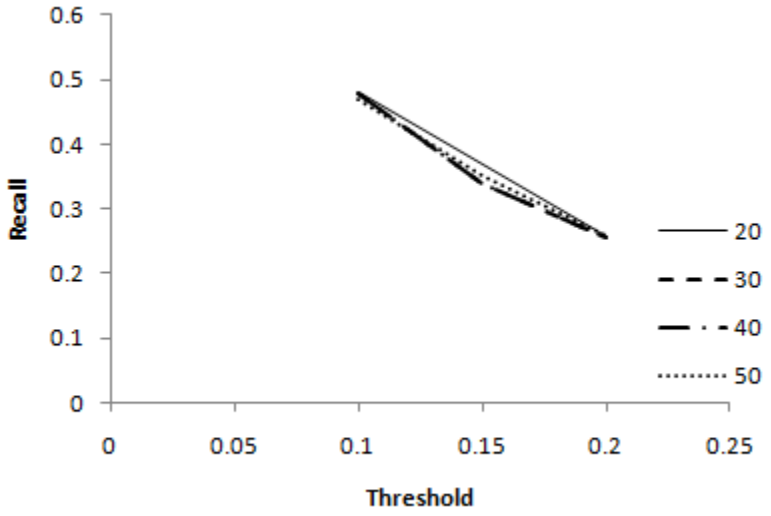
3.4 Simulation on Random Synthetic Networks

The PMDL algorithm is compared with network MDL algorithm on synthetic random networks. For a specific size of the network, both algorithms were run for different threshold values 30 times each and the average of precision and recall were calculated. The algorithms were run for 20, 30, 40 and 50 numbers of genes. The precision recall curve for each of these networks with different threshold values are given in Figure 3.

Zhao et al. [38] reported 0.2 to 0.4 as the suitable tuning parameter value range for the model length computation of the network MDL approach; hence, 0.2, 0.3 and 0.4 were used as the fine tuning parameters to build the networks. Based on simulations of PMDL algorithm, it was observed that the threshold for CMI worked best for values in the range 0.1 to 0.2. Thus, threshold values 0.1, 0.15 and 0.2 were used to build the networks. In Figure 3, it is observed that the precision of the proposed algorithm is higher but recall is lower in most of the cases as compared to network MDL. The number of false negatives is fewer in the proposed algorithm and as most biologists are interested in true positives, our proposed PMDL algorithm is preferred over the network MDL algorithm. Although the improvements in inference accuracy achieved by the PMDL algorithm were at best marginal over network MDL for random synthetic networks, we did observe much better performance of PMDL algorithm over time-series data-sets as reported next.



(a) Precision sensitivity



(b) Recall sensitivity

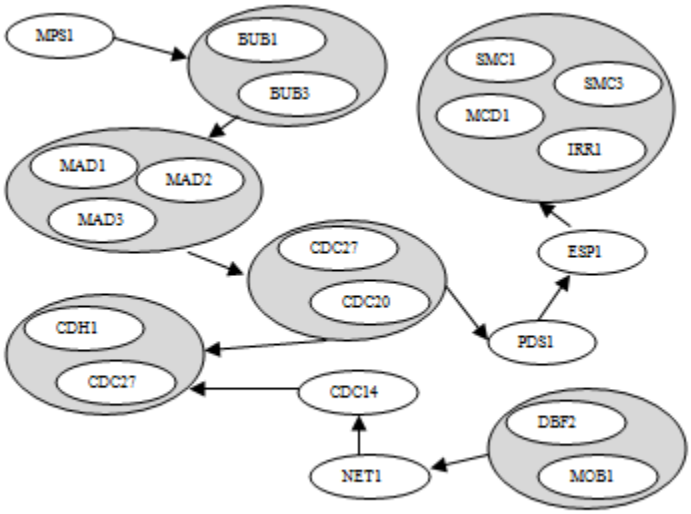
Fig. 4. Sensitivity analysis of conditional mutual information metric for the PMDL algorithm

3.5 Threshold Sensitivity

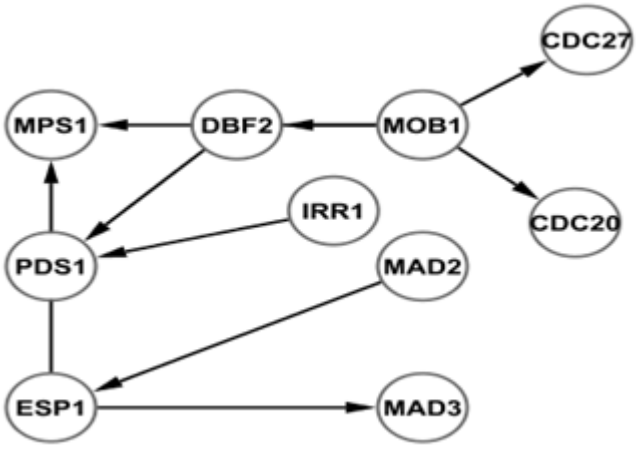
The performance of the PMDL algorithm based on different values of the user specified conditional mutual information threshold over synthetic networks was studied. For threshold values of 0.15 and 0.2, a high precision (over 90% in most cases) was observed but the recall for these thresholds was low (from 25% to 30%) as compared to a threshold value of 0.1 which had a fair recall (over 47%) and good precision (63% to 79%) performance. Figure 4 indicates that as the threshold value is increased, precision increases while the recall decreases. The simulation experiments show that 0.1 is the optimal threshold value.

3.6 Performance on Saccharomyces Cerevisiae Data Set

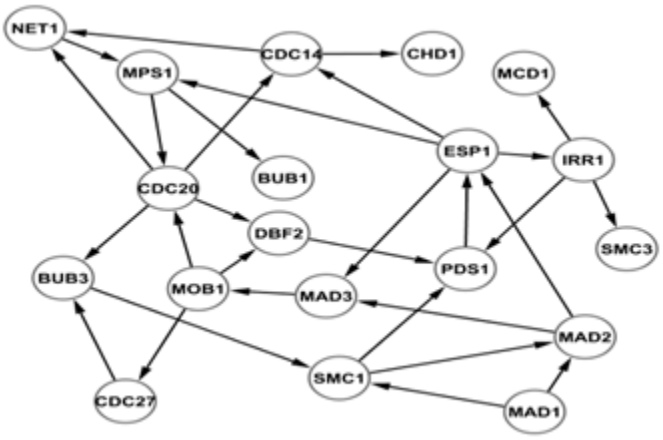
The time series cDNA microarray data from [46] was used to infer gene regulatory networks. The Spellman experiment was chosen because it provides a comprehensive series of gene expression datasets for Yeast cell cycle. In the experiment, four time series expression datasets were generated using four different cell synchronization methods: cdc15, cdc28, alpha-factor and elutriation with 24, 17, 18 and 14 time points respectively. The alpha-factor dataset contained more time points than the cdc28 whereas the elutriation dataset had fewer missing values than the cdc15 dataset. Therefore, the alpha-factor dataset to infer gene regulatory networks was selected.
Data was preprocessed as in [38]. Initially the data is quantized to 0 or 1. In order to quantize, the expression values of every gene they are sorted in ascending order. The first and last values of the sorted list are discarded as outliers. Then the upper 50% of the expression values are set to 1 and the lower 50% are set to 0. Any missing time points are set to the mean of their respective neighbours. If the missing time point is the first or the last one, it is set to the nearest time point value.



(a) Biological network



(b) Network inferred using network MDL



(c) Network inferred using PMDL

Fig. 5. Performance analysis of network MDL and PMDL algorithms on network derived from *Saccharomyces cerevisiae*

The true biological network used for comparison analysis was derived from the Yeast cell cycle pathway [47, 48, 49]. A total of six networks were reverse engineered, of which three were inferred using the proposed PMDL algorithm (each using CMI thresholds 0.1, 0.15 and 0.2) and the remaining three using the network MDL algorithm (each using model length tuning parameters 0.2, 0.3 and 0.4). The best network out of the three in each case was used for comparison. The true network and the networks inferred using PMDL algorithm and the network MDL algorithm are shown in Figure 5. Of the 30 edges inferred by the proposed PMDL algorithm, nine were correctly inferred edges. The network MDL algorithm inferred a total a nine edges, of which only one was correctly inferred edge. Clearly the precision and recall parameters of PMDL algorithm are higher than the network MDL algorithm. Hence, the results favor the proposed PMDL algorithm.

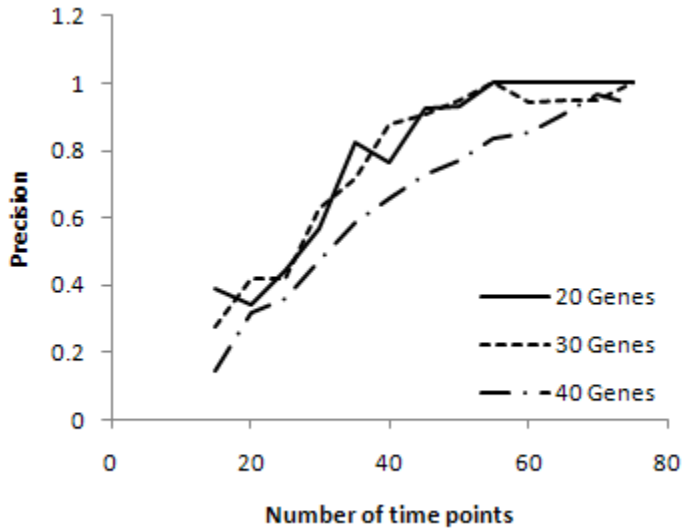
3.7 Effect of Data Size on Regulatory Network Inference

It was observed that network inference varied with different data size (number of microarray experiments) which motivated an empirical analysis on the effects of data size on the inference approaches as reported next. A network and a data set with 75 time points were generated using GeneNetWeaver tool [50, 51, 52]. Both network MDL and PMDL algorithms were run 13 times starting with the first 15 time points of data. An increment of five time points was made for every subsequent run. For every run, the values of precision and recall were computed. In Network MDL algorithm, the model length fine tuning parameter was set to 0.2 and in PMDL algorithm, the conditional mutual information threshold was set to 0.1. The precision and recall plots for PMDL and network MDL algorithms are shown in Figure 6.

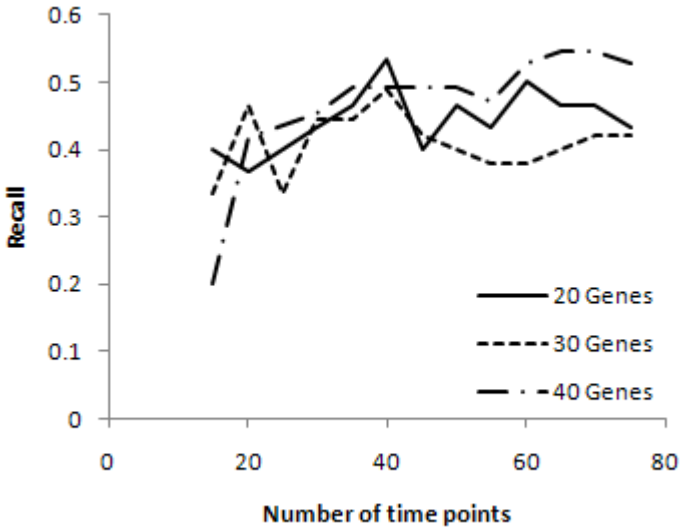
The precision analysis for the PMDL algorithm, as shown in Figure 6(a), reveals that the precision in 20-gene and 30-gene networks, saturated beyond 55 time points of data. For the 40-gene network, precision increased till 70 time points and reduced for the 75th time point.

The recall analysis for the PMDL algorithm, as shown in Figure 6(b), reveals that the recall in the 20-gene network increased till 40 time points beyond which it started to fluctuate. In the 30-gene network the recall fluctuated beyond 20 time points data but saturated beyond 50 time points data. In the 40 gene network, recall saturated between 35 to 55 time points and slightly increased beyond 55 time points data.

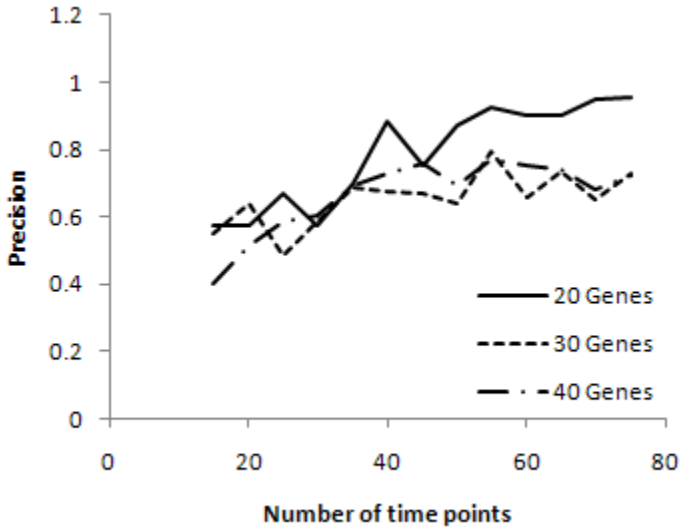
The precision analysis for the network MDL algorithm, as shown in Figure 6(c), reveals that the precision in 20-gene network saturated beyond 60 time points data and in 30-gene network the precision saturated between 35 to 55 time points data and fluctuated beyond 55 time points data. In the 40-gene network, precision increased till 40 time points data and saturated beyond that.



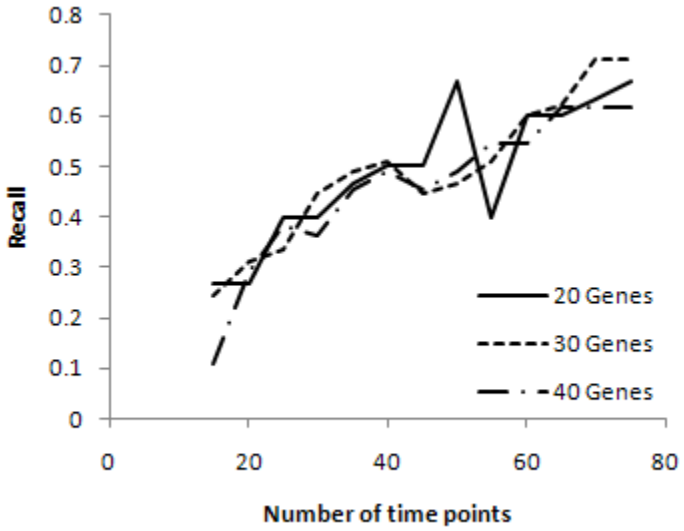
(a) PMDL Precision



(b) PMDL Recall



(c) Network MDL Precision



(d) Network MDL Recall

Fig. 6. Sensitivity analysis of data size of network MDL and PMDL algorithms

The recall analysis for the PMDL algorithm, as shown in figure 6(d), reveals that the recall in 20-gene network increased till 40 time points and beyond that the recall fluctuated. In the 30-gene network the recall increased till 40 time points data and fluctuated beyond that. In the 40 gene network recall increased till 25 time points data and fluctuated beyond that.

For further analysis the recall/precision ratio (Figure 7) was considered. Saturation beyond 55 time points was observed for PMDL for recall/precision ratio. Saturation was also found from 30 to 45 time points for Network MDL algorithm and beyond that the ratio either increased or decreased. Curve fits of Figure 7 are shown in Figure 8.

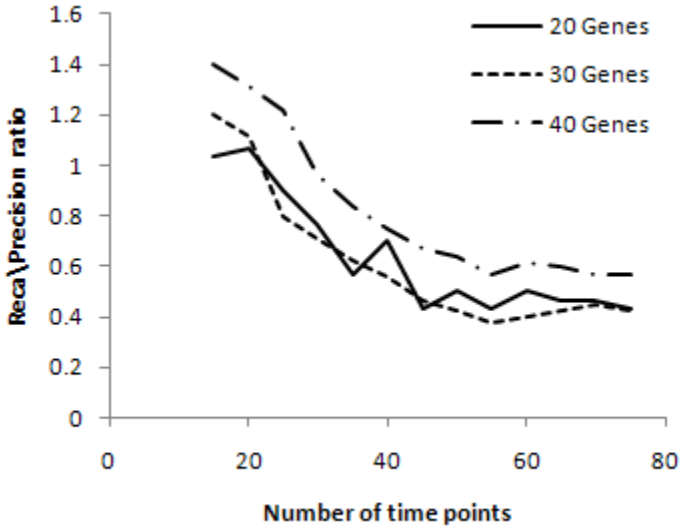
3.8 Why Synthetic Data?

It was imperative to use synthetic data over time series micro-array experimental data in this analysis phase due to the following reasons:

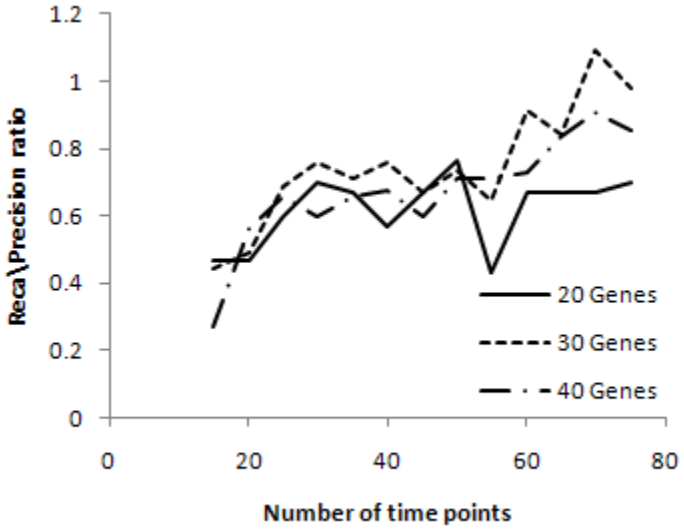
1. Very few experimental data sets have equal time intervals between experiments and also the data size is generally limited to around 20 time points. In our *in-silico* runs, equal time intervals between each time point was desired in order to perform proper sensitivity analysis on the data size.

2. Also, the saturation in inference accuracy generally requires a larger data size ($>$ 30 time points as shown later) and it would have been difficult to identify the role of MI in bringing about this theoretical limit on the accuracy of information theoretic schemes with a smaller biological data set (of ˜20 time points).

It should be noted that the Genenetweaver software [50, 51, 52] derives the *in-silico* GRNs from the prior knowledge database of Yeast (*Saccharomyces cerevisiae*) which

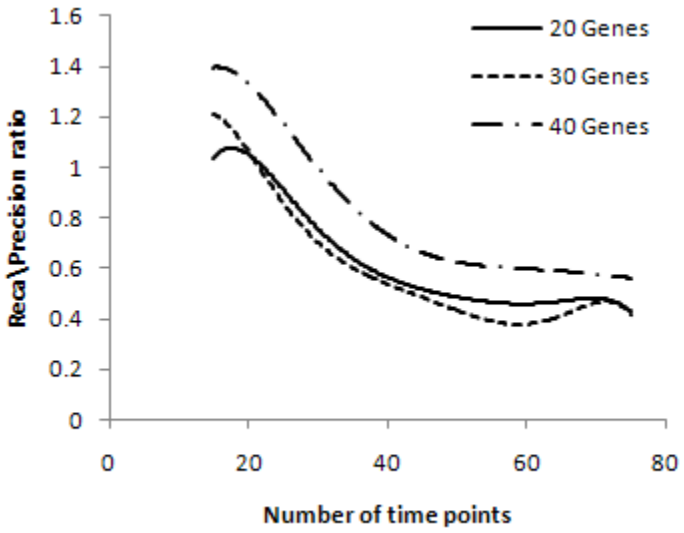


(a) PMDL algorithm



(b) Network MDL algorithm

Fig. 7. Sensitivity analysis of data size using recall precision ratio metric of network MDL and PMDL algorithms



(a) PMDL algorithm

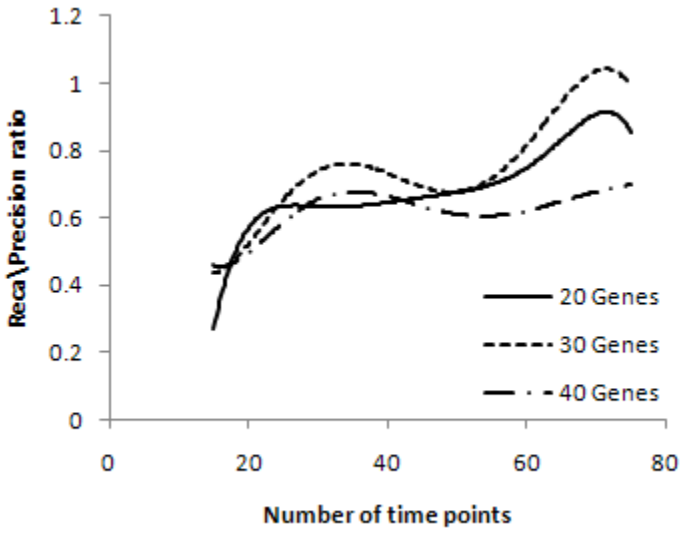


Fig. 8. Recall/precision ratio curve fit graphs

contains 4441 genes and 12873 interactions. Thus, in order to create a sample GRN with 10 nodes, GeneNetWeaver clusters the Yeast transcriptomic network into modules and selects the module having number of genes closest to the given input to create the *in-silico* network. Each such module maps to a particular biological function and this strategy essentially ensures that there is minimum cross-talk of these set of genes with the others in the Yeast network. This results in a higher efficacy of the inference algorithms that use them.

