Inference of gene networks from temporal gene expression profiles

M. Bansal and D. di Bernardo

Abstract: Genes interact with each other in complex networks that enable the processing of information and the metabolism of nutrients inside the cell. A novel inference algorithm based on linear ordinary differential equations is proposed. The algorithm can infer the local network of gene–gene interactions surrounding a gene of interest from time-series gene expression profiles. The performance of the algorithm has been tested on in silico simulated gene expression data and on a nine gene subnetwork part of the DNA-damage response pathway (SOS pathway) in the bacteria *Escherichia coli*. This approach can infer regulatory interactions even when only a small number of measurements is available.

1 Introduction

Genes interact with each other in complex networks that enable the processing of information and the metabolism of nutrients inside the cell. Identification of gene regulatory networks is essential for understanding cellular functions. Understanding such networks requires simultaneous measurements of thousands of molecules in the cells. Gene expression microarrays are a recent high-throughput technology that allows measuring quantitatively the expression level of each of the thousands of genes in a cell at a specific condition and time. Molecular biology is therefore rapidly evolving into a quantitative science, and as such, it is increasingly relying on engineering, applied physics and mathematics to make sense of quantitative high-throughput data.

In molecular, biology experiments often involve the perturbation of gene of interest (i.e. overexpression or knockdown) and the measurement of the changes in expression level of all the genes at different time points following the perturbation. The focus of computational systems biology is to infer, or 'reverse-engineer', from this gene expression microarray data, the regulatory interactions among genes.

Recently, there have been many attempts to reverse engineer gene networks from gene expression data that are based on different mathematical models. These models are mainly classified in four broad classes: (i) Boolean networks, (ii) Bayesian networks, (iii) Information-theoretic approaches and (iv) Differential equations.

Boolean networks [1–3] offer a binary, discrete-time description of a system. This model assumes that each gene is in two possible states, expressed or not expressed. Interactions among genes are modelled through Boolean logic functions. These models are quite simple and can be easily applied but the underlying assumption seems very unrealistic, in particular, modelling genes as discrete entities assuming one of only two states. Also the amount of data needed to infer the network increases as 2^n, where n is the number of genes.

A Bayesian network [4, 5] is a graphical model for probabilistic relationships among a set of random variables $\boldsymbol{X}_i$, where $i = 1, \ldots, n$. These relationships are encoded in the structure of a directed acyclic graph G, whose vertices (or nodes) are the random variables $\boldsymbol{X}_i$. The relationships between the variables are described by a joint probability distribution $P(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)$ that is consistent with the independence assertions embedded in the graph G. These models are good to capture the stochastic nature of gene regulatory network and they can also deal with noisy measurements. Learning Bayesian networks from data is NP hard; hence heuristic search methods have to be used, which do not guarantee convergence to the global optimal solution.

Information-theoretic approaches [6–8], use a generalisation of pairwise correlation coefficient called mutual information (MI), to compare expression profiles from a set of microarrays. For each pair of genes (i, j), their MI_{ij} is computed and the edge is set to 0 or 1 depending on a significant threshold to which MI_{ij} is compared. MI measures the degree of independence between two genes. The computation of MI requires each data point, that is, each experiment, to be statistically independent from others. Thus it can only deal with steady-state gene expression data set and not with time series.

Reverse-engineering algorithms based on ordinary differential equations [9–13] provide a deterministic approach which is based on relating changes in gene transcript concentration to each other. To reverse-engineer a network using ODE requires a choice of functional form and nonlinear functions can lead to exponential rise in the unknown parameters to be estimated. Higher noise in the microarray data limits their application to make only qualitative statement and not quantitative statement about the underlying network.

Current methods [14] use steady-state expression measurements following multiple perturbations and achieve a very good performance. However, reverse engineering from time-series experiments is still problematic [14] because of the limited number of datapoints that is possible to collect. Only few of the methods have shown some

doi:10.1049/iet-syb:20060079

Paper first received 30th November 2006 and in revised form 27th July 2007

The authors are with Telethon Institute of Genetics and Medicine, Via P.Castellino 111, Naples 80131, Italy and also with European School of Molecular Medicine (SEMM), Naples, Italy

E-mail: bansal@tigem.it

success with time series data [4, 5, 13]. Yu *et al.* [5] recovered the network among 20 nodes in the network but to reach a good accuracy they used number of datapoints order of magnitude higher than the number of nodes in network. van Someren *et al.* [13] used only 11 datapoints to recover the network among 101 genes but their validation was limited to the co-citation of the genes in recovered connection in literature. Dojer *et al.* [4] successfully recovered the network among 10 genes but they used multiple perturbation experiments with many time points in each experiment.