3.9 Time and Space Complexities

The performance of the PMDL algorithm depends on three factors. The number of genes, number of time points and most importantly, the number of parents (regulators) inferred for each gene by the algorithm. Time and space complexities of PMDL algorithm were computed and analyzed to observe the role of the above mentioned factors in its performance. Line 4 of PMDL Algorithm 1 iterates n^2m times, where n is number of genes and m is the number of time points; between lines 5 and 18 the algorithm iterates n^4 times. The algorithm iterates n^3m times for lines 15 and 16. Finally, from lines 20 to 31 the algorithm iterates n^3 times. Thus, the time complexity of the algorithm is $O(n^4 + n^3)$.

When it comes to space complexity, the conditional probability tables play a major role. If a gene has n parents, then the conditional probability tables take 2^n units of space. Thus, the amount of memory needed by the algorithm depends on the number of parents inferred by the network. As the space complexity grows exponentially based on the number of parents, it is possible that the algorithm may run out of memory for a data set with as few as 50 genes but run for as little as five minutes for a data set with several hundred genes. There are two ways to overcome

this limitation:

1. Restrict the number of parents and

2. Take the next smallest description length, instead of using the smallest one.

The first approach guarantees results when the number of parents is restricted to small values (between three to six) but this may lower the accuracy of the results. The second approach may take more time to run but as the number of parents is not restricted, the accuracy of the algorithm is not affected.

3.10 Conclusions

The simulation results show that the PMDL algorithm is fair in determining MI threshold. A problem with the PMDL algorithm is determining CMI threshold. Sensitivity analysis of the threshold identified that the value of 0.1 is optimal for most synthetic networks. This was also true in the case of reverse engineering of gene regulatory networks from biological time series DNA microarray data. In synthetic network simulations, the proposed PMDL algorithm produced fewer false edges compared to the network MDL. However, it resulted in a larger number of missing edges. Currently, the space complexity of the algorithm increases exponentially based on the number of parents inferred for genes. Finally, the effects of different data size on the algorithms were studied. Based on recall-precision ratio metric it was observed that the performance of PMDL algorithm saturates after 50^{th} time point, whereas saturation for network MDL was observed between 30 to 45 time points and beyond 45^{th} time point, the performance fluctuated (better than saturation point). The work in this chapter was published in [37, 53, 54]

CHAPTER 4

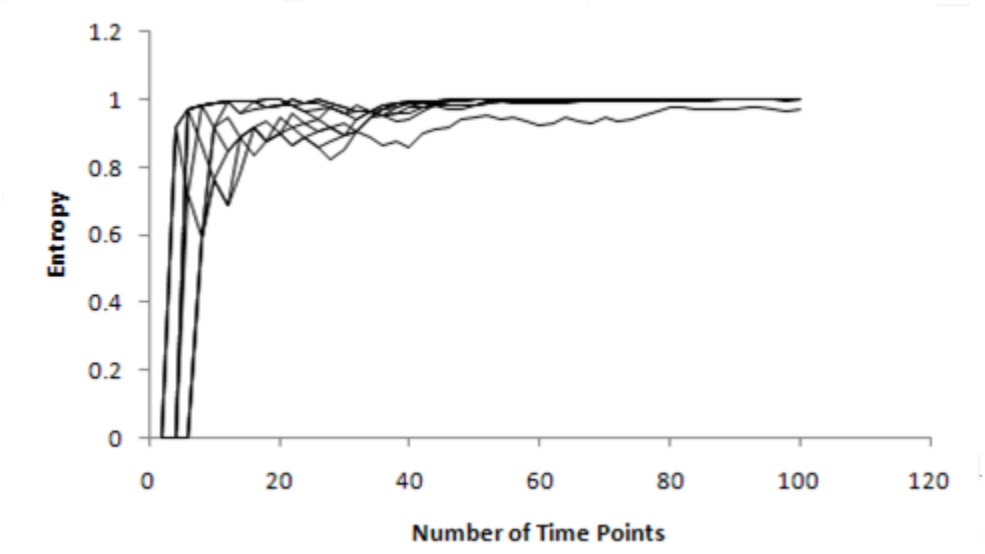
TIME LAGGED INFORMATION THEORETIC APPROACHES

The performance saturation of PMDL and network MDL prompted further study of the information theoretic metric. This also motivated in developing newer information theoretic metrics: time-lagged MI and time-lagged CMI. In this Chapter, the reasons for performance saturation of PMDL and network MDL algorithms are also discussed.

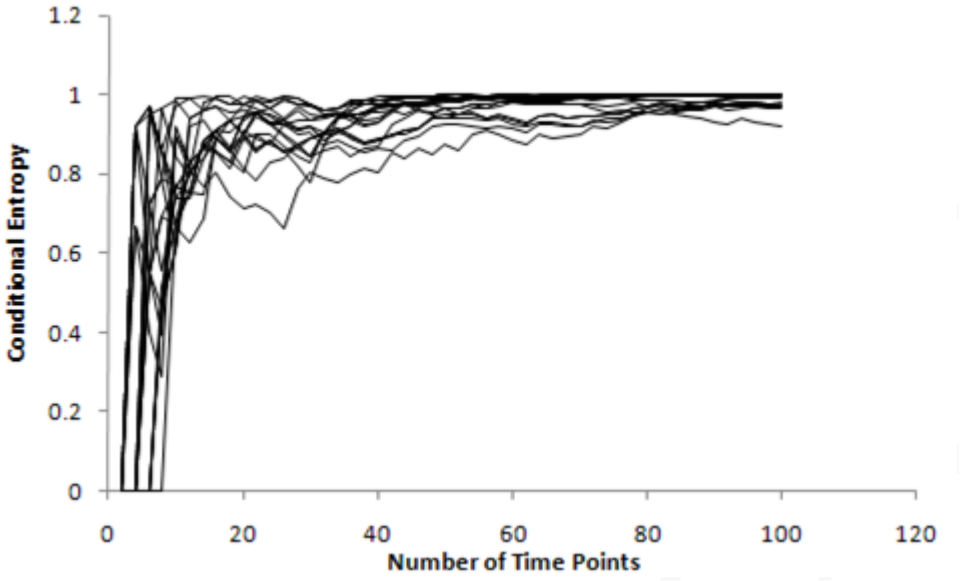
4.1 Why do Information Theory Based Models Saturate?

Visual analysis of basic information theoretic metrics for different data sizes was performed to explore the reasons for the saturation of the accuracy of the corresponding methods. For this analysis, *in-silico* data was created using the GeneNetWeaver tool.

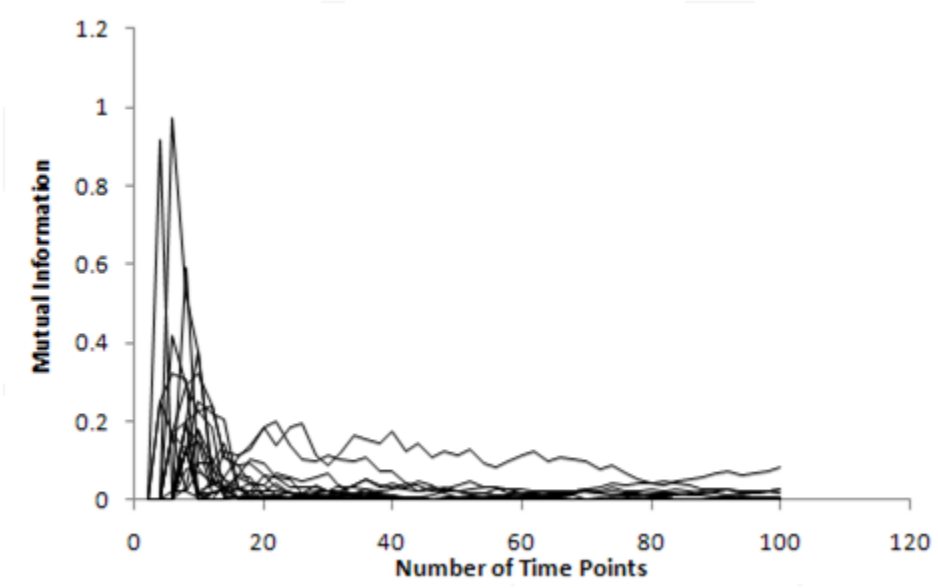
A five gene network and data with 100 time points was generated. The *in-silico* data was quantized to two levels after which the information theoretic quantities were computed from two to 100 time points data. Figure 9 shows the plots for entropy, conditional entropy and mutual information metrics. While the entropy of all genes in the network across 100 time points was computed, the conditional entropy and MI was computed for all possible pair of genes. It was observed that with more data (and correspondingly with more time points) both entropy and conditional entropies increased in the network (tending to unity) resulting in very low values for MI (which tends to zero). A higher MI value ideally signifies the possibility of an edge between the two genes involved. Hence, as this plot only considered the true edges, they should have a higher MI; as more number of time-points are considered (i.e., as



(a) Entropy



(b) Conditional Mutual Information



(c) Mutual Information

Fig. 9. Data size sensitivity analysis for information theoretic metrics

the data size increases), the MI values should ideally get more accurate and start increasing. However, we observe just the opposite in the plots. As the MI tends to zero, this actually demonstrates that the true edge should not be present as a larger data size is used, which point to the fact that MI by itself (computed over time-series datasets) may not be the best metric to signify the possibility of an edge between two genes.

These plots conclude that the saturation in PMDL and network MDL algorithms was due to the saturation in mutual information quantity which goes close to zero even though the entropy increased in the network. This would conceptually mean that there is room to improve the inference accuracy (due to high entropy), yet the mutual information metric will not be able to point in the right direction. Other information theoretic algorithms, like REVEAL, use the ratio of MI and entropy to infer the network. The plots conclude that the saturation in the methods was due

to the saturation in the mutual information quantity which goes close to zero even though the entropy increases in the network. This would conceptually mean that there is room to improve on the inference accuracy (due to high entropy), yet the mutual information metric will not be able to point us to the right direction. Other information theoretic algorithms, like REVEAL use the ratio of MI and entropy to infer the network for this purpose which supposedly gives good performance.

However, from the entropy and mutual information curves in Figure 9, it can be deduced that the ratio of mutual information and entropy will also saturate for large data sizes. Hence, this ratio might also not be the right metric to achieve better accuracy by making use of more time points data. The recently proposed Directed Mutual Information metric [55] might be a better metric than the conventional MI metric. It is imperative to identify the right metric for the research community to decide which class of GRN inference algorithms can work best with time-series data and also understand the ideal data size for them.

The saturation in MI due to increasing number of time points would suggest that MI should not be computed for the entire range of time points of micro-array data available from the experiments. GRNs are inherently time varying, and hence the pair-wise MI between any two genes needs to be computed over the time range where the first gene will have *substantial regulatory effect* on the other one. This can be best approximated by estimating the regulatory time-lags between each gene pair, and subsequently computing the MI between them for this particular time range. The time lag computation concept was initially proposed in [12] to predict potential regulators in their DBN based scheme.

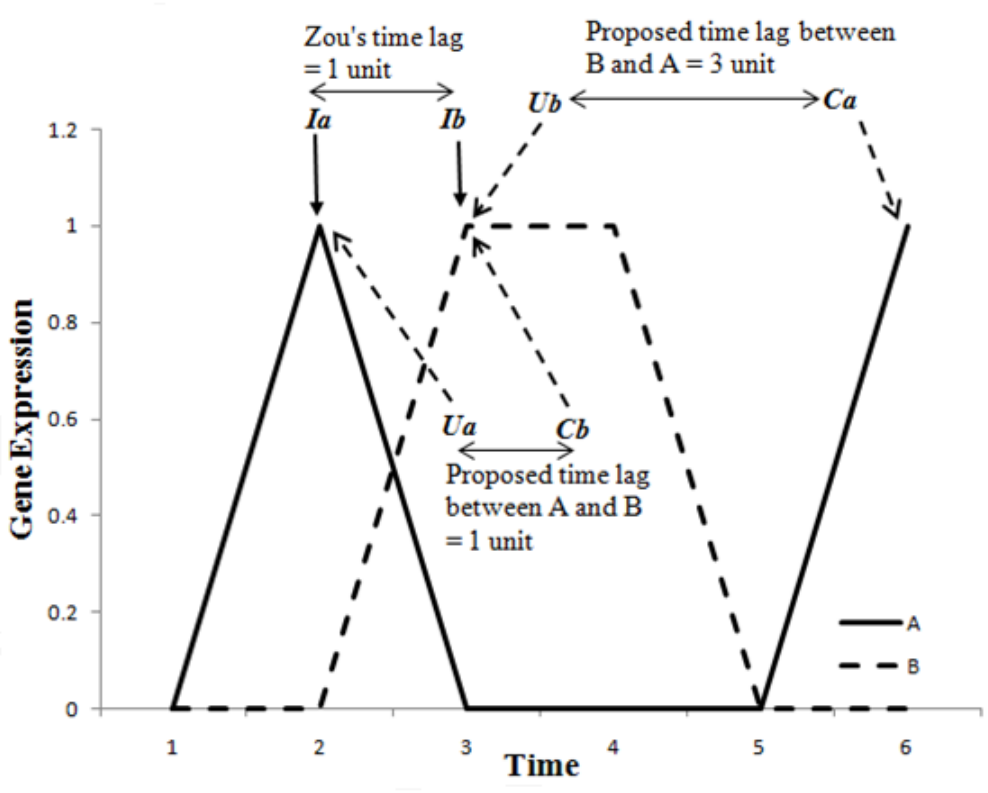


Fig. 10. Example for different time lag computation methods

4.2 Time Lags

The concept of time lags was first introduced by Zou et.al [12], where they proposed that the time difference between the initial expression change of a potential regulator (parent) and its target gene represents a biologically relevant time lag. Here, potential regulators are those set of genes whose initial expression change happened before the target gene. Also, initial expression change is up or down-regulation (ON or OFF) of genes.

The motivation lies in changing the MI metric to incorporate the effect of time lags. While implementing Zou et.als method of calculating time lags, the problem of negative lags is encountered. For every pair of gene, when the initial expression

change is not at the same time point, one of the two time lags turns out to be negative. Figure 10 illustrates this problem. In the figure I_a and I_b indicate the initial change in expression of gene A and gene B at time points 2 and 3 respectively. As per Zou et.al [12], gene A is potential parent whereas gene B is not a potential parent of gene A. Time lag between A and B is $I_a - I_b = 3 - 2 = 1$. Time lag between B and A is $I_b - I_a = 2 - 3 = -1$. Thus, the time lag between B and A is a negative time lag. As per [12], if a pair of genes have negative time lage, then no regulatory interaction exists from gene whose expression change occurred later to the gene whose expression change occurred earlier.

It is important to consider such time lags, both in forward and backward directions (i.e., from A to B and vice-versa) as this models the loops between 2 genes (i.e., A->B and B->A connections). Hence, Zou et.als [12] time lag computation scheme needs to change to handle such cases. It can be argued that a gene can regulate another gene only when it is up-regulated (ON). Based on the above discussion, ***a time lag is proposed as the difference between initial up-regulation of first gene and the initial expression change of the second gene after the up-regulation of first gene***. This solves the issue of negative time lags besides being biologically more relevant as compared to the existing method of calculating time lags.

Figure 10 also illustrates the proposed time lag computation based on our approach. In the figure, U_a and U_b indicate the initial up-regulation of genes A and B respectively at time points two and three respectively. C_a and C_b indicate that time points six and three are the time points where the expression values of genes A andB changed after the initial up-regulation of genes B and A. Time lag between A and B is calculated as $C_b - U_a$ and time lag between B and A is calculated as $C_a - U_b$ respectively. In this example, time lag between A and B is one and time lag between B and A is three.

4.3 Time Lagged Mutual Information (TLMI) and Time Lagged Conditional Mutual Information (TLCMI)

While computing the time-lagged information theoretic quantities, the computation of entropy and/or joint entropy over all time point data is not required. If a time lag t is computed between two genes A and B then the last t time points data of A and first t time points of B are removed to obtain a reduced data set (of length m-t each) for A and B respectively. Computing the MI over these reduced sequences gives TLMI. TLMI is not a symmetric quantity though, i.e., $TLMI(A, B)! = TLMI(B, A)$ Considering a time lag, t, between A and B, TLCMI$(A; B|C)$ can be computed by deleting the last t time points data of A and C and first t time points data of B . Computing the CMI on this reduced data set gives TLCMI. Figure 11 demonstrates computation of the information theoretic quantities MI, TLMI, and TLCMI.

4.4 Performance of TLMI Based Network MDL Implementation

4.4.1 Data Set and Pre-processing

The time series DNA microarray data from Spellman et al [46] was used to infer gene regulatory networks using the time-lagged version of network MDL algorithm. The same preprocessing steps as in [38] were used. Initially, the data is quantized to 0 or 1. In order to quantize, the expression values of every gene are sorted in ascending order. The first and last values of the sorted list are discarded as outliers. Then the upper 50% of the expression values are set to 1 and the lower 50% is set to 0. If the missing time point is the first or the last one, it is set to the nearest time point value.

Three separate biological networks used for comparison purposes were derived from the Yeast cell cycle pathway [47, 48, 49]. The fine tuning parameter required by the network MDL based algorithms is set to 0.1.

Gene \ Time	1	2	3	4	5	6
A	1	1	0	0	0	1
B	0	0	1	1	0	0

$H(A) = -0.5 \log(0.5) - 0.5 \log(0.5) = 1$
$H(B) = -0.5 \log(0.5) - 0.5 \log(0.5) = 1$
$H(A,B) = -0.5 \log(0.5) - 0 - 0 - 0.5 \log(0.5) = 1$
$I(A,B) = 1 + 1 - 1 = 1$

MI Computation

Gene \ Time	1	2	3	4	5	6
A	1	1	0	0	0	1
B	0	0	1	1	0	0

$H'(X) = -0.5 \log(0.5) - 0.5 \log(0.5)$
$H'(Y) = -0.5 \log(0.5) - 0.5 \log(0.5)$
$H'(X,Y) = -0.5 \log(0.5) - 0 - 0 - 0.5 \log(0.5)$
$TMLI(A,B) = 1 + 1 - 1 = 1$

Actual time series data is up to 6 time points, but based on the time lags the MI Computation involves only 4 (*6-2=4 i.e. the difference between total number of time points and time lag*) time points (as shaded). Thus TLMI is computed as follows:

Gene \ Time	1	2	3	4	5	6
A	1	1	0	0	0	1
B	0	0	1	1	0	0
C	0	1	1	0	0	1

$H'(A,C) = -(1/4)\log(1/4) - (1/4)\log(1/4) - (1/4)\log(1/4) - (1/4)\log(1/4) = 2.0$
$H'(B,C) = -(1/4)\log(1/4) - (1/4)\log(1/4) - (1/4)\log(1/4) - (1/4)\log(1/4) = 2.0$
$H'(C) = -(2/4)\log(2/4) - (2/4)\log(2/4) = 1.0$
$H'(A,B,C) = -(1/4)\log(1/4) - (1/6)\log(1/6) - 0 - 0 - 0 - 0 - (1/4)\log(1/4) - (1/4)\log(1/4) = 2.0$
$TLCMI(A;B|C) = 2.0 + 2.0 - 1.0 - 2.0 = 1.0$

TLCMI Computation

Fig. 11. Example for (A) MI (B) CMI, and (C) TLCMI computations

On a network with 10 genes, the TLMI implementation of the network MDL algorithm inferred 12 edges of which three were correct; whereas MI implementation of network MDL algorithm inferred seven edges of which one was correct. As the network has 12 true edges the precision and recall for the TLMI based network MDL based implementation are both 0.25. Whereas, the precision and recall for MI based implementation are 0.14 and 0.08 respectively. As both the precision and recall values of TLMI based implementation are higher than the MI based implementation, the initial results favor our approach. Figure 12 shows the true network and networks inferred using TLMI based network MDL and the original network MDL algorithms.