In this work, we propose a novel method, based on linear differential equations, able to achieve a good performance even for a short time-series gene expression profile coming from a single perturbation experiment.

2 Methods

2.1 Network model description

Our model is based on relating the changes in gene transcript concentration to each other and to the external perturbation (Fig. 1). By external perturbation we mean an experimental treatment that can alter the transcription rate of the genes in the cell. An example of perturbation is the treatment with a chemical compound (i.e. a drug), or a genetic perturbation involving overexpression or downregulation of particular genes. We used a linear system of ordinary differential equations [15] to represent the rate of synthesis of a transcript as a function of the concentrations of every other transcript in a cell and the external perturbation

$$\dot{x}_i(t_k) = \sum_{j=1}^{N} a_{ij} x_j(t_k) + \sum_{l=1}^{P} b_{il} u_l(t_k) \tag{1}$$

where $i = 1, \ldots, N$; $k = 1, \ldots, M$; $x_i(t_k)$ is concentration of transcript i measured at time t_k; $\dot{x}_i(t_k)$ is the rate of change of concentration of gene transcript i at time t_k, that is the first derivative of the mRNA concentration of gene i measured at time t_k; a_{ij} represents the influence of gene j on gene i with a positive, zero or negative sign indicating activation, no interaction and repression, respectively; b_{il} represents the effect of lth perturbation on x_i and $u_l(t_k)$ represents the lth external perturbation at time t_k.

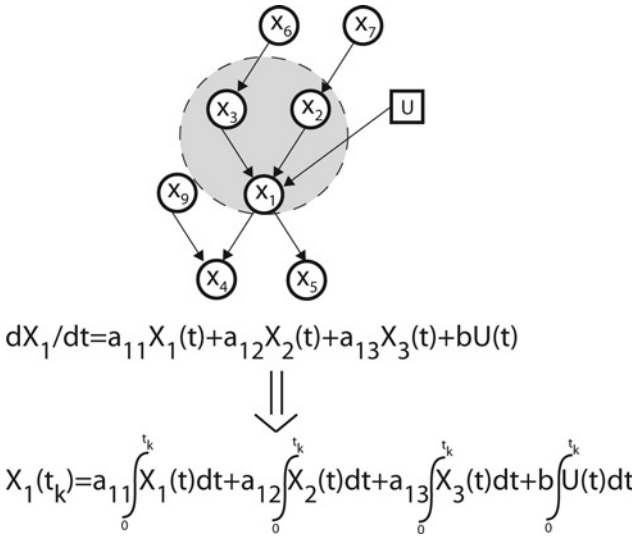

Fig. 1 *Methodology*

We use a deterministic approach based on linear differential equations where the rate of transcription of gene $\boldsymbol{X}_1$ is a function of the expression of its direct causal regulators and an external perturbation ($\boldsymbol{U}$).

Equation (1) at time t_k can be rewritten in a more compact form using matrix notation

$$\dot{\boldsymbol{X}}(t_k) = \boldsymbol{A}\boldsymbol{X}(t_k) + \boldsymbol{B}\boldsymbol{U}(t_k) \quad k = 1, \ldots, M \tag{2}$$

where $\boldsymbol{X}(t_k)$ is an $N \times 1$ vector of mRNA concentration of N genes at time t_k, $\dot{\boldsymbol{X}}(t_k)$ is an $N \times 1$ vector of the first derivatives of $\boldsymbol{X}$ at time t_k, $\boldsymbol{A}$ is an $N \times N$ connectivity matrix, composed of elements a_{ij}, $\boldsymbol{B}$ is a $N \times P$ matrix representing the effect of P perturbations on N genes, $\boldsymbol{U}(t_k)$ is $P \times 1$ vector representing P perturbations at time t_k and M is the number of time points t_k in the time-series experiment. The unknown connectivity matrix $\boldsymbol{A}$ is what we want to learn from the data. Element (i, l) of $\boldsymbol{B}$ will be different from zero if the ith gene is a direct target of the lth perturbation.

2.2 Algorithm

To solve (2), we need to compute the derivative of $\boldsymbol{X}(t_k)$. Since gene expression measurements are noisy, computing derivatives is not appropriate since this will amplify the noise in the data. To avoid this problem, we integrated (2)

$$\int_0^{t_k} \dot{\boldsymbol{X}}(t)\,\mathrm{d}t = \boldsymbol{A}\int_0^{t_k} \boldsymbol{X}(t)\,\mathrm{d}t + \boldsymbol{B}\int_0^{t_k} \boldsymbol{U}(t)\,\mathrm{d}t, \quad k = 1, \ldots, M \tag{3}$$

or

$$\boldsymbol{X}(t_k) = \boldsymbol{A}\int_0^{t_k} \boldsymbol{X}(t)\,\mathrm{d}t + \boldsymbol{B}\int_0^{t_k} \boldsymbol{U}(t)\,\mathrm{d}t, \quad k = 1, \ldots, M \tag{4}$$

where we assumed null initial conditions [$\boldsymbol{X}(0) = 0$].

Gene expression levels are measured at discrete time points, therefore assuming a ZOH (zero-order-hold) model we can rewrite the (4) as

$$\boldsymbol{X}(t_k) = \boldsymbol{A}\Sigma_{i=1}^{k}\boldsymbol{X}(t_i) * \delta t + \boldsymbol{B}\Sigma_{i=1}^{k}\boldsymbol{U}(t_i) * \delta t, \quad k = 1, \ldots, M \tag{5}$$

where δt is the sampling interval. This equation can be written as

$$\boldsymbol{Y}(t_k) = \boldsymbol{A} * \boldsymbol{H}(t_k) \tag{6}$$

where $\boldsymbol{Y}(t_k) = \boldsymbol{X}(t_k) - \boldsymbol{B}\Sigma_{i=1}^{k}\boldsymbol{U}(t_i) * \delta t$ and $\boldsymbol{H}(t_k) = \Sigma_{i=1}^{k}\boldsymbol{X}(t_i) * \delta t$. Writing (6) in matrix form after removing time subscript yields

$$\boldsymbol{Y} = \boldsymbol{A} * \boldsymbol{H} \tag{7}$$

where $\boldsymbol{Y}$ and $\boldsymbol{H}$ are $N \times M$ dimensional. In what follows, we will use the functional expanded form for each row i and column j, $\boldsymbol{Y}_{ij} = f^{a_{ik}}(\boldsymbol{H}_j)$ which is equivalent to $\boldsymbol{Y}_{ij} = \sum_{l=1}^{k} a_{il}\boldsymbol{H}_{lj}$.

To identify the network means to compute matrix $\boldsymbol{A}$ from the available data $\boldsymbol{Y}$ and $\boldsymbol{H}$. This can be achieved by applying the pseudo-inverse method to each row of $\boldsymbol{A}$ to obtain

$$\boldsymbol{A}_i = \boldsymbol{Y}_i * \boldsymbol{H}' * (\boldsymbol{H} * \boldsymbol{H}' + \lambda_i * \boldsymbol{I})^{-1} \tag{8}$$

where $\boldsymbol{A}_i$ is the ith row of $\boldsymbol{A}$. $\boldsymbol{Y}_i$ is the ith row of $\boldsymbol{Y}$. $\boldsymbol{H}'$ is the transpose of $\boldsymbol{H}$. $\boldsymbol{I}$ is the identity matrix and λ_i is the ridge parameter. λ_i is computed as the smallest singular value of $[\boldsymbol{H}' \; \boldsymbol{Y}_i']$ and is useful to control for bad conditioning of $\boldsymbol{H}^*\boldsymbol{H}'$ [16]. Equation (8) can be solved only if $M \geq N$, that is if we have a sufficient number of data points. Usually in biology $M \leq N$.

Generally gene networks are not fully connected but sparse, that is each gene regulates only a limited number of genes [10]. Therefore we can solve (8) assuming that only K out of N parameters of row $\boldsymbol{A}_i$ are different from zero. Further assuming that $K < M$, ensures that we can always solve (8). The problem is to select which K parameters to use out of all the possible K over N combinations.

We applied a forward step-wise regression method [17] where we first considered all solutions with a single parameter $K = 1$. We then ranked the solutions according to an error (i.e. least square error) and picked the D solutions with the smallest error. For each one of the D solutions, we then consider all the possible pairwise combinations of $K = 2$ parameters that contained the previously chosen parameter. This can be repeated up to $K = M$ connections. We chose D equal to 5 as suggested in [18] as a good compromise between computation time and accuracy.