The same process for two other biological networks with 11 and nine genes from the Yeast cell cycle was repeated. The corresponding results are shown in Figure 13 and Figure 14 respectively. For the 11 gene network, MDL inferred a total of 19 edges of which seven were correct whereas TLMI based network MDL approach inferred a total of 17 edges of which eight were correct, resulting in an improvement in both precision and recall. For the 9 gene network, the network MDL algorithm inferred a total of nine edges of which three were correct whereas the proposed TLMI based MDL approach inferred a total of six edges of which three were correct. In this case, while recall for both methods is the same, the precision of the time-lagged network MDL algorithm is better.

4.5 Performance of TLMI and TLCMI Based PMDL Implementation

The TLMI and TLCMI metrics were used in the PMDL based algorithm. A network with 14 genes was derived from the Yeast cell cycle [47, 48, 49] and Spellman′s data [46] was used for performance evaluation. The new PMDL implementation inferred 27 edges of which five were correct while the earlier PMDL algorithm inferred 26 edges of which five were correct (the true and inferred biological networks from

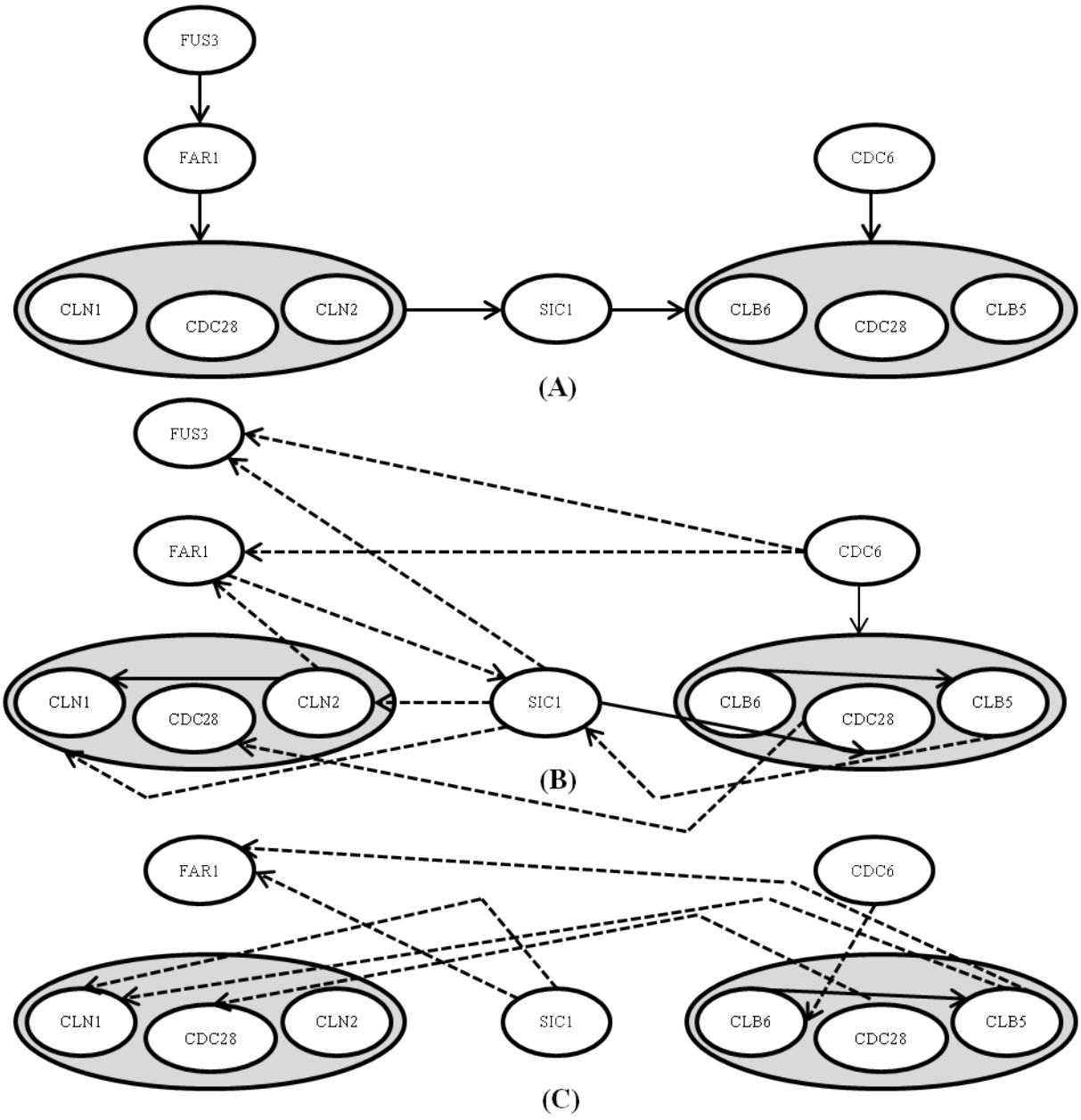
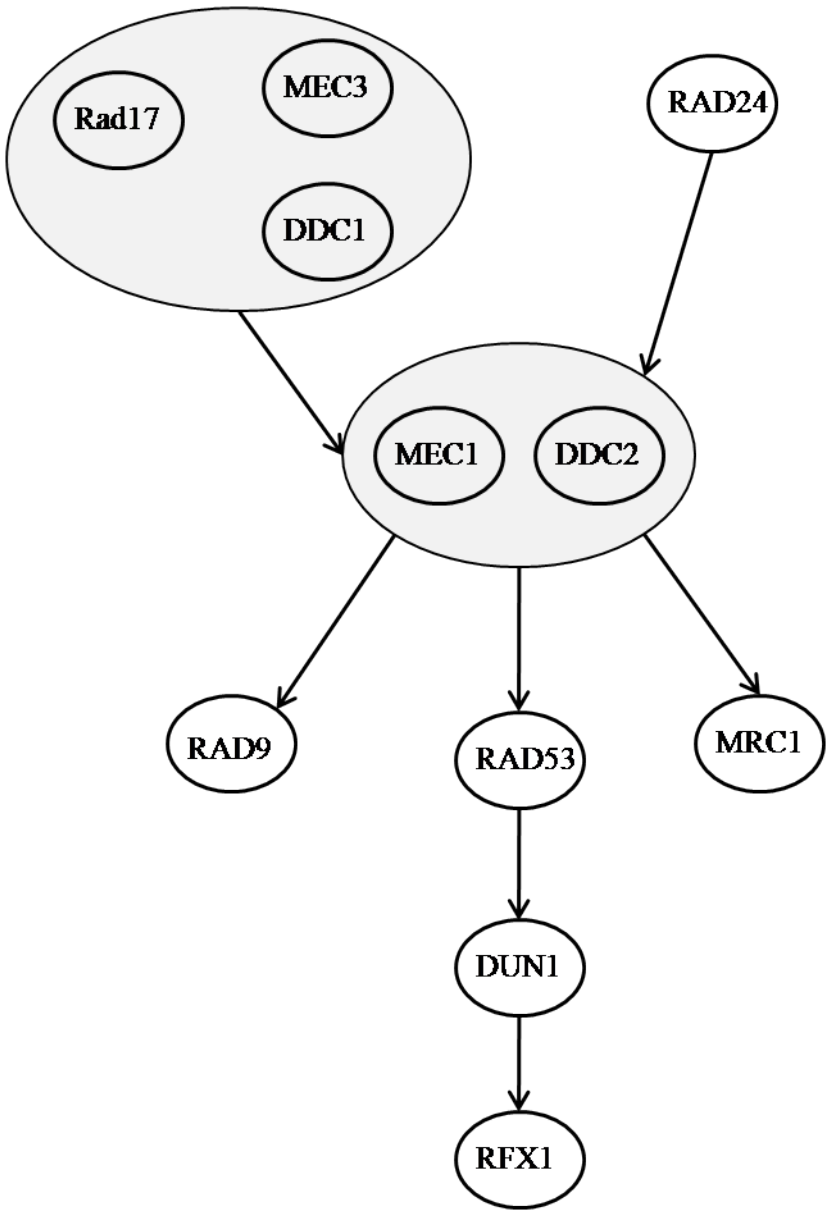
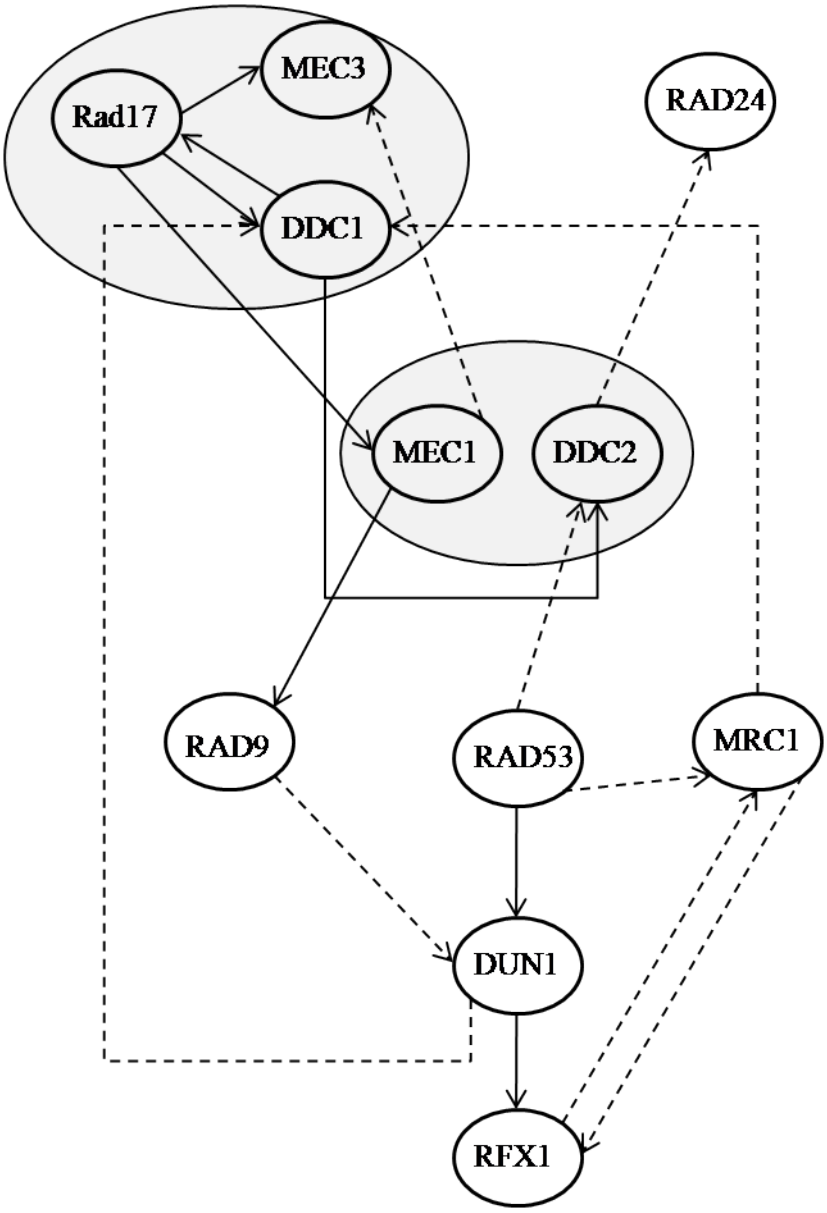


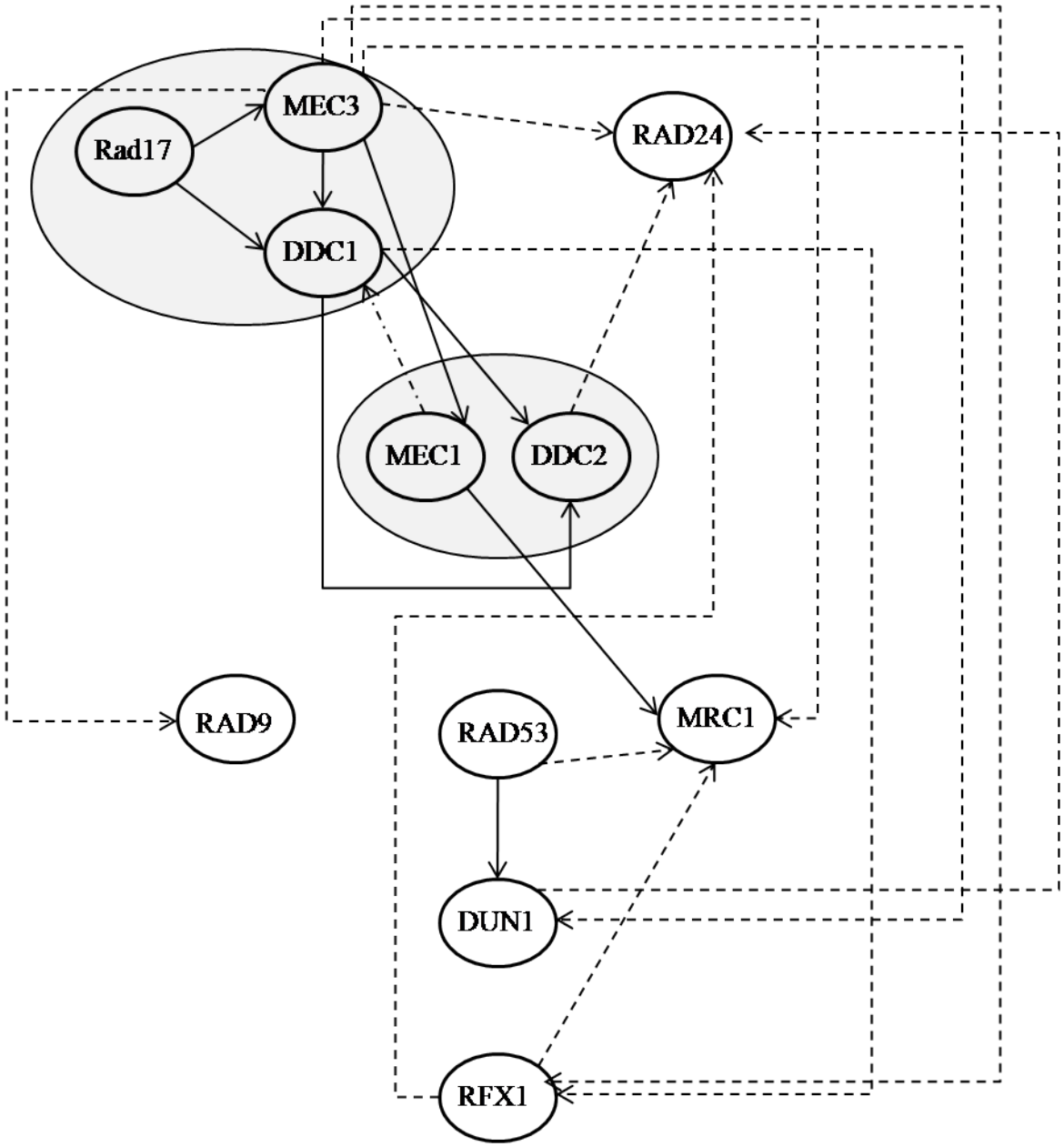
Fig. 12. Performance comparision of network MDL and time lagged network MDL algorithms on first network derived from *Saccharomyces cerevisiae*: (A) True network (B) Network inferred using TLMI-based network MDL (C) Network inferred using network MDL



(a) True network

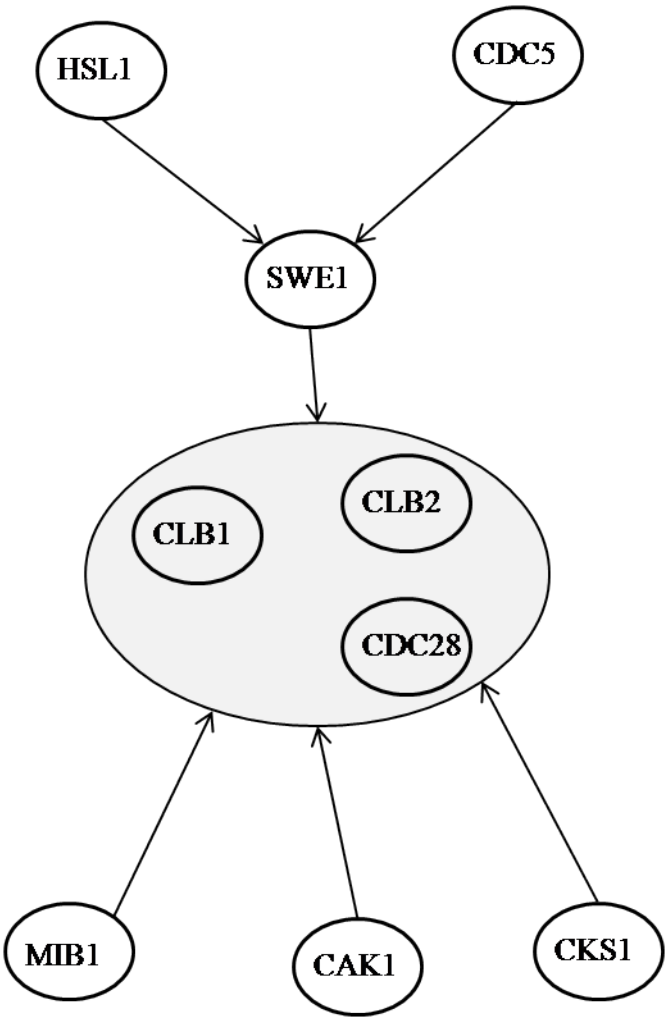


(b) Network inferred using TLMI-based network MDL

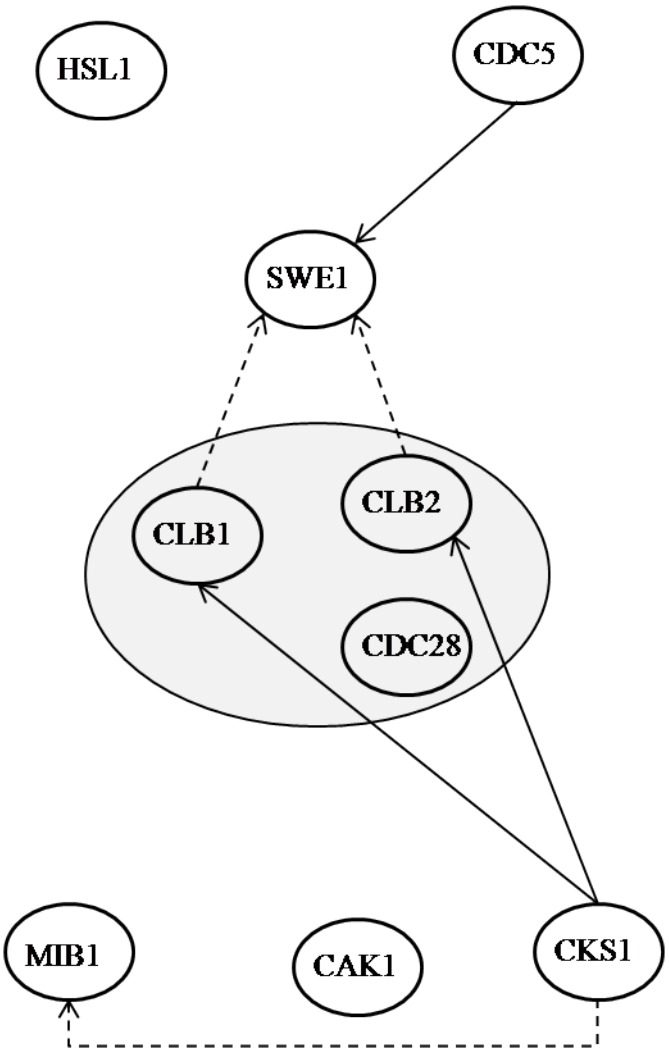


(c) Network inferred using network MDL

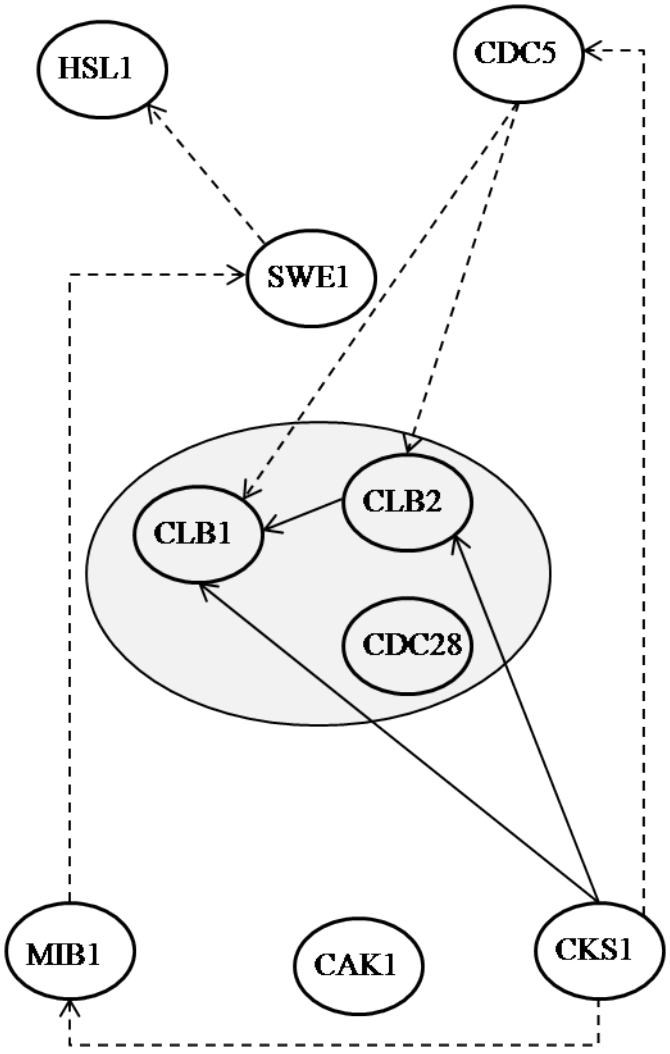
Fig. 13. Performance comparision of network MDL and time lagged network MDL algorithms on second network derived from *Saccharomyces cerevisiae*



(a) True network



(b) Network inferred using TLMI-based network MDL



(c) Network inferred using network MDL

Fig. 14. Performance comparision of network MDL and time lagged network MDL algorithms on third network derived from *Saccharomyces cerevisiae*

Table 2. Performance of MI and TLMI-based PMDL algorithmson real time Yeast data-set

Method/Metric	**PMDL**	**Time-lagged PMDL**
Precision	22.73%	19.23%
Recall	22.73%	18.52%

this phase have not been shown). While the number of correctly inferred edges is same, the total number inferred edges is different, indicating that while precision is same the recall varies. But as the difference in number of edges inferred is just one, the difference in recall is marginal. The comparable performance of the two PMDL implementations point to a need for further investigation on the time-lagged CMI metric. The precision and recall values for the algorithms are given in Table 2.

4.6 Time and Space Complexities of Time Lagged Based Network MDL and PMDL Algorithms

The performance of the time-lagged PMDL algorithm depends on three factors: number of genes, the number of time points and most importantly, the number of parents inferred for each gene by the algorithm. Time and space complexities of PMDL and network MDL algorithms were computed and analyzed to observe the role of the above mentioned factors in its performance. Pseudo-code for the time lagged versions for Network MDL and PMDL algorithms are shown in Algorithm 2 and Algorithm 3 respectively.

Line 4 of the PMDL algorithm iterates $n^2(m-t)$ times where n is number of genes, m is the number of time points, and t is the time lag. From lines 5-18 the algorithm iterates n^4 times. The algorithm iterates $n^3(m-t)$ times for lines 15 and 16. Finally from lines 20 to 31, the algorithm iterates n^3 times. Thus, the time

```
1: Input time series data
2: Preprocess data
3: Initialize M_{n×n}, C_{n×n} and P(x_j) ⇐ φ
4: Calculate the cross-time mutual info between genes and fill M_{nxn}
5: for i = 1 to n do
6:    for j = 1 to n do
7:       δ ← M_{i×j}
8:       for k = 1 to n do
9:          for l = 1 to n do
10:            if M_{k×l} ≥ δ then
11:               C_{k×l} = 1, P(x_l) ⇐ P(x_l) ∪ x_k
12:            end if
13:         end for
14:      end for
15:      Compute probabilities
16:      Computer description length
17:   end for
18: end for
19: Select MI/TLMI of model having least model length as ML/TLMI threshold δ
    and compute connectivity matrix
```

Algorithm 2: Pseudo-code for time lagged network MDL algorithm

complexity of the over-all algorithm is $O(n^4 + n^3(m-t))$.

When it comes to space complexity, the conditional probability tables play a major role. If a gene has n parents then the conditional probability tables take 2^n units of space. Thus, the amount of memory needed by the algorithm depends on the number of parents inferred by the network. As the space complexity grows exponentially based on the number of parents, it is possible that the algorithm may run out of memory for a data set with as few as 50 genes whereas it may run for as little as 5 minutes for a data set with several hundred genes. There are 2 ways to overcome this limitation:

1. Restrict the number of parents.

2. Take the next smallest description length, instead of using the smallest one.

The first approach will guarantee results when the number of parents is restricted to a small value but this may lower the accuracy of the result. The second approach may

```
Input time series data
Preprocess data
Initialize M_{n×n}, C_{n×n} and P(x_j) ⇐ φ
Calculate the cross-time mutual info between genes and fill M_{nxn}
for i = 1 to n do
    for j = 1 to n do
        δ ← M_{i×j}
        for k = 1 to n do
            for l = 1 to n do
                if M_{k×l} ≥ δ then
                    C_{k×l} = 1, P(x_l) ⇐ P(x_l) ∪ x_k
                end if
            end for
        end for
        Compute probabilities
        Computer description length
    end for
end for
Select MI/TLMI of model having least model length as ML/TLMI threshold δ and compute
  connectivity matrix
for i = 1 to n do
    for j = 1 to n do
        if C_{i×j} == 1 then
            for k = 1 to n and k != i, j do
                if CMI_{i,j,k}/TLCMI_{i,j,k} < Th then
                    C_{i×j} ← 0
                    break
                end if
            end for
        end if
    end for
end for
```

Algorithm 3: Pseudo-code for time lagged PMDL algorithm

Table 3. Comparison analysis of CLR and Time-lagged CLR algorithms on in-silico data-set

Network	**CLR**		**Time-lagged CLR**	
	Precision	**Recall**	**Precision**	**Recall**
Network 1	0.1111	0.2222	0.1250	0.4444
Network 2	0.1250	0.2500	0.1000	0.3750
Network 3	0.0909	0.2222	0.0938	0.3333
Network 4	0.1111	0.2000	0	0
Network 5	0.0909	0.2222	0.0833	0.3333
Network 6	0.1429	0.2222	0.0500	0.1111
Network 7	0.1875	0.2000	0.2500	0.6667
Network 8	0.1429	0.2222	0.0333	0.1111

take more time to run but as the number of parents is not restricted, the accuracy of the algorithm is not affected.

4.7 TLMI Based CLR Implementation

The CLR and the time lagged version of CLR were run on eight different 10-genenetworks generated using GeneNetWeaver tool. The standard precision (P) and recall (R) metrics were computed to compare the algorithms. There were 3 cases where both the metrics were better in case of CLR and 3 cases where both metrics were better in time-lagged CLR. In the remaining two cases, CLR had higher P and -lower R, but the Ps were close whereas Rs of time lagged CLR were higher than the regular CLR implementation. This shows that time-lagged CLR is effective in GRN inference. The results are tabulated in Table 3.

4.8 Conclusions

In a bid to understand the performance saturation of the information theoretic approaches, it was found that mutual information saturates and effectively tends to zero, where as the entropy of the network increases with increase in the data size. These observations indicate that MI by itself might not be the best metric in devising information theoretic approaches for GRN inference. Based on these findings, two new information theory metrics viz. TLMI and TLCMI were introduced and implemented in the network MDL and PMDL based algorithms to develop two novel GRN inference algorithms. The results indicate that transcriptional time lags play an important role in gene regulatory network inference methods as evidenced by the higher accuracy provided by the time lagged versions of network MDL and PMDL algorithms. Work in this chapter has been published in [56, 57, 58].

CHAPTER 5

INFERENCE USING KNOCK-OUT DATA

While the inference algorithms primarily work on the microarray data, other data sets can be incorporated to further improve the quality of inferred networks. One such data set that can be used is the knock-out data. This Chapter proposes a novel method to incorporate the knock-out data towards regulatory network inference. The performance analysis shows that including knock-out data is effective for inferring networks.

5.1 Knock-out Data

Knock-out data provides the expression levels of genes when a particular gene(s) is completely down-regulated. This data is effective in determining direct regulatory interactions [59] and is optimal for inferring regulatory networks [60]. These experiments are useful when certain genes of interests are targeted. External conditions that down-regulate these genes are used to carry out the experiments. As microarray experiments are expensive both in terms of financial and human resources, they cannot be carried out on a large scale. But the knowledge from these experiments can be used in network inference to improve the inference accuracy.

5.2 Knock-out Network Generation

To generate the knock-out network, A simple model where fold-change computation of gene expression with respect to the wild type data (or expression data in normal condition) was used. If the fold-change is greater than 1.2 or less than 0.7, it

is assumed that the gene has a direct regulatory relationship with the knocked out gene. The thresholds chosen were based on suggestions given in [12].

5.3 Number of Parents in *E. coli* Genome

Studies on Yeast network [1] showed that a gene is usually regulated by a small number of transcription factors. This property can be used to significantly reduce the complexities of algorithms such as network MDL, PMDL, DBN etc. In this section, the study is extended over the *E. coli* network. The gene regulatory network for *E. coli* was obtained from RegulonDB release 6.2 [61]. The network has 1502 genes and 3587 regulatory interactions. A simple MATLAB [62] script was used to find the number of parents (regulators) for every gene in the network. Figure 15 shows that genes usually have a small number of parents as observed in Yeast network [1]. This plot suggests that restricting the maximum number of parents (regulators) between three and six seems to be the optimal choice in GRN inference.

5.4 Network Generation Using Parent Restriction

The parent restriction approach for network inference takes the following as inputs:

1. Knock-out network
2. Network inferred using expression data
3. MI values of edges in network inferred using expression data
4. Time-lag values of genes connected in the network inferred using expression data

For every gene the number of unique parents (regulators) from both knock-out network and network inferred using expression data is computed. If the number of

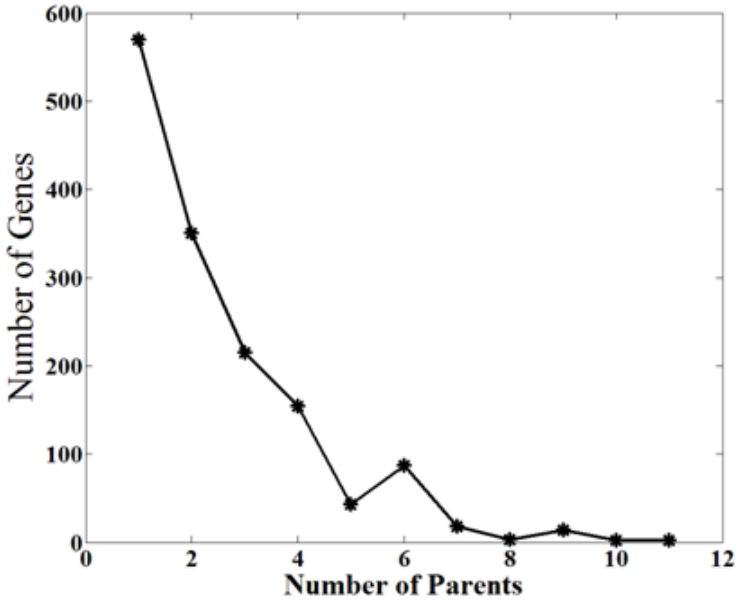


Fig. 15. Analysis on number of parents that each gene has in e-coli network

regulators is above the desired number of regulators then a parent selection approach is implemented. Figure 16 illustrates the parent selection process. In this Figure, three networks, Network 1, Network 2 and Network 3 are shown. Network 1 is obtained using expression data. The edges in this network are labelled by the MI values and nodes are labelled by the time lags. Network 2 is the knock-out network and Network 3 is the network obtained after parent selection process is applied for gene A.

During the parent selection process a union of Network 1 and Network 2 is made and then the parents are selected from the merged network based on the following priorities:

1. Knock-out edges

2. MI values

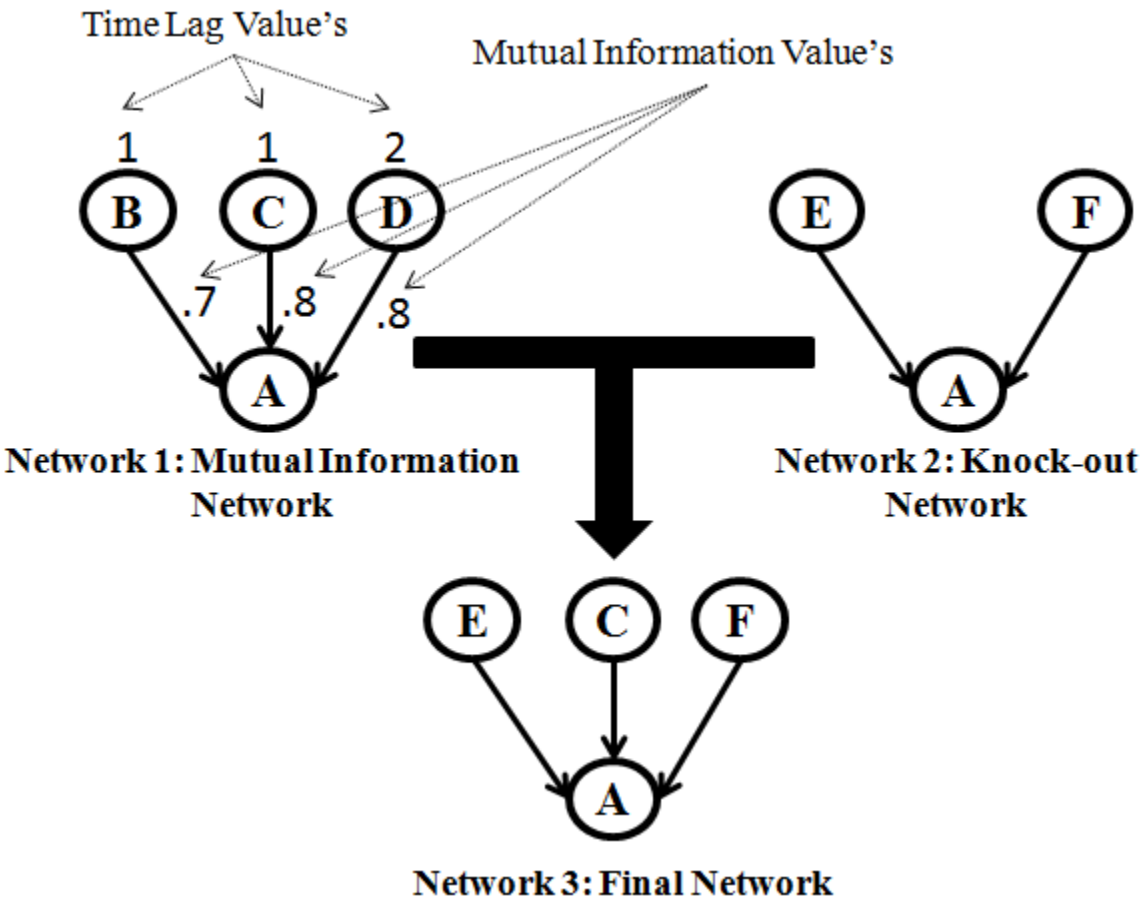


Fig. 16. Parent restriction approach towards network inference

3. Time Lags

Irrespective of the MI values or time lags, the edges from knock-out network are added first. If the number of parents in Network 2 (knock-out network) is less than the number of desired parents, then the remaining parents from Network 1 are chosen based on MI values first. If there is a conflict in MI values, then the edge which has smaller time lag is chosen. Figure 16 gives an example of parent restriction approach. In this example the maximum number of parents that node A can have is three. Genes E and F, from Network 2, are chosen as the first two parents; the third parent can now be selected from Network 1. Two genes C and D in Network 1

have maximum MI value but as gene C has a shorter time lag, gene C is chosen as the third parent. Similarly, this process is applied to remaining genes to infer their respective parents. These genes along with the inferred parents are merged to obtain the final network.

5.5 Sensitivity Analysis on Number of Parents

The knock-out network incorporation algorithm was implemented on *in-silico* synthetic networks. This network and the time series data for networks of size 50 and 100 were generated using the GeneNetWeaver. The knock-out data for the networks were also generated using GeneNetWeaver. For every network, the time series duration was 200 minutes and the time gap between each experiment was 10 minutes. Hence, the data set has 21 time points in which the first one represents the expression values for control and the remaining 20 of them simulate gene expression changes under some external perturbation. For each of these *in-silico* networks, the execution time and precision and recall metrics were computed from three to ten number of parents.

As the algorithm is designed to handle cases where large number of connections is inferred in the MI matrix, low MI and CMI thresholds were chosen instead of running it over an array of MI values and selecting the best one based on description length. Table 4 shows precision and recall for networks inferred with parents restricted from three to 10. The precision decreased as the number of parents increased, indicating that there were more wrong edges whereas recall slightly increased in both networks. The precision and recall values in the table indicate that restricting the number parents for each gene between three and six is desirable.

The memory consumed by each of these networks was also studied. The computation of memory consumption by the network follows the description in [23]. Figure

Table 4. Sensitivity Analysis of Precision and Recall Values for Different Number of Parents on a 50-gene and 100-gene network

Number of Parents	Precision (50)	Recall (50)	Precision (100)	Recall (100)
3	0.5212	0.4851	0.3379	0.5214
4	0.4666	0.4851	0.2941	0.5357
5	0.4561	0.5148	0.2682	0.5500
6	0.4344	0.5247	0.2468	0.5571
7	0.4076	0.5247	0.2309	0.5642
8	0.3970	0.5346	0.2185	0.5714
9	0.3802	0.5346	0.2087	0.5785
10	0.3716	0.5445	0.2053	0.6000

17 shows that as the number of parents increased, there was an exponential increase in the memory consumption.

5.6 Scalability and Performance of the Algorithm

As knock-out data at genome scale is currently not obtainable, the knock-out network was randomly generated. The goal here is to show that the overall algorithm can handle large networks. The complete *E. coli* time series data was obtained using GeneNetWeaver. Table 5 shows the average time and memory consumed by the network to run for each parent restriction between three and seven for the complete *E. coli* network.

The algorithm was run 10 times and the average run time for the algorithm was computed. A machine with quad-core Xeon processor and 8 GB RAM was used to run the simulations. The algorithm was coded in MATLAB 7.8.0 (R2009a) [62] and the operating system used was Windows 7 professional. Table 5 shows the number

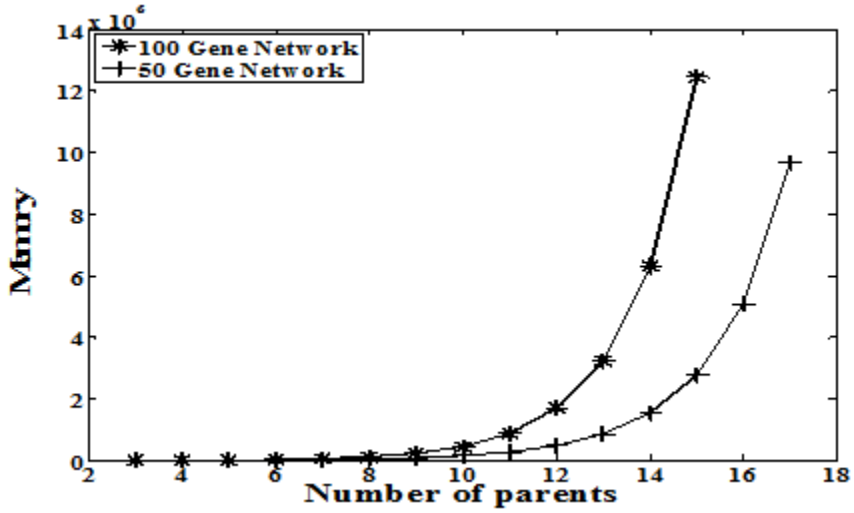


Fig. 17. Memory consumption of parent restriction approach for different number of parents

Table 5. Memory Consumption of parent restriction approach for Different Number of Parents

Number of Parents	Number of connections inferred	Memory consumed in bits	Average run time in seconds
3	4506	109646	3457
4	6008	186248	3459
5	7510	322930	3458
6	9012	579772	3458
7	10514	1076934	3460

of connections inferred, the memory consumed in bits and the time taken in seconds. It can be noted that the execution time did not increase significantly with higher number of parents (that also increased the number of inferred edges). One plausible reason for this is because the number of inferred edges did not change by any order of magnitude with different levels of parent restriction for the considered data sets. This observation however requires further investigation to be proven empirically.