2.3 Time-series network identification using bootstrapping

At each step, we could have used the least-squared error as the function to minimise. But least-squared error may cause the computed solution to be very sensitive to noise in the data (over-fitting). Bootstrapping is a general technique to avoid overfitting. The basic idea is to randomly draw data sets with replacement from the training data, each samples the same size as the original training set. This is done G ($G = 100$) number of times, producing G bootstrap samples. We fitted the model to each of the bootstrap samples using (8). The drawback of bootstrap method is that it still does not provide a good estimate of the parameters because of the overlap of data in the training data set and original data set. This overlap makes overfitted predictions look unrealistically good. Therefore we used a cross-validation method that explicitly uses non-overlapping data for the training and test samples. To do this, for each observation, we only kept track of predictions from bootstrap samples not containing that observation and computed the 'leave-one-out' bootstrap estimate of prediction error, which is defined as

$$\widehat{\mathrm{Err}} = \frac{1}{M}\sum_{j=1}^{M}\frac{1}{|C^{-j}|}\sum_{g\in C^{-j}}(\boldsymbol{Y}_{ij} - f^{a^g_{ik}}(\boldsymbol{H}_j))^2 \tag{9}$$

Here C^{-j} is the set of indices of the bootstrap samples g that do not contain data j, and $|C^{-j}|$ is the number of such samples. a^g_{ik} are the k parameters of row i of matrix $\boldsymbol{A}$ estimated by solving (8) for the gth bootstrap sample. The 'leave-one-out' error reduces the overfitting problem but has a training-set size bias due to the fact that the bootstrap samples are not independent [16]. This can be reduced by using '$0.632+$' estimator (for details see [16]) which is designed to alleviate this bias. It pulls the 'leave-one-out' bootstrap estimate down towards the training error rate, and hence reduces its upward bias. '$0.632+$' estimator is defined as

$$\widehat{\mathrm{Err}}^{0.632+} = (1 - \widehat{\omega})\mathrm{err} + \widehat{\omega}\widehat{\mathrm{Err}} \tag{10}$$

where

$$\mathrm{err} = \frac{1}{M}\sum_{j=1}^{M}(\boldsymbol{Y}_{ij} - f^{a'_{ik}}(\boldsymbol{X}_j))^2 \tag{11}$$

a'_{ik} is computed using the complete original training dataset instead of a bootstrap sample.

$$\widehat{\omega} = \frac{0.632}{1 - 0.632\widehat{R}} \tag{12}$$

$\widehat{\omega}$ ranges from 0.632 if $\widehat{R} = 0$ to 1 if $\widehat{R} = 1$. It produces a compromise between the leave-one-out bootstrap and the training error rate that depends on the amount of overfitting. $\widehat{R}$ is called 'relative overfitting rate' defined by

$$\widehat{R} = \frac{\widehat{\mathrm{Err}} - \mathrm{err}}{\widehat{\gamma} - \mathrm{err}} \tag{13}$$

$\widehat{\gamma}$ is called no-information error rate which is the error rate of our prediction rule if input and output are independent, It is computed as

$$\widehat{\gamma} = \frac{1}{M^2}\sum_{j=1}^{M}\sum_{j'=1}^{M}(\boldsymbol{Y}_{ij} - f^{a'_{ik}}(\boldsymbol{H}_j))^2 \tag{14}$$

We then used $\widehat{\mathrm{Err}}^{0.632+}$ to select the best D solutions at each step. We used this best D solutions for next step of forward stepwise regression. At the end of next step, if we found that the minimum $\widehat{\mathrm{Err}}^{0.632+}$ is more than the minimum $\widehat{\mathrm{Err}}^{0.632+}$ of previous step, we stopped the forward stepwise regression and considered the solution from previous step as best solution. At the end, we were left with a unique combination of l ($=K$ if forward stepwise method runs until the end, or less if forward step-wise method stops earlier) out of N parameters for each row $\boldsymbol{A}_i$; however, we had G different estimates of the parameters a^g_{il}. We used 'bagging' [16] to select a unique solution. Bagging simply chooses the a^g_{il} estimate that minimises the least-square error Err_g

$$g\colon \arg\min_g\{\mathrm{Err}_g = \frac{1}{M}\sum_{j=1}^{M}(\boldsymbol{Y}_{ij} - f^{a^g_{il}}(\boldsymbol{H}_j))^2\} \tag{15}$$