5.7 Conclusions

The relationship between a gene and the number of parents that each gene has in *E. coli* was studied. This study showed that most of the genes have a small number of parents (less than six). Similar results were also found in the genome level studies of other organisms. This motivated restriction of the number of parents each gene can have from our previous work which led to the proposed scalable knock-out data incorporation inference algorithm. It was also shown that including gene knock-out data in the algorithm substantially improved the inference accuracy. The proposed algorithm has the ability to infer very large networks at genome scale as shown in preliminary analysis. Work in this chapter has been published in [9].

CHAPTER 6

SREVEAL: SCALABLE EXTENSIONS OF REVEAL

The scalability of the REVEAL algorithm can be improved by utilizing the transcription factor list as prior knowledge and implementing time lags to further reduce the potential transcription factor list for each gene (note that transcription factors act as the parents for each gene in the regulatory networks). Two variations of the REVEAL algorithm are discussed in this chapter. While the first algorithm is a direct extension of the basic concept of REVEAL, the second variation extends this algorithm wherein the scalability is further improved by implementing time lags to reduce the potential transcription factor list of each gene. The two extensions will be referred as to sREVEAL-1 and sREVEAL-2 respectively in the rest of the chapter. This chapter is primarily focused on implementing the scalable extensions to REVEAL to obtain more accurate regulatory networks. Note that prior knowledge such as knock-out network can be easily incorporated into these algorithms as discussed in Chapter 5.

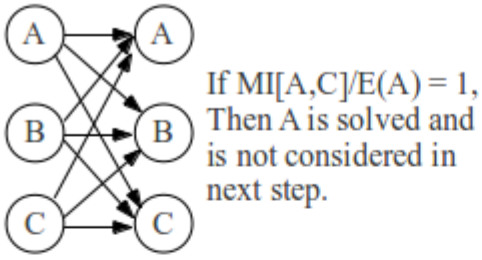
6.1 sREVEAL-1

By considering a set of known transcription factors, the number of iterations REVEAL has to execute for different input combinations is reduced significantly thus reducing both space and time complexities. Transcription factor prediction approach [63, 64] can be used to identify transcription factors with high accuracy. As empirical studies in the past have shown that a gene is regulated by a small set of transcription factors [1, 9], the number of regulators for each gene is restricted to three, four and five respectively. This consideration further reduces complexity of the algorithm. Unlike

in REVEAL where it keeps solving for higher number of parents for all genes as it proceeds, the proposed sREVEAL-1 and sREVEAL-2 algorithms solve one gene at a time. If the number of parents is restricted to, for example, three transcription factors, mutual information between all combinations of one, two and three pairs of transcription factors is computed. Then the combination of transcription factors that produces the maximum MI value with the gene is chosen as the regulators of the gene. Figure 18 gives a pictorial presentation of REVEAL and sREVEAL-1 algorithms.

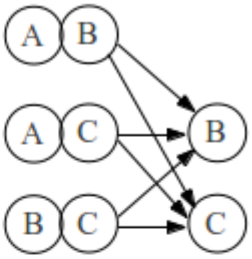
In REVEAL, first all pair-wise combinations are checked to find the regulators of the genes; the genes for which the MI to entropy ratio equals "*one*" are taken out of consideration in next step. For e.g. in Figure 18(A), gene A is solved in the first step as the ratio of MI between A and C to the entropy of A is equal to one, thus gene A is not considered in the next step of the algorithm. For the remaining genes, all combination of two genes that can act as their parents are checked similarly, and if the ratio equals 1, are taken out of consideration; the algorithm next looks into combination of three genes to act as parents of the remaining genes and so on until all genes are solved.

Figure 18(B) explains how sREVEAL-1 works. A maximum of two regulators for each gene is shown in this case (note that actual number of regulators restricted in performance analysis is three, four, and five). Hence, for every gene all possible combinations of one and two genes are checked and the combination having maximum MI is chosen. While REVEAL may not find solutions for certain genes, sREVEAL-1 finds a solution for every gene as the maximum MI was selected. It should be noted that in practice only transcription factors can be regulators in a transcriptional regulatory network.



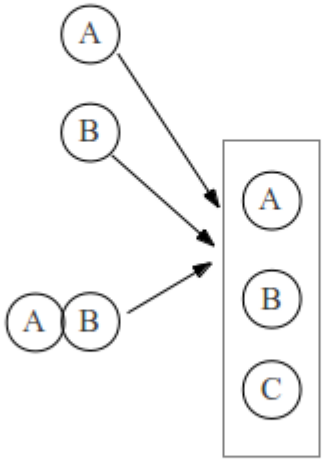
If MI[A,C]/E(A) = 1, Then A is solved and is not considered in next step.

Step 1 of REVEAL



Step 2 of REVEAL

(A)



In sREVEAL-1 approach, for every gene, MI between all possible 1..*m* combinations of TF's are computed where *m* is the maximum number of desired parents/regulators. Then the TF's in the combination that gives maximum MI are inferred as parents of that gene.

(B)

Fig. 18. REVEAL and sREVEAL approaches methodology

6.2 sREVEAL-2

Though the above mentioned approach reduces complexity and can infer the network for several hundred genes, it can still run out of memory when the transcription factor list itself is large (>50) as in the case of the complete *E. coli* transcriptional regulatory network. This issue can be solved by further reducing the potential set of transcription factors for each gene by using time lags. A transcription factor can regulate a gene if and only if the time lag between the transcription factor and the gene is greater than zero. It is further suggested that, very large time lags should be set to zero. Hence, transcription factors having very large time lags must not be considered as potential regulators and higher priority must be given to transcription factors having smaller time lags [65]. Thus even after filtering the potential regulators using time lags, the potential regulators list can still be large; further filtering can be performed by considering only small time lags between transcription factors and genes. This approach can be used to infer fairly large regulatory networks.

6.3 Results

6.3.1 Data Sets

Two networks with 75 genes/transcription factors and their respective data sets were generated using the GeneNetWeaver tool. The tool provided six types of data sets viz. wild-type, knockouts, knockdowns, multi-factorial perturbations, and time series. Only time series data-set was used in performance analysis. The time series data set for each network had five different gene expression matrices under different perturbations; the first data set for each network was chosen for the analysis, also the data had 11 time points with a time interval of 10 minutes between each time point. The data is quantized to three levels. The log ratio of every time point is computed;

Table 6. Sensitivity Analysis on threshold selection for the CLR algorithm

Network	Network 1		Network 2	
Threshold/Metric	Recall	Precision	Recall	Precision
0.9	8.3	5.0	8.8	4.1
1.0	4.4	5.2	2.6	2.3
1.1	2.4	5.8	2	3.5
1.2	2	4.3	1.0	5.7

for ratios above 1.1 and below 0.9, values of 1 and -1 are assigned; else a value of 0 is assigned.

6.3.2 Performance of sREVEAL-1 and sREVEAL-2

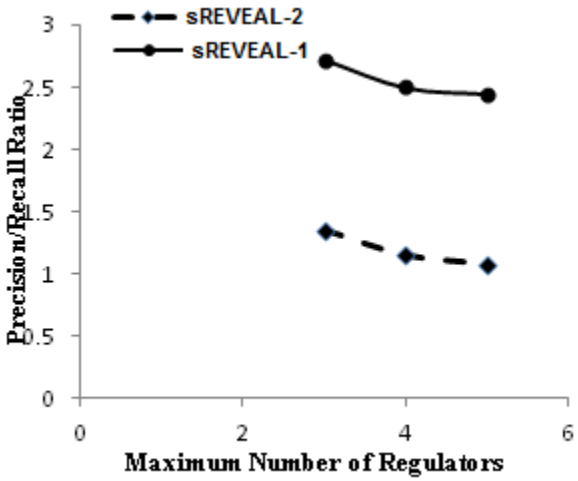
As the empirical studies in the past have indicated that a gene is generally regulated by a small number of transcription factors [1, 65], the sREVEAL-1 and sREVEAL-2 algorithms were implemented with a maximum of three, four and five regulators for each gene. The precision-recall ratio plots for the networks as shown in Figure 19 indicate that selecting three as maximum number of regulators gives the best performance and sREVEAL-1 performs better than sREVEAL-2.

6.3.3 Performance of CLR

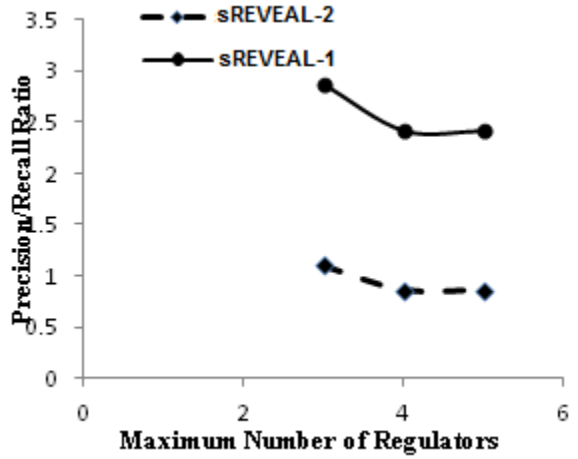
CLR requires the user to mention a threshold value in order to infer a network. While different thresholds on Z-score can result in significant fluctuations in the accuracy of the inferred networks, only the best results are reported in the Table 6 (corresponding to the best threshold).

6.3.4 Comparison of sREVEAL-1 and sREVEAL-2 with CLR

Table 7 displays the recall and precision metrics for the sREVEAl-1 and sREVEAL-2 algorithms. As the precision recall plots suggest, that setting the maximum number



(a) Network 1



(b) Network 2

Fig. 19. Sensitivity analysis on number of parents using precision recall ratio metric

Table 7. Performance comparison of sREVEAL Approaches

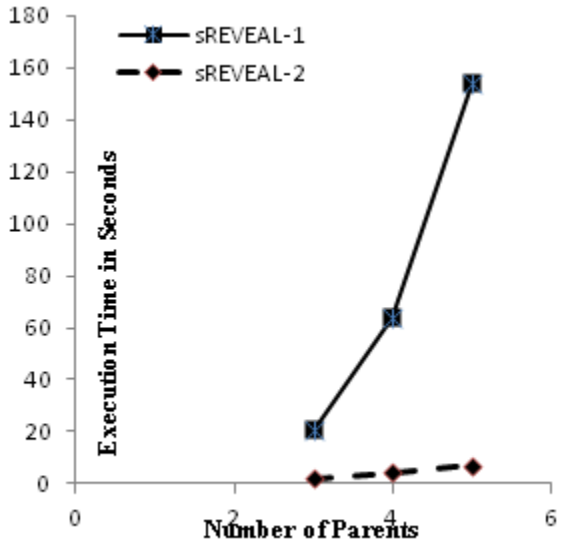
Network	Network 1		Network 2	
Algorithm/Metric	**Recall**	**Precision**	**Recall**	**Precision**
sREVEAL-1	2.9%	7.9%	3.6%	10.3%
sREVEAL-2	2.4%	3.25%	14%	15.5%

of regulators to three performed well in general, only the results from these algorithms in which the maximum number of regulators was set to three are shown. The best solution for Network 1 amongst all the three algorithms is achieved by CLR when the threshold is set to 0.9. For every other value of the CLR threshold, the proposed sREVEAL-1 and sREVEAL-2 algorithms performed better (data not shown). However, for Network 2, both proposed algorithms outperform the CLR algorithm even with the optimal threshold (note that such optimal thresholds are not known apriori).

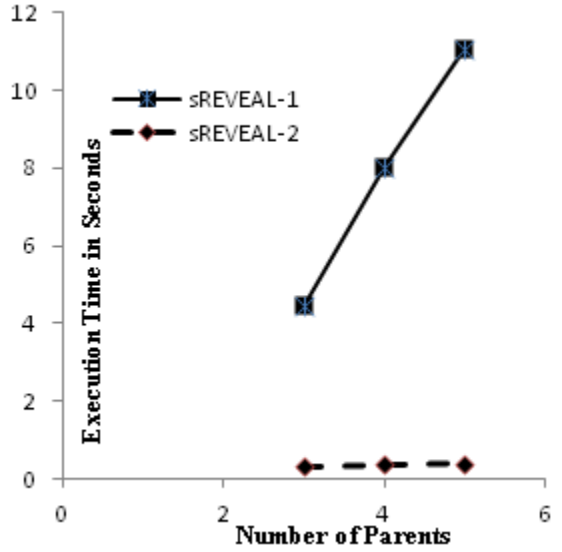
While CLR did outperform the proposed approaches for that specific threshold, in practice, selecting the optimal threshold is not practical. Moreover, the optimal threshold value varies between different datasets and across different networks. It is difficult to come up with an optimal range of thresholds that can work in all cases. The sREVEAL-1 and sREVEAL-2 algorithms are thus more practical and robust as they do not depend on any user-defined threshold and can still outperform CLR in most cases as the results indicate. Even for Network 1, the sREVEAL-1 and sREVEAL-2 algorithms performance is comparable to the best possible score achieved by CLR with the optimal threshold settings.

6.3.5 Performance of sREVEAL-1 and sREVEAL-2 in Terms of Execution Time

The worst case time and space complexity of sREVEAL-1 and sREVEAL-2 algorithms are the same as REVEAL, but such a case in highly unlikely (note that the



(a) Network 1



(b) Network 2

Fig. 20. Sensitivity analysis on execution time for sREVEAL-1 and sREVEAL-2 algorithms

worst case scenario will occur when all the transcription factors can act as potential parents of each gene in the network). Hence, the number of potential parents is the most important factor in evaluating the performance of these algorithms. Figure 20 indicates that the sREVEAL-2 is more efficient in terms of execution time (as time lags further reduce the number of potential parents for each gene in the network). The performance (in terms of accuracy) of sREVEAL-1 was best over the 250 gene network followed by sREVEAL-2 and CLR in that order. Also, for sREVEAL-2, a maximum time-lag value of five was chosen as the cut-off, such that TF-gene pairs having time-lag >5 are removed from consideration as potential parents.

6.4 Conclusions

Scalable extensions of the REVEAL algorithm to infer larger regulatory networks were presented in this chapter. These extensions outperform the CLR algorithm in most practical settings where the optimal threshold for CLR will not be known. When multiple combinations provide the maximum MI value, the algorithm implements an ad-hoc scheme of selecting the first occurrence. A more logical implementation using a MDL based approach can be achieved. This work appeared in [65].

CHAPTER 7

INFERENCE BASED ON SCORING COMPLEX INTERACTIONS

Structural analysis over well studied transcriptional regulatory networks indicates that these networks are made up of small set of reoccurring patterns called motifs [66, 67, 68]. This chapter introduces and discusses novel scoring schemes based on complex interactions/structures to improve the relevance networks class of inference algorithms. Instead of inferring direct edges based on interactions, the goal is to infer edges by scoring them based on structures.

7.1 Scoring Schemes

Unlike the relevance network class of algorithms and other works, in which network inference is based on just MI computations, an attempt to infer a network by scoring edges based on complex interactions is made. To scale up to genome level inference, the mutual information computations are restricted to infer three node structures (a gene and its two potential regulators). The proposed scoring schemes based on complex interactions start with computing the mutual information between a gene/transcription factor and every other transcription factor and store them in bin-1. After computing the pair-wise scores, the MI between a gene/transcription factor and every two node combinations of transcription factors are computed and stored in bin-2. Both bins are assembled in to a main MI matrix M.
The number of combinations in each bin is given by $\binom{t}{b}$, where b is bin number and t is the number of transcription factors. Hence, the total number of TF combinations n_c considered across both the bins for any particular gene/TF is given by Equation

7.1.

$$n_c = \sum_{b=1}^{2} \binom{t}{b} \tag{7.1}$$

Studies on genome scale networks [1, 9] have shown that there are a number of genes which are regulated by just one regulator, in order to infer these edges, two different weighting scheme based on MI values of pair-wise as well as complex structures is implemented. The following section discusses the weighting scheme.

7.1.1 Scoring Scheme 1

In order to compute a weighted score between a gene and a transcription factor, all the pair-wise MI scores and the MI scores for the two node combinations in which the transcription factor participates is taken in to consideration. Let us denote a vector V which holds all the MI values from bin 2 in which TF T_i has participated. For example as in Figure 21 the vector V for TF T_1 consist of the MI values of combinations T_1T_2, T_1T_3, and T_1T4. Note that V holds the MI values in the order of occurrence of a specific transcription factor from bin 2. The length l of the vector, V, when maximum number of regulators for each gene/TF is restricted to two, is:

$$l = t - 1 \tag{7.2}$$

The score between a gene G_i and a TF T_j is given as:

$$B^2(G_i, T_j) = \sum_{k=1,l=1,k!=j}^{t,l} \frac{I(G_i, T_j)}{I(G_i, T_j) + I(G_i, T_k)} * V_l \tag{7.3}$$

7.1.2 Scoring Scheme 2

In this scoring scheme instead of just using pair-wise MI values as weights (as in scoring scheme 1), the actual MI values are also used to score an edge. As MI values between higher number of variables is usually higher [69], i.e. MI values between

	Bin 1			Bin 2		
	T_1	...	T_t	T_1T_2	...	$Tt_{-1}T_t$
G_1	$I(G_1,T_1)$		$I(G_1,T_t)$	$I(G_1, T_1T_2)$		$I(G_1,T_{t-1}T_t)$
. .						
G_g	$I(G_g,T_1)$		$I(G_g,T_t)$	$I(G_g, T_1T_2)$		$I(G_g,T_{t-1}T_t)$
T_1	$I(T_1,T_1)$		$I(T_1,T_t)$	$I(T_1, T_1T_2)$		$I(T_1,T_{t-1}T_t)$
. .						
T_t	$I(T_t,T_1)$		$I(T_t,T_t)$	$I(T_t, T_1T_2)$		$I(T_t,T_{t-1}T_t)$

(A)

	Bin 1				Bin 2					
	T_1	T_2	T_3	T_4	T_1T_2	T_1T_3	T_1T_4	T_2T_3	T_2T_4	T_3T_4
G_1	0.4	0.3	0.2	0.7	0.5	0.2	0.8	0.4	0.6	0.7

$$I(G_1,T_1) = \frac{I(G_1,T_1)}{I(G_1,T_1)+I(G_1,T_2)}\ X\ I(G_1,T_1\ T_2) + \frac{I(G_1,T_1)}{I(G_1,T_1)+I(G_1,T_3)}\ X\ I(G_1,T_1\ T_3) + \frac{I(G_1,T_1)}{I(G_1,T_1)+I(G_1,T_4)}\ X\ I(G_1,T_1\ T_4)$$

$$= \frac{0.4}{0.4+0.3}\ X\ 0.5 + \frac{0.4}{0.4+0.2}\ X\ 0.2 + \frac{0.4}{0.4+0.7}\ X\ 0.8 \sim = 0.7099$$

$B^2(G_1,T_i)$	0.7099	0.6342	0.3822	1.4735

(B)

	Normalized B^1				Normalized B^2			
	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4
G_1	0.25	0.1875	0.125	0.4375	0.2218	0.1981	0.1194	0.4604

$F_2(G_1,T_i)$
0.4718
0.3856
0.2444
0.8979

(C)

Fig. 21. (A) MI computation scheme (B)Example for scoring scheme 1 (B)Example for scoring scheme 2

different number of variables are at different scales, the scores in bin 1 and bin 2 needs to be normalized first; such bin-wise normalized scores can then be combined to achieve a final score. After implementing scoring scheme 1, the scores from bins 1 and 2 are combined to obtain the final score. If $B^1_{i,j}$ designates the pair-wise MI scores and $B^2_{i,j}$ designates the scores from scoring scheme 1 then the final score in scoring scheme 2 is given as:

$$F_2(G_i, T_j) = \frac{B^1_{i,j}}{\sum_{k=1}^{t} B^1_{i,k}} + \frac{B^2_{i,j}}{\sum_{k=1}^{t} B^2_{i,k}} \tag{7.4}$$

Using equations 7.3 or 7.4 the scores are computed for each gene/TF individually, the simple scoring schemes combines the effects of higher level structures (stored in each higher order bin) for each gene/TF individually. Figure 21 shows the MI computation scheme and an example for scoring scheme 1 and scoring scheme 2. In this example a network with a single gene and four transcription factors is considered. Using scoring scheme 1 three algorithms aCoIn (ad-hoc complex interactions inference algorithm), aCoIn-5 and aCoIn-10 are proposed. aCoIn implements the scoring scheme 1 to score a gene/TF and all its possible regulators. The final scoring matrix here is exactly similar to any other relevance network based scheme, where the rows designate a node (gene/TF) and the columns designate all possible regulators, i.e., TFs. Hence, a threshold can be applied on this final scoring matrix to obtain the final connectivity matrix. aCoIn-5 and aCoIn-10 are extensions of aCoIn algorithm in which the top 5% and 10% scored edges from bin one are added to the resulting networks in an ad-hoc fashion. Using scoring scheme 2, three algorithms sCoin (scalable complex interactions inference algorithm), sCoin-Z1 and sCoin-Z2 are proposed. While sCoIn implements scoring scheme 2 as is, ScoIn-Z1 and sCoIn- Z2 are extensions of the sCoIn algorithm in which the initial MI matrix is further updated using

different Z-score schemes before the scoring scheme is applied. Again as MI values between different numbers of variables are at different scales, Z-score computations were performed for each bin separately. In sCoIn-Z1 algorithm, the MI values are standardized for each gene/TF separately considering the computed means and standard deviations for each row in each bin separately, as highlighted in 21, where as in sCoIn-Z2, the MI values were standardized for each bin as in the CLR algorithm. The pseudo-code for the main approach is given in algorithm 4.

Expression Data: data
Number of nodes: N
Regulator List: T
Number of regulators: t
$muInfoMat(N, n_c) = 0$
for $i = 1$ to N **do**
  for $j = 1$ to n_c **do**
    if i is not in regulator combination **then**
      $muInfoMat(i, comb(j)) = MI$
    else
      $muInfoMat(i, comb(j)) = 0$
    end if
  end for
end for
for $i = 1$ to N **do**
  for $j = 1$ to t **do**
    Obtain final scores using equation 7.3 or equation 7.4
  end for
end for

Algorithm 4: Pseudo-code of sCoIn and aCoIn algorithms

7.2 Results

7.2.1 Networks and Datasets

7.2.1.1 E. coli Network and Expression Data using the GeneNetWeaver tool

The known complete *E. coli* etwork and data sets were obtained using GeneNetWeaver tool. The tool provided six types of data sets viz. wild-type, knockouts, knockdowns, multi-factorial perturbations, and time series. Only the time series data sets were

chosen for the analysis (the algorithm works for non time-series data too). All the data sets were generated under the DREAM 4 challenge settings. The time series data set had ten different gene expression matrices under different perturbations and the first three data sets were chosen for the analysis. The time series data had 21 time points with a time interval of ten minutes between each time point. The first time point in the data is the control where the expression levels of untreated cells are recorded. For performance analysis, a simple fold-change model was implemented. Using the control time point, the fold-change for every time point was computed. The fold-change is the ratio between the expression levels of a gene/TF at that time point to the control time point. The network provided by GeneNetWeaver had a total of 1389 genes and 176 transcription factors.

7.2.1.2 *E. coli* Network and Data from the CLR compendium Data Set

Compendium data sets for E. coli which has 524 expression profiles for the complete genome are provided in [23]. While a much more comprehensive network was consolidated in the data set, the known network only from the Regulon database was considered in this analysis. The data set of the third version of Regulon database used had 1144 genes/TF among which 152 were TFs.

7.2.1.3 Entropy Estimation for Continuous Data

A number of methods such as binning/histogram method [70], K-th nearst neighbor (KNN) method [69], B-spline method [71], kernel-based methods [72, 73, 74] etc. exist for entropy computation. The B-spline method in [71] was implemented and used in this analysis.

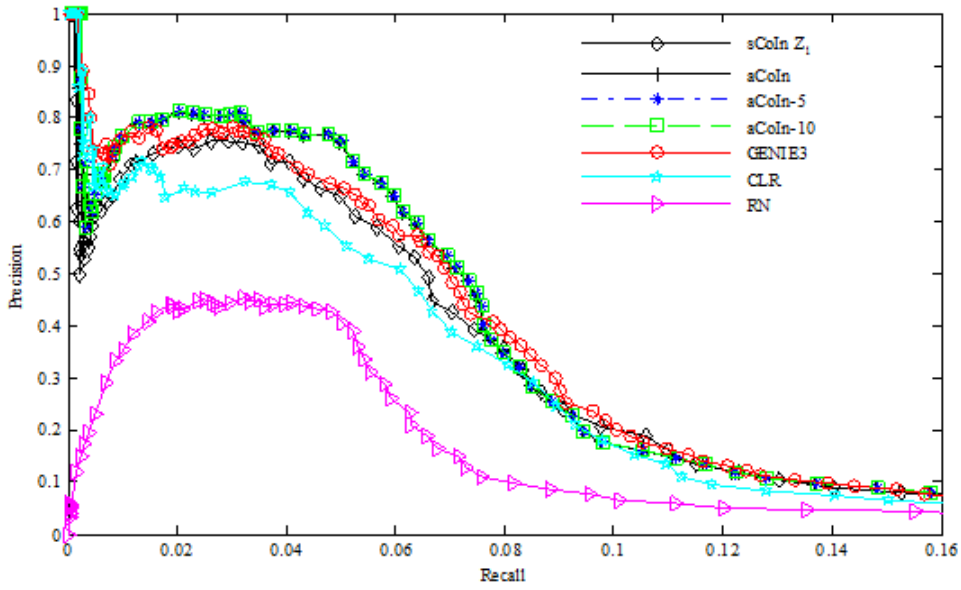


Fig. 22. Performance analysis of sCoIn-Z1 , aCoIn, aCoIn-5, aCoIn-10, GENIE3, CLR, and RN algorithms on CLR compendium data set

7.2.2 Performance Analysis of Algorithms

7.2.2.1 Precision-Recall Analysis

The results for the CLR compendium datasets are first reported. aCoIn performed best on these datasets in comparison to the proposed sCoIn variations, CLR, GENIE3 [69] and RN. However, it was observed that including 5% or 10% of the top-scored edges from bin-1 in the aCoIn algorithm did not appreciably impact the inference accuracy (either precision or recall). As seen in Figure 22, all the variations of aCoIn overlapped significantly. Hence, it appears that in this particular dataset, the scoring scheme automatically brought in most of the top-scored bin-1 edges pointing to its overall efficacy. While this requires further investigation, only aCoIn implementations based on bin-2 will be considered in the subsequent analysis. From Figure 22 it can also be observed that sCoIn-Z1 algorithm worked better than

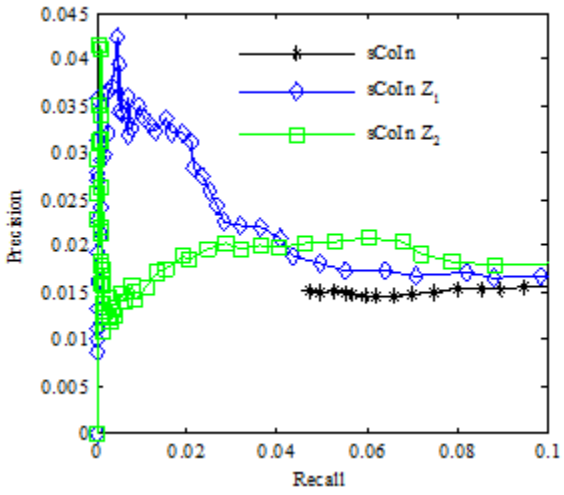


Fig. 23. Performance analysis on sCoIn approaches

the CLR and RN algorithms.

In Figure 23, the performance of the three sCoIn variations on GeneNetWeaver data set 1 is reported. It can be observed that sCoIn-Z1 outperforms sCoIn and sCoIn-Z2 algorithms. The same trend was observed for the other two datasets as well. Hence, sCoIn-Z1 is considered over sCoIn and sCoIn-Z2 algorithms in further analysis.

Figures 24 and 25 report the performance of sCoIn-Z1 and aCoIn in comparison to CLR, GENIE3 and RN for the time-series datasets 1 and 2. aCoin and sCoIn-Z1 performance is comparable to CLR and GENIE3 on dataset 1. aCoIn and GENIE3 outperforms the other schemes in dataset 2. Hence, it can be argued that the relative performance of sCoIn and aCoIn algorithms are dependent on the specific time-series datasets and further analysis is required to assess the conditions in which one works better than the other. However, the proposed algorithms do seem to outperform the other inference methods as shown in these results.

Time-series dataset-3 however was an outlier, wherein the RN method performed

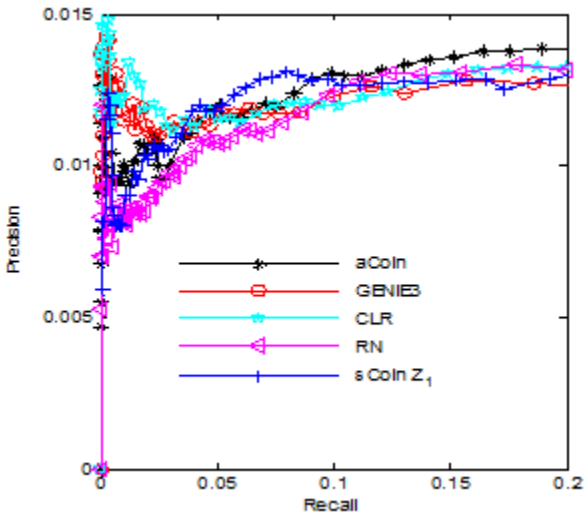


Fig. 24. Performance analysis of aCoIn, GENIE3, CLR, RN, and aCoIn-Z1 algorithms on data set 1

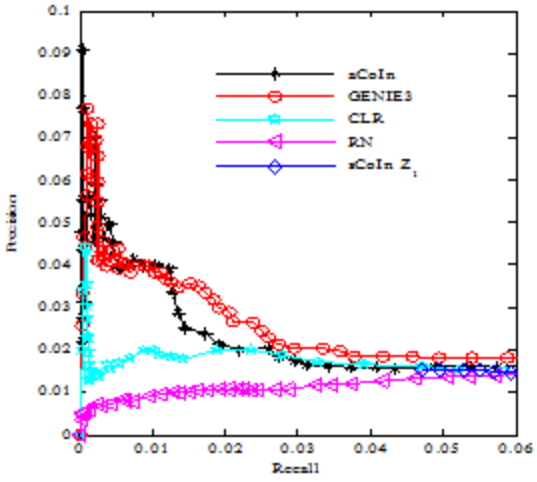


Fig. 25. Performance analysis of aCoIn, GENIE3, CLR, RN, and aCoIn-Z1 algorithms on data set 2

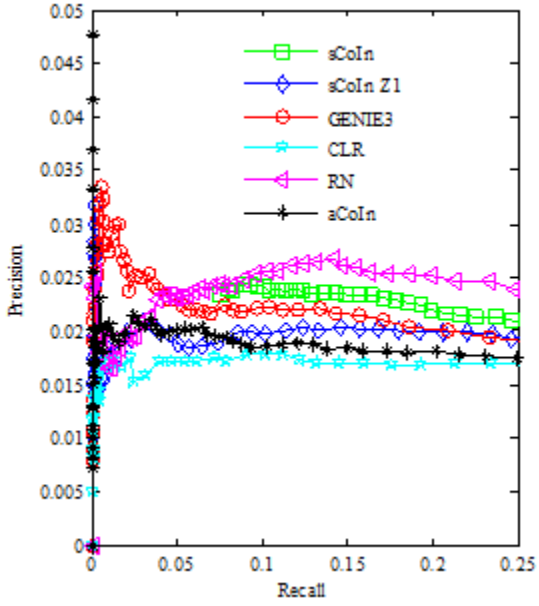


Fig. 26. Performance analysis of aCoIn, GENIE3, CLR, RN, and aCoIn-Z1 algorithms on data set 3

the best (Figure 26) followed by the sCoIn implementation. Hence, it appears that the sCoIn and aCoIn algorithms seem to have different features that makes one outperform the other depending on the datasets analyzed. However, both appear to work better than the other approaches that have been implemented.

7.2.2.2 Sensitivity Analysis on Number of Bins:

Note that the earlier results were generated by considering only up to 2 bins in the proposed sCoIn and aCoIn algorithms. However, as these algorithms can potentially consider even higher number of TF combinations for inference, in this analysis the accuracy behavior of the sCoIn-Z1 algorithm is studied in which two, three and four number of bins are considered. As the computational complexity of the scoring scheme increases exponentially with more number of bins, a smaller sized network with 100 genes and 16 TF's was chosen for this particular analysis. The

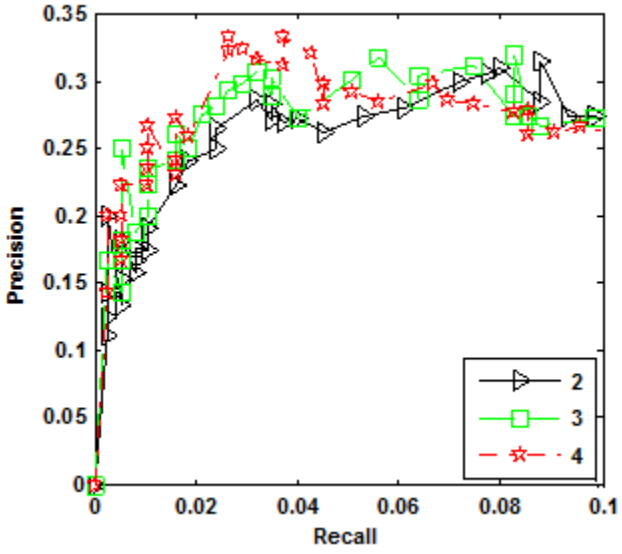


Fig. 27. Sensitivity analysis on number of bins for sCoIn-Z1 algorithm

network and data sets were obtained using the GeneNetWeaver tool.

Plots in Figure 27 show that there was a small improvement in accuracy when higher number of bins was considered for inference. Hence, to keep our algorithms scalable to genome-scale inference, it is sufficient to consider up to 2-bins only. Higher order bins may only give a slight improvement in the accuracy.

7.3 Extensions of aCoIn and sCoIn Approaches

The aCoIn and sCoIn approaches considered structure-based inference. The three node structures in these approaches consisted of a gene and two transcription factors. As a direct extension to this structure-based inference approaches, aCoIn-e and sCoIn-e are proposed. These proposed extensions considered all possible combinations of genes and transcription factors. The three node structures consisting of genes and transcription factors in these extension methods are:

1. Gene-Gene-Gene.

2. Gene-TF-Gene.

3. Gene-TF-TF.

4. TF-TF-TF.

5. TF-TF-GENE.

6. TF-GENE-GENE.

These extensions are computationally more complex than the sCoIn and aCoIn approaches, as they involve computing mutual information scores for a higher number of combinations.

7.3.1 Preliminary Analysis of aCoIn-e and sCoIn-e Approaches on DREAM 4 Data Sets

7.3.1.1 Data Sets

Five DREAM 4 challenge time series data sets of network size 100 were chosen for performance evaluation. The data set contained time series data for 10 replicates. Only the data from first replicate was chosen for the analysis.

7.3.1.2 Performance Evaluation Metric

The DREAM challenge performance evaluation score was used for comparison analysis. Both precision-recall (PR) and receiver operating characteristic (ROC) curves were used in this score. The different statistics used by the scoring scheme is summarized below:

1. AUPR: The area under the PR curve.

2. AUROC: The area under the ROC curve.

Table 8. Performance comparison of sCoIn and aCoIn approaches with their extensions

Algorithm	Only TF	All
aCoIn	23.4365	23.3690
aCoIn-Z1	8.8948	9.0944
aCoIn-Z2	14.2170	14.3860
aCoIn	23.4352	23.3614
aCoIn-Z1	8.7881	8.8662
aCoIn-Z2	14.0426	14.3130

3. AUPR p-value: The probability that a given or larger AUPR is obtained by random ordering of the potential network edges.

4. AUROC p-value: The probability that a given or larger AUROC is obtained by random ordering of the potential network edges.

An overall score defined as $-0.5 * \log(P_1 P_2)$ was used to evaluate the predictions for the five networks, where P_1 and P_2 are respectively the geometric means of AUPR p-values and AUROC p-values taken over the five networks.

7.3.2 Performance Analysis

The aCoIn-e and sCoIn-e approaches were compared with the sCoIn and aCoIn approaches. The preliminary results indicate that the performance of these approaches were comparable. The DREAM scores of the approaches are given in Table 8. The second column of the table gives the scores for sCoIn and aCoIn approaches whereas the third column gives the scores for sCoIn-e and aCoIn-e approaches. As it was observed, that the extended versions of sCoIn and aCoIn do not significantly improve the performance of the original versions of these algorithms that consider lesser number of parent combinations and hence is computationally more efficient, it

might be worthwhile to only consider the original versions for inferring genome-scale networks in the future.

7.4 Conclusions

Novel inference approaches were discussed in this chapter wherein edges were scored based on complex structures. These approaches consistently outperformed existing popular state of the art methods. Works discussed in this chapter were published in [75, 76].

CHAPTER 8

CONSENSUS NETWORK GENERATION

There is no gold standard amongst different GRN inference methods as the accuracy of the networks inferred by these algorithms fluctuate with different data sets. As an example, the algorithm published by Yip et.al. [59], which won the DREAM 3 Network Inference Challenge did not perform well in the DREAM 4 Network Inference Challenge where different networks and data sets were provided.

Such analysis of the inference accuracy of different methods was first performed by the DREAM challenge organizers [50, 51, 52]. In these works, the authors evaluated the resulting networks of different inference methods by principal-component analysis [77]. The evaluation showed four clusters of inference methods, and these clusters coincide with the major categories of inference methods. The connectivity pattern analysis for different categories of network inference methods was also studied [78]. It was observed that feed-forward motifs were recovered reliably by Information Theoretic methods and correlation methods. However, Bayesian network and regression methods performed worse at this task. It was also observed that knock-out data-based experiments were most effective in recovering independent edges in a network.

Based on the above observation that network inference methods have complimentary advantages and limitations, the authors in [78] suggested that combining the results of multiple inference methods could be a good strategy for building networks with higher confidence. While there are various approaches for integrating networks [78, 79, 80], the authors suggested the use of the average rank method. In this method, the network interactions are re-scored according to their average rank

across all methods.

8.1 Consensus Network Inference Tool

As building a consensus network involves integrating multiple inference methods, developing an automated pipeline for building consensus networks can be of immense help to the research community. An automated pipeline Consensus Network Inference Tool (CNIT) for this purpose was built and implemented as a web tool as a part of this dissertation. This tool is available at http://bnet.egr.vcu.edu/grntool/software.php. The tool was implemented on the Ubuntu 11.04 operating system [81]. Apache server [82] was used for processing web requests and the web interface was developed using PHP (Hypertext Preprocessor) [83]. A number of other components tools/components such as MATLAB, R [84], CAPTCHA [85, 86, 87] were used in the tool implementation. While MATLAB and R environments were used to implement the different network inference approaches, the CAPTCHA functionality was implemented as a security feature. Finally, a Linux cron job was created which runs tasks on a first-in first-out request basis. Once a task is processed the user is notified via e-mail in which the URL to the consensus network built over the submitted datasets by the user is provided. The original tool [78, 88] or building a consensus network is also implemented as a web-based tool and provides the following six algorithms:

1. ANOVerence (Inferring gene regulatory networks by ANOVA) [89].
2. CLR
3. GENIE3
4. Inferelator [90].
5. TIGRESS (Trustful Inference of Gene REgulation using Stability Selection)

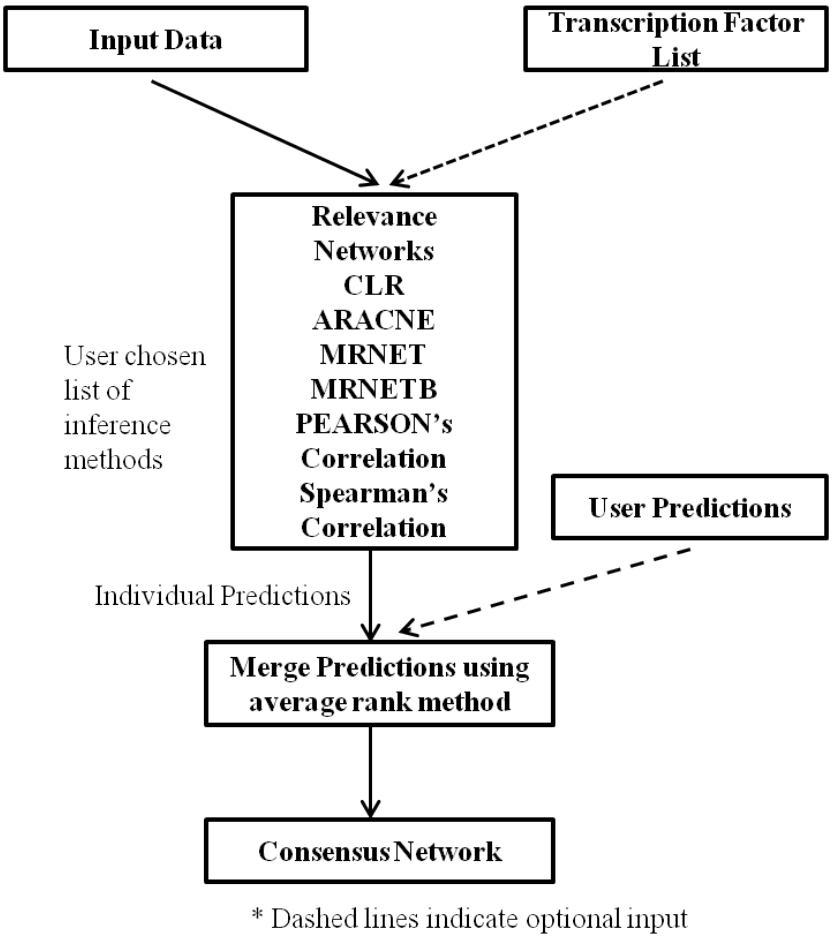


Fig. 28. Consensus network generation approach

[91].

6. Correlation [80].

However, the drawback of the existing tool is its inability to provide an option to select desired algorithms from the given list. Also, the tool does not have the ability to let the user upload predictions for their choice of algorithms. Another drawback is that the tool strictly builds transcriptional regulatory networks, thus it requires a list of transcription factors as input. This drawback limits the usage as transcription factor list is not always known; also there are cases when users might be interested in

building an undirected network in which edges exist between genes. An example of such a scenario is - when a user is interested in learning the participation of genes in signaling. CNIT gives the user more flexibility on method selection. CNIT also allows users to upload their own predictions and allows the user more flexibility in building the consensus network. Also uploading the transcription factor list is optional in CNIT. Figure 28 gives an outline of consensus network generation approach.

The following algorithms are available by default in the CNIT tool:

1. RN
2. CLR
3. ARACNE
4. MRNET [92]
5. MRNETB [93]
6. Spearmans Correlation
7. Pearsons correlation

Bioconductor R package [94] is used by the tool to implement ARACNE, MRNET and MRNETB methods. The tool allows a user to upload upto twenty network predictions. These network prediction files must have edges with their confidence scores sorted in a descending order. Figure 29 shows the CNIT web interface.

8.2 Consensus Network Inference for Western Fence Lizard (WFL) Data

Evaluation of multiple-stressor effects stemming from habitat degradation, climate change, and exposure to chemical contaminants is crucial for addressing challenges to ecological and environmental health [95, 96, 97]. Environmental pollution

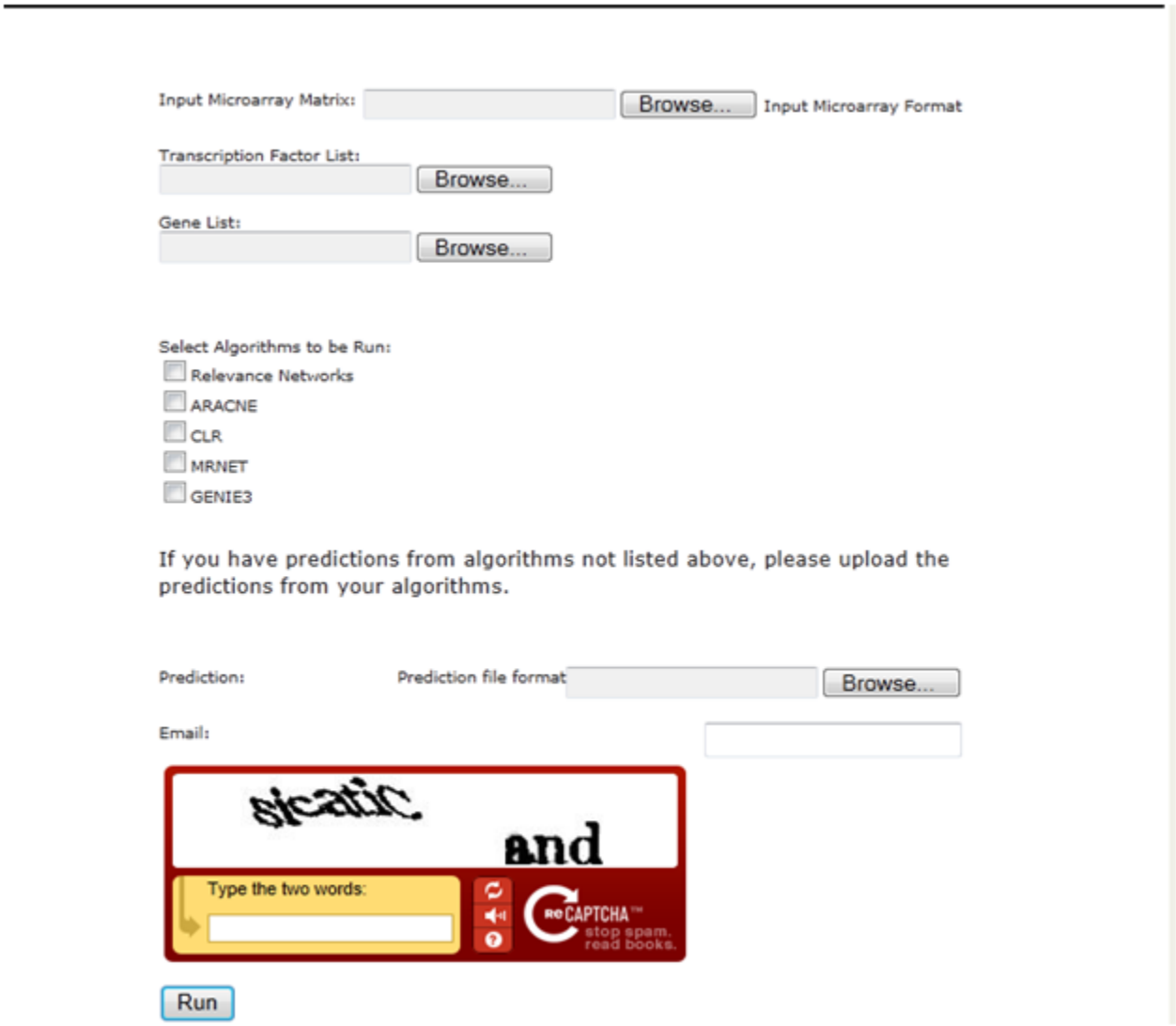


Fig. 29. CNIT web interface

has been one of the major factors in designating reptiles as threatened and endangered in the United States and around the world [95]. Western fence lizard (Sceloporus occidentalis, abbreviated as Western Fence Lizard) is a modal organism for reptiles and is an excellent model for evaluating the effects of environmental stressors on reptiles [97]. In order to study the effects on environmental stressors on Western Fence Lizard, the following individual and combines stressors were used on the WFL in the study [98].

1. Food limitation,
2. Malaria infection, and
3. Exposure to TNT

8.2.1 Experiment Design and Data Sets

Three experimental assays were conducted including: Experiment ITNT X Food Limitation, Experiment IIFood Limitation X Malaria Infection, and Experiment II-ITNT X Malaria Infection. All experiments had a 30 day duration; the malaria treatment included infected and non infected control lizards, food limitation treatments included an ad-libitum control and at least one reduced food ration and TNT exposures consisting of daily oral doses of corn oil control or a corn oil-TNT suspension at 5, 10, 20, 40 mg/kg/day. In these experiments, along with the microarray data, the datasets for various toxicological endpoints such as White Blood Cells concentration, liver weight etc. were recorded. Learning the relationships between the components of regulatory networks with the various toxicological end points is desired in this analysis. In order to do this, both the data sets were merged for all the three experiments and consensus networks were built using the web tool, CNIT.

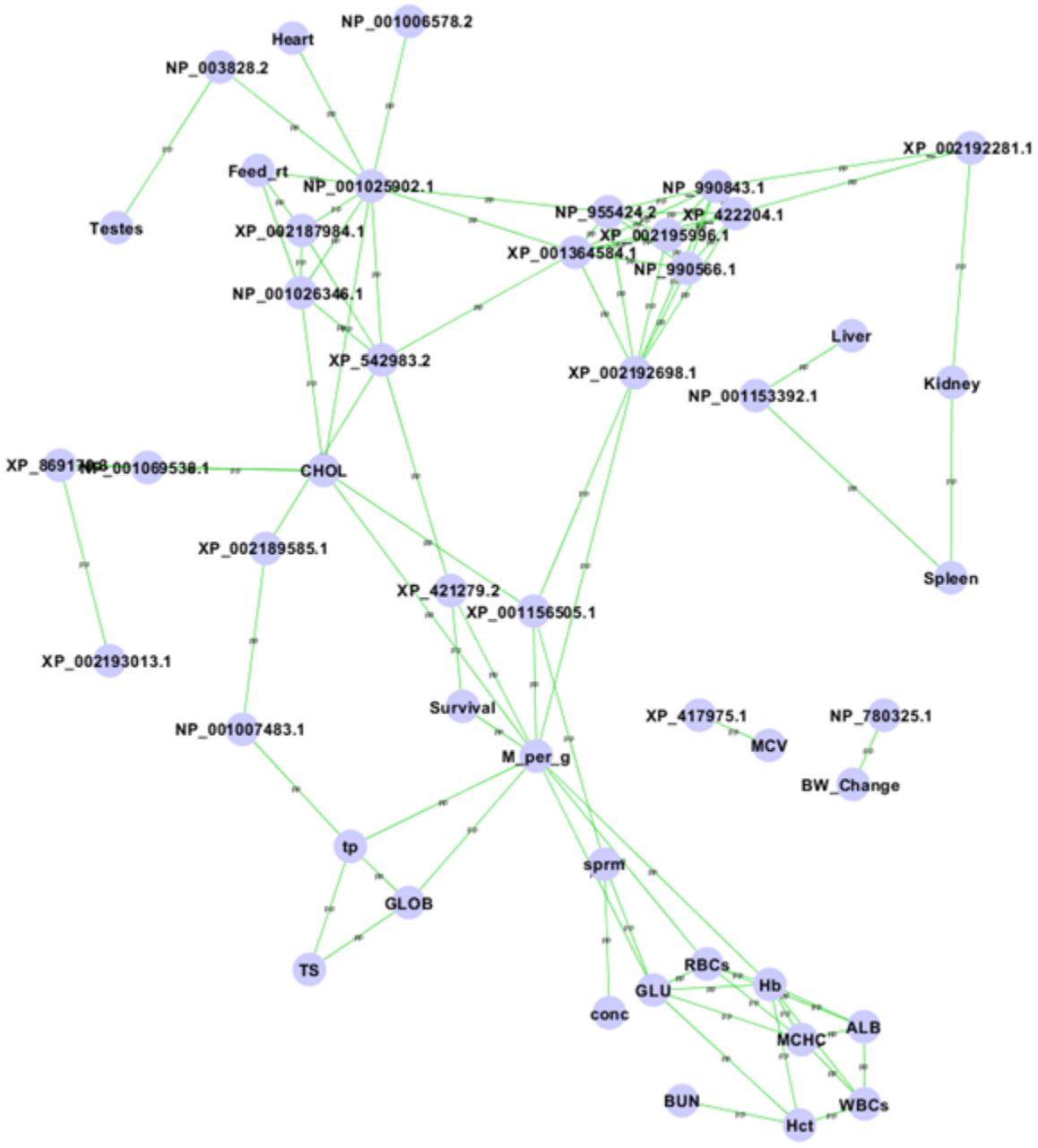


Fig. 30. Consensus network for WFL with Marial Infection

8.2.2 WFL Consensus Network

All default seven methods available in the tool were used in this cast study. The generated networks are under validation at the Environmental Research Lab, E.R.D.C, Vicksburg, MS. All the networks built were visualized using Cytoscape [99]. Cytoscape network format was desirable as it aids the biologist in their analysis. Figure 30 shows a sample generated network.

8.3 Modified Average Ranking Method for Building Consensus Networks

The average rank method to build consensus networks as originally proposed in [78] and implemented in CNIT, did not take into consideration the fact that different algorithms would capture different kinds of interactions. For example knock-out data based networks are very effective in inferring weak interactions [59] but other state-of-the-art methods are not effective in inferring weak interactions. The existing average ranking methodology computes the average rank of an edge across all algorithms by taking the mean and hence does not exploit the fact that different methods infer different kinds of relationships. In order to overcome this issue, a modified average rank method is proposed in this dissertation. In this modified average ranking method instead of computing the average rank of an edge across all algorithms, the average rank is computed across algorithms in which the score of the edge was above zero. In other words, if a particular algorithm didnt infer the edge, then that specific algorithm was not considered while computing the average rank for that edge. Table 9 and Table 10 give simple examples for computing average rank and modified average rank of two edges.

Table 9. Average rank computation

Edge	Edge Rank Algorithm 1	Edge Rank Algorithm 2	Average Rank
A to B	1	0	0.5
A to C	2	1	1.5

Table 10. Modified average rank computation

Edge	Edge Rank Algorithm 1	Edge Rank Algorithm 2	Modified Average Rank
A to B	1	0	1
A to C	2	1	1.5

8.4 Performance Comparison of Average Rank and Modified Average Rank Methods

The average rank and modified average rank methods were applied over the CLR compendium data set (details of network and data sets are discussed in Section 7.2.1.2). Seven algorithms RN, CLR, ARACNE, MRET, MRNETB, Spearmans correlation, and Pearsons correlation were used to generate individual networks. These individual networks were then merged using the average rank and modified average rank methods. The precision recall plot for the two methods is shown in Figure 31 which points to the efficacy of the proposed approach.

8.5 Multi-level Consensus Network Building Approach

While the information theoretic approaches primarily work on MI metric, other correlation metrics such as Spearmans correlation and Pearsons correlation can be used instead of the MI metric. There are two ways of building consensus networks when different algorithms may implement different scoring metrics. The possible

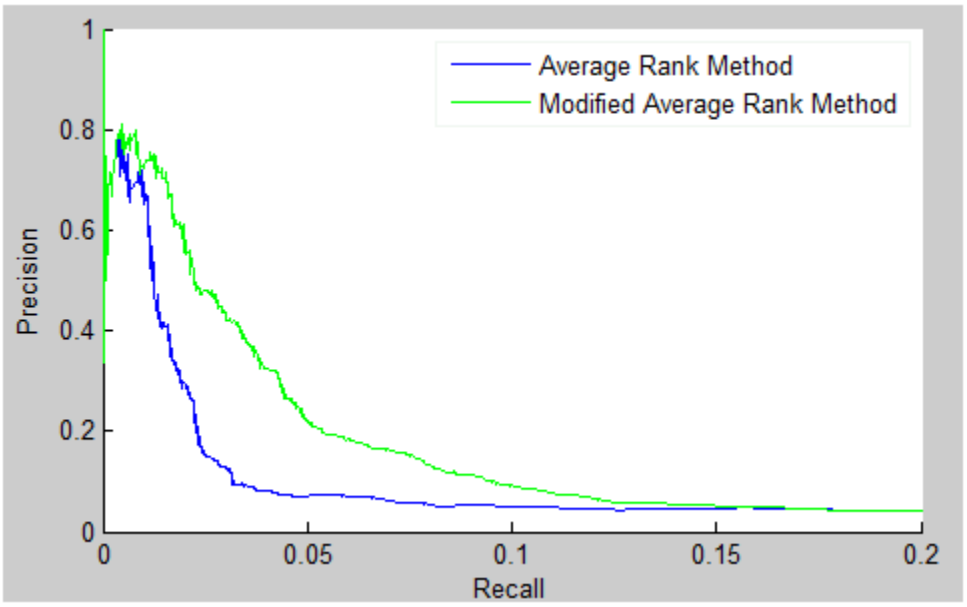


Fig. 31. Precision recall analysis of average rank and modified average rank methods

approaches are:

1. Build consensus network over independent networks generated using the same algorithm but different metrics. Each of these consensus networks generated for each algorithm can then be treated as independent networks to build the final consensus of consensus network.

2. Build consensus network over independent networks generated using the same metric but different algorithms. Each of these consensus networks generated for each metric can then be treated as independent networks to build the final consensus of consensus network.

The above two methods were applied over the CLR compendium data sets; the precision recall plots from Figure 32 indicates that the first approach performed better.

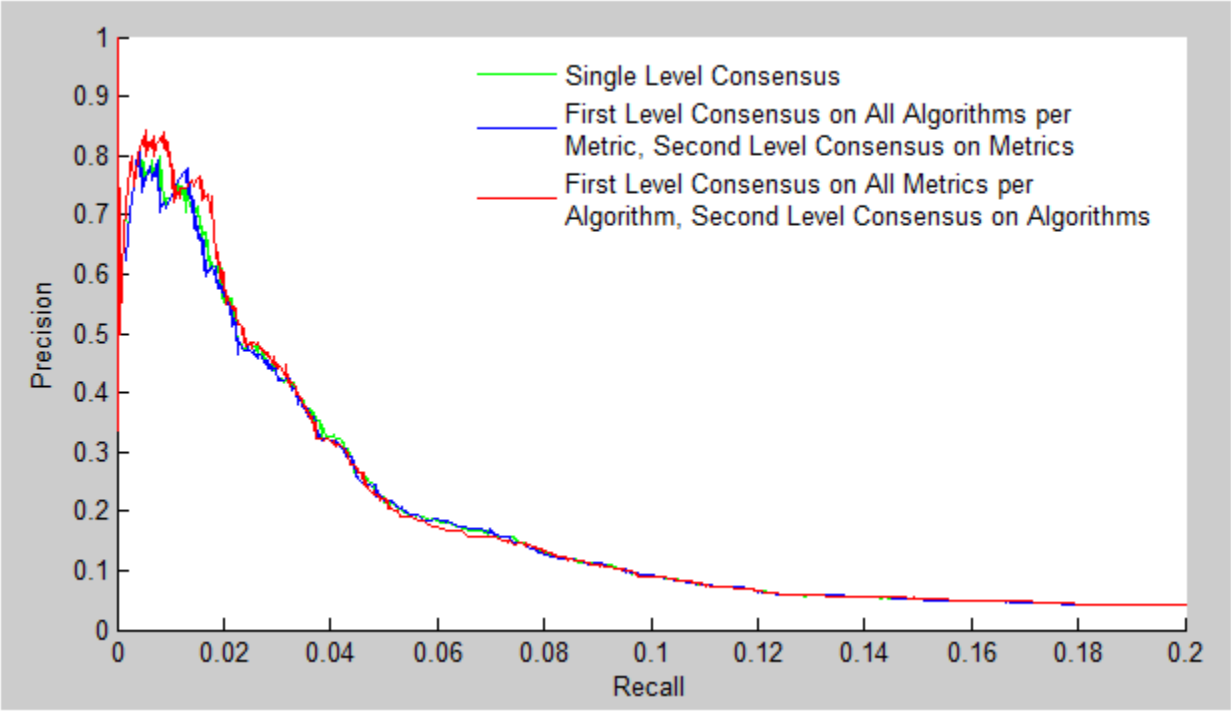


Fig. 32. Precision recall analysis of multi-level consensus network inference methods

8.6 Conclusions

Consensus network based inference is discussed in this chapter. A web interface (CNIT) to build a consensus network was developed which is much more flexible in terms of algorithm selection and usability as compared to the existing web tool. CNIT web tool was used to infer networks from the in-house toxicological data generated at environmental lab, E.R.D.C., Vicksburg, MS. As an extension to CNIT, a modified average ranking method to score edges more accurately and schemes to generate the consensus of consensus networks were proposed to achieve higher inference accuracy.

FUTURE WORK

1. The aCoIn and sCoIn approaches were our first step towards structure based inference. These approaches primarily consider three nodes structures for inference. However we have shown that incorporating higher order structures improves the inference accuracy. While multiple kinds of motifs exist for each complex structure, we aim at inferring motifs of interest such as the feed forward motifs in future.

2. A time lag in biological networks is a continuous and stochastic quantity, in this dissertation so far a time lag was considered to be a discrete quantity [57, 58]. Our efforts in modeling time lags are under progress.

3. We have built a web tool for inferring consensus networks. The existing approach towards building consensus network is based on ranking edges from independent predictions. We intend to build an approach that would score motifs instead of independent edges in future.

Appendix A

ABBREVIATIONS

ARACNE	Algorithm for the Reconstruction of Accurate Cellular Networks
CLR	Context Likelihood of Relatedness
CMI	Conditional Mutual Information
CNIT	Consensus Network Inference Tool
DBN	Dynamic Bayesian Networks
DEM	Differential Equations Methods
GENIE	Gene Network Inference with Ensemble of Trees
KNN	Kth Nearest Neighbor
LM	Linear Methods
MDL	Minimum Description Length
MI	Mutual Information
NNM	Neural Network Methods
P	Precision
PBN	Probabilistic Boolean Networks
PMDL	Predictive Minimum Description Length
R	Recall
RN	Relevance Networks
TIGRESS	Trustful Inference of Gene REgulation using Stability Selection
TLCMI	Time Lagged Conditional Mutual Information
TLMI	Time Lagged Mutual Information
WFL	Western Fence Lizard

REFERENCES

[1] T.I. Lee et al. "Transcriptional regulatory networks in Saccharomyces cerevisiae". In: *Science Signalling* 298.5594 (2002), p. 799.

[2] M. M. Babu, B. Lang, and L. Aravind. "Methods to reconstruct and compare transcriptional regulatory networks". In: *Methods Mol. Biol.* 541 (2009), pp. 163–180.

[3] M. Madan Babu, S.A. Teichmann, and L. Aravind. "Evolutionary dynamics of prokaryotic transcriptional regulatory networks". In: *Journal of molecular biology* 358.2 (2006), pp. 614–633.

[4] H. Yu et al. "Annotation transfer between genomes: protein–protein interologs and protein–DNA regulogs". In: *Genome research* 14.6 (2004), pp. 1107–1118.

[5] I. Lozada-Chavez, S.C. Janga, and J. Collado-Vides. "Bacterial regulatory networks are extremely flexible in evolution". In: *Nucleic acids research* 34.12 (2006), pp. 3434–3445.

[6] W.B.L. Alkema, B. Lenhard, and W.W. Wasserman. "Regulog analysis: detection of conserved regulatory networks across bacteria: application to Staphylococcus aureus". In: *Genome research* 14.7 (2004), pp. 1362–1373.

[7] T. Wang and G.D. Stormo. "Identifying the conserved network of cis-regulatory sites of a eukaryotic genome". In: *Proceedings of the National Academy of Sciences of the United States of America* 102.48 (2005), pp. 17400–17405.

[8] D.A. Rodionov et al. "Reconstruction of regulatory and metabolic pathways in metal-reducing delta-proteobacteria". In: *Genome Biol* 5.11 (2004), R90.

[9] V. Chaitankar et al. "A Scalable Information Theory Based Gene Regulatory Network Inference Method from Time Series and Knock-Out Data". In: *BICoB*. ISCA, 2011, pp. 74–79.

[10] Kevin Murphy and Saira Mian. *Modelling gene expression data using dynamic bayesian networks*. Tech. rep. 1999.

[11] S. Imoto, T. Goto, and S. Miyano. "Estimation of genetic networks and functional structures between genes by using Bayesian networks and nonparametric regression". In: *Pac Symp Biocomput* (2002), pp. 175–186.

[12] M. Zou and S. D. Conzen. "A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data". In: *Bioinformatics* 21.1 (2005), pp. 71–79.

[13] I. Shmulevich et al. "Probabilistic Boolean Networks: a rule-based uncertainty model for gene regulatory networks". In: *Bioinformatics* 18.2 (2002), pp. 261–274.

[14] Ilya Shmulevich, Edward R. Dougherty, and Wei Zhang. "From Boolean to Probabilistic Boolean Networks as Models of Genetic Regulatory Networks". In: *Proc. IEEE*. 2002, pp. 1778–1792.

[15] M. Madan Babu. *Chapter 11 An Introduction to Microarray Data Analysis*.

[16] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. New York, NY, USA: Wiley-Interscience, 1991. ISBN: 0-471-06259-6.

[17] Wentao Zhao, Erchin Serpedin, and Edward R. Dougherty. "Inferring Connectivity of Genetic Regulatory Networks Using Information-Theoretic Criteria". In: *IEEE/ACM Trans. Comput. Biol. Bioinformatics* 5.2 (Apr. 2008),

pp. 262–274. ISSN: 1545-5963. DOI: 10.1109/TCBB.2007.1067. URL: http://dx.doi.org/10.1109/TCBB.2007.1067.

[18] Xin Zhang, Chitta Baral, and Seungchan Kim. "An algorithm to learn causal relations between genes from steady state data: simulation and its application to melanoma dataset". In: *Proceedings of the 10th conference on Artificial Intelligence in Medicine*. AIME'05. Aberdeen, UK: Springer-Verlag, 2005, pp. 524–534. ISBN: 3-540-27831-1, 978-3-540-27831-3. DOI: 10.1007/11527770_69. URL: http://dx.doi.org/10.1007/11527770_69.

[19] S. A. Kauffman. "Metabolic stability and epigenesis in randomly constructed genetic nets". In: *J. Theor. Biol.* 22.3 (1969), pp. 437–467.

[20] S.A. Kauffman. *The origins of order: Self-organization and selection in evolution*. Oxford University Press, USA, 1993.

[21] L. Glass and S. A. Kauffman. "The logical analysis of continuous, non-linear biochemical control networks". In: *J. Theor. Biol.* 39.1 (1973), pp. 103–129.

[22] N. Friedman et al. "Using Bayesian networks to analyze expression data". In: *J. Comput. Biol.* 7.3-4 (2000), pp. 601–620.

[23] A. J. Hartemink et al. "Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks". In: *Pac Symp Biocomput* (2001), pp. 422–433.

[24] T. Chen, H. L. He, and G. M. Church. "Modeling gene expression with differential equations". In: *Pac Symp Biocomput* (1999), pp. 29–40.

[25] T. Mestl, E. Plahte, and S.W. Omholt. "A mathematical framework for describing and analysing gene regulatory networks". In: *Journal of Theoretical Biology* 176.2 (1995), pp. 291–300.

[26] P. D'haeseleer et al. "Linear modeling of mRNA expression levels during CNS development and injury". In: *Pac Symp Biocomput* (1999), pp. 41–52.

[27] E. P. van Someren, L. F. Wessels, and M. J. Reinders. "Linear modeling of genetic networks from experimental data". In: *Proc Int Conf Intell Syst Mol Biol* 8 (2000), pp. 355–366.

[28] J. Vohradsky. "Neural model of the genetic network". In: *J. Biol. Chem.* 276.39 (2001), pp. 36168–36173.

[29] D. C. Weaver, C. T. Workman, and G. D. Stormo. "Modeling regulatory networks with weight matrices". In: *Pac Symp Biocomput* (1999), pp. 112–123.

[30] A. J. Butte and I. S. Kohane. "Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements". In: *Pac Symp Biocomput* (2000), pp. 418–429.

[31] M. B. Eisen et al. "Cluster analysis and display of genome-wide expression patterns". In: *Proc. Natl. Acad. Sci. U.S.A.* 95.25 (1998), pp. 14863–14868.

[32] A.A. Margolin et al. "ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context". In: *BMC bioinformatics* 7.Suppl 1 (2006), S7.

[33] A.A. Margolin et al. "Reverse engineering cellular networks". In: *Nature Protocols* 1.2 (2006), pp. 662–671.

[34] S. Liang, S. Fuhrman, and R. Somogyi. "Reveal, a general reverse engineering algorithm for inference of genetic network architectures". In: *Pac Symp Biocomput* (1998), pp. 18–29.

[35] J. J. Faith et al. "Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles". In: *PLoS Biol.* 5.1 (2007), e8.

[36] P. Dhaeseleer, S. Liang, and R. Somogyi. "Genetic network inference: from co-expression clustering to reverse engineering". In: *Bioinformatics* 16.8 (2000), pp. 707–726.

[37] V. Chaitankar et al. "A novel gene network inference algorithm using predictive minimum description length approach". In: *BMC Systems Biology* 4.Suppl 1 (2010), S7.

[38] W. Zhao, E. Serpedin, and E. R. Dougherty. "Inferring gene regulatory networks from time series data using the minimum description length principle". In: *Bioinformatics* 22.17 (2006), pp. 2129–2135.

[39] J. Rissanen. "Modeling by shortest data description". In: *Automatica* 14.5 (1978), pp. 465–471.

[40] J. Rissanen. "An introduction to the MDL principle". In: *Available online at www. mdl-research. org* (2005).

[41] P.D. Grünwald, I.J. Myung, and M.A. Pitt. *Advances in minimum description length: Theory and applications*. MIT press, 2005.

[42] M.H. Hansen and B. Yu. "Model selection and the principle of minimum description length". In: *Journal of the American Statistical Association* 96.454 (2001), pp. 746–774.

[43] J. Dougherty, I. Tabus, and J. Astola. "Inference of gene regulatory networks based on a universal minimum description length". In: *EURASIP Journal on Bioinformatics and Systems Biology* 2008 (2008), p. 5.

[44] J. Rissanen. "Universal coding, information, prediction, and estimation". In: *Information Theory, IEEE Transactions on* 30.4 (1984), pp. 629–636.

[45] A.P. Dawid. "Present position and potential developments: Some personal views: Statistical theory: The prequential approach". In: *Journal of the Royal Statistical Society. Series A (General)* (1984), pp. 278–292.

[46] P.T. Spellman et al. "Comprehensive identification of cell cycle–regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization". In: *Molecular biology of the cell* 9.12 (1998), pp. 3273–3297.

[47] M. Kanehisa et al. "KEGG for linking genomes to life and the environment". In: *Nucleic acids research* 36.suppl 1 (2008), pp. D480–D484.

[48] M. Kanehisa et al. "From genomics to chemical genomics: new developments in KEGG". In: *Nucleic acids research* 34.suppl 1 (2006), pp. D354–D357.

[49] H. Ogata et al. "KEGG: Kyoto encyclopedia of genes and genomes". In: *Nucleic acids research* 27.1 (1999), pp. 29–34.

[50] D. Marbach et al. "Revealing strengths and weaknesses of methods for gene network inference". In: *Proceedings of the National Academy of Sciences* 107.14 (2010), pp. 6286–6291.

[51] R.J. Prill et al. "Towards a rigorous assessment of systems biology models: the DREAM3 challenges". In: *PloS one* 5.2 (2010), e9202.

[52] D. Marbach et al. "Generating realistic in silico gene networks for performance assessment of reverse engineering methods". In: *Journal of Computational Biology* 16.2 (2009), pp. 229–239.

[53] V. Chaitankar et al. "Predictive Minimum Description Length Principle Approach to Inferring Gene Regulatory Networks". In: *Software Tools and Algorithms for Biological Systems* (2011), pp. 37–43.

[54] V. Chaitankar et al. "Gene regulatory network inference using predictive minimum description length principle and conditional mutual information". In: *Bioinformatics, Systems Biology and Intelligent Computing, 2009. IJCBS'09. International Joint Conference on*. IEEE. 2009, pp. 487–490.

[55] A. Rao, A.O. Hero, J.D. Engel, et al. "Inference of biologically relevant gene influence networks using the directed information criterion". In: *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*. Vol. 2. IEEE. 2006, pp. II–II.

[56] V. Chaitankar et al. "Effects of cDNA microarray time-series data size on gene regulatory network inference accuracy". In: *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*. ACM. 2010, pp. 410–413.

[57] V. Chaitankar et al. "Time lagged information theoretic approaches to the reverse engineering of gene regulatory networks". In: *BMC bioinformatics* 11.Suppl 6 (2010), S19.

[58] V. Chaitankar et al. "Poster: Gene regulatory network inference using time lagged context likelihood of relatedness". In: *Computational Advances in Bio and Medical Sciences (ICCABS), 2011 IEEE 1st International Conference on*. IEEE. 2011, pp. 236–236.

[59] K.Y. Yip et al. "Improved reconstruction of in silico gene regulatory networks by integrating knockout and perturbation data". In: *PloS one* 5.1 (2010), e8121.

[60] F. Geier, J. Timmer, and C. Fleck. "Reconstructing gene-regulatory networks from time series, knock-out data, and prior knowledge". In: *BMC Systems Biology* 1.1 (2007), p. 11.

[61] S. Gama-Castro et al. "RegulonDB (version 6.0): gene regulation model of Escherichia coli K-12 beyond transcription, active (experimental) annotated promoters and Textpresso navigation". In: *Nucleic acids research* 36.suppl 1 (2008), pp. D120–D124.

[62] M.U. Guide. "The mathworks". In: *Inc., Natick, MA* 5 (1998).

[63] S.K. Kummerfeld and S.A. Teichmann. "DBD: a transcription factor prediction database". In: *Nucleic acids research* 34.suppl 1 (2006), pp. D74–D81.

[64] D. Wilson et al. "DBD—-taxonomically broad transcription factor predictions: new content and functionality". In: *Nucleic acids research* 36.suppl 1 (2008), pp. D88–D92.

[65] V. Chaitankar et al. "sREVEAL: Scalable extensions of REVEAL towards regulatory network inference". In: *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*. IEEE. 2011, pp. 1365–1370.

[66] R. Milo et al. "Network motifs: simple building blocks of complex networks". In: *Science Signalling* 298.5594 (2002), p. 824.

[67] S.S. Shen-Orr et al. "Network motifs in the transcriptional regulation network of Escherichia coli". In: *Nature genetics* 31.1 (2002), pp. 64–68.

[68] U. Alon. "Network motifs: theory and experimental approaches". In: *Nature Reviews Genetics* 8.6 (2007), pp. 450–461.

[69] A. Kraskov, H. Stögbauer, and P. Grassberger. "Estimating mutual information". In: *Physical Review E* 69.6 (2004), p. 066138.

[70] R. Moddemeijer. "On estimation of entropy and mutual information of continuous distributions". In: *Signal Processing* 16.3 (1989), pp. 233–248.

[71] C.O. Daub et al. "Estimating mutual information using B-spline functions–an improved similarity measure for analysing gene expression data". In: *BMC bioinformatics* 5.1 (2004), p. 118.

[72] Y.I. Moon, B. Rajagopalan, and U. Lall. "Estimation of mutual information using kernel density estimators". In: *Physical Review E* 52.3 (1995), p. 2318.

[73] B.W. Silverman. *Density estimation for statistics and data analysis*. Vol. 26. Chapman & Hall/CRC, 1986.

[74] R. Steuer et al. "The mutual information: detecting and evaluating dependencies between variables". In: *Bioinformatics* 18.suppl 2 (2002), S231–S240.

[75] V. Chaitankar et al. "Genome scale inference of transcriptional regulatory networks using mutual information on complex interactions". In: *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine*. ACM. 2012, pp. 643–648.

[76] V. Chaitankar et al. "sCoIn: A Scoring algorithm based on COmplex INteractions for reverse engineering regulatory networks". In: *Bioinformatics and Bioengineering (BIBE), IEEE 12th International Conference on*. To appear. IEEE. 2012.

[77] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2005.

[78] D. Marbach, C. Mattiussi, and D. Floreano. "Combining Multiple Results of a Reverse-Engineering Algorithm: Application to the DREAM Five-Gene Network Challenge". In: *Annals of the New York Academy of Sciences* 1158.1 (2009), pp. 102–113.

[79] D. Marbach et al. “Wisdom of crowds for robust gene network inference”. In: *Nature Methods* (2012).

[80] R.J. Prill et al. “Crowdsourcing network inference: the DREAM predictive signaling network challenge”. In: *Science Signalling* 4.189 (2011), mr7.

[81] W. Von Hagen. *Ubuntu Linux Bible: Featuring Ubuntu 10.04 LTS*. Vol. 701. Wiley, 2010.

[82] R.B. Bloom and B. Foreword By-Behlendorf. *Apache Server 2.0: The Complete Reference*. Osborne/McGraw-Hill, 2002.

[83] D. Sklar and A. Trachtenberg. *PHP cookbook*. O’Reilly Media, Incorporated, 2002.

[84] P. Teetor. *R cookbook*. O’Reilly Media, Incorporated, 2011.

[85] L. Von Ahn et al. “CAPTCHA: Using hard AI problems for security”. In: *Advances in CryptologyEUROCRYPT 2003* (2003), pp. 646–646.

[86] L. Von Ahn, M. Blum, and J. Langford. “Telling humans and computers apart automatically”. In: *Communications of the ACM* 47.2 (2004), pp. 56–60.

[87] L. Von Ahn et al. “recaptcha: Human-based character recognition via web security measures”. In: *Science* 321.5895 (2008), pp. 1465–1468.

[88] M. Reich et al. “GenePattern 2.0”. In: *Nature genetics* 38.5 (2006), pp. 500–501.

[89] R. Küffner et al. “Inferring gene regulatory networks by ANOVA”. In: *Bioinformatics* 28.10 (2012), pp. 1376–1382.

[90] A. Greenfield et al. “DREAM4: Combining genetic and dynamic information to identify biological networks and dynamical models”. In: *PloS one* 5.10 (2010), e13397.

[91] A.C. Haury et al. "TIGRESS: trustful inference of gene regulation using stability selection". In: *arXiv preprint arXiv:1205.1181* (2012).

[92] P.E. Meyer et al. "Information-theoretic inference of large transcriptional regulatory networks". In: *EURASIP Journal on Bioinformatics and Systems Biology* 2007 (2007).

[93] P.E. Meyer et al. *Information-Theoretic Inference of Gene Networks Using Backward Elimination*. 2010.

[94] P.E. Meyer, F. Lafitte, and G. Bontempi. "minet: AR/Bioconductor package for inferring large transcriptional networks using mutual information". In: *BMC bioinformatics* 9.1 (2008), p. 461.

[95] J.W. GIBBON et al. "The global decline of reptiles, déjà vu amphibians". In: *BioScience* 50.8 (2000), pp. 653–666.

[96] L.G. Talent et al. "Evaluation of western fence lizards (Sceloporus occidentals) and eastern fence lizards (Sceloporus undulatus) as laboratory reptile models for toxicological investigations". In: *Environmental toxicology and chemistry* 21.5 (2002), pp. 899–905.

[97] RR Holem, W.A. Hopkins, and L.G. Talent. "Effect of acute exposure to malathion and lead on sprint performance of the western fence lizard (Sceloporus occidentalis)". In: *Archives of environmental contamination and toxicology* 51.1 (2006), pp. 111–116.

[98] C.A. McFarland et al. "Multiple environmental stressors elicit complex interactive effects in the western fence lizard (Sceloporus occidentalis)". In: *Ecotoxicology* (2012), pp. 1–19.

[99] M. Neuditschko, M.S. Khatkar, and H.W. Raadsma. "NetView: A High-Definition Network-Visualization Approach to Detect Fine-Scale Population Structures from Genome-Wide Patterns of Variation". In: *PloS one* 7.10 (2012), e48375.

VITA

Vijender Chaitankar was born in Hyderabad, India. He graduated from Bharatiya Vidya Bhavan Public School, Ramachandrapuram, Hyderabad, India in 1998. He received his Bachelor of Engineering in Computer Science from Dr. Baba Saheb Ambedkar Marathwada University, Aurangabad, India in 2004. He received his Master of Sciences in Computer Science from The University of Southern Mississippi, Hattiesburg, MS in 2006. He is the recipient of best poster presentation award at the MCBIOS 2010 conference held in Jonesboro, AK. He is also the recipient of three NSF-sponsored travel awards for MCBIOS 2010, ACM-BCB 2010, and ICCABS 2011 conferences.