2.4 Time-series network identification using final prediction error

We also tested another method for model selection that we named as time-series network identification using final prediction error (TSNIF). This method is based on computing Akaike's Final Prediction error (FPE) [19] which is defined as

$$\mathrm{FPE} = \frac{M + l}{M - l}\mathrm{err} \tag{16}$$

where err is same as defined in (11), M is the number of time points and l is the number of parameters in the model. According to Akaike's theory, the most accurate model has the smallest FPE. To achieve this, at each step of forward stepwise regression we computed FPE and moved to next the step in the iteration, until we did not find further improvement in FPE. Final solution is obtained from least-square method for the corresponding l which gives the minimum FPE.

3 Data set

3.1 Simulated gene expression data

To test the performance of the algorithm, we generated random networks of N genes with an average of K connections per gene. We produced two sets of simulations choosing $N = 10$ with $K = 2$ for the first set of 20 random networks, and $N = 20$ with $K = 5$ for the second set of 20

random networks. To generate each random network, we first created a random network using MATLAB© command rss which generates stable continuous-time state-space models ($\dot{x} = \boldsymbol{A}x + \boldsymbol{B}u; y = \boldsymbol{C}x + \boldsymbol{D}u$). rss command needs as inputs: the order of model, number of outputs (y) and the number of inputs (u) to the system. rss generates four matrices, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$ which all together define the state-space models [19].

The algorithm which rss uses to generate state-space model is as follows. For the network of size N, first it randomly chooses the number of complex and real eigen-values for matrix $\boldsymbol{A}$ (complex eigen-values always exist in pairs, eigen-value and its complex conjugate). For real part of eigen-values, negative random numbers generated from normal distribution are taken and for imaginary part positive random numbers (again from normal distribution) are taken. These eigen-values are placed on the diagonal of a zero matrix, $\boldsymbol{Z}$, of order $N \times N$. If the eigen-value pair is complex, then the real and imaginary part of the eigen-values and its complex conjugate are placed on block diagonal of $\boldsymbol{Z}$. Another matrix, $\boldsymbol{T}$, of size $N \times N$ is generated randomly from normal distribution and is then orthogonalised. Finally $\boldsymbol{A}$ is computed as $\boldsymbol{T}^{-1}\boldsymbol{Z}\boldsymbol{T}$. $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ are generated randomly using normal distribution.

Once we generated state-space model, we set C equal to identity matrix and D equal to a zero matrix so that the output of the system equals its states. Here $\boldsymbol{A}$ represents the network which is a full-rank matrix with eigen-values whose real part are less than 0 to ensure the stability of the dynamical system [19], that is all the gene mRNAs reach an equilibrium between their transcription rate and degradation rate after a given time period. For each row of $\boldsymbol{A}$, we then randomly selected K elements (including the diagonal), and set the other elements to zero (to make the network sparse). We then checked the stability of this new network by computing its eigen-values, and if $\boldsymbol{A}$ was not stable we selected a different set of K elements in each row unless we got a stable network $\boldsymbol{A}$.

We applied a single perturbation to a single gene of each random network $\boldsymbol{A}$. The information of the gene that is perturbed is contained in matrix $\boldsymbol{B}$ ($N \times 1$). $\boldsymbol{B}$ has all its elements equal to 0 except for the gene that is perturbed, which is chosen randomly. $\boldsymbol{U}$ ($1 \times M$) contains the information about what kind of perturbation is applied. In our simulations, we applied a constant step perturbation of amplitude equal to 1. Once matrix $\boldsymbol{A}$ ($N \times N$) and $\boldsymbol{B}$ ($N \times 1$) were generated, we simulated the gene expression profile data set $\boldsymbol{X} = [\boldsymbol{X}(t_1), \ldots, \boldsymbol{X}(t_M)]$ of equally sampled time points which was obtained using lsim command in MATLAB© [10]. Time t_M was chosen equal to four times the inverse of the real part of the smallest eigen-value of $\boldsymbol{A}$ [19]. This ensures that at time t_M all the genes are close to their steady-state value. We tested the algorithm performance for increasing number of time points ($M = 10$, 20, 40 and 100 for 10 gene network and $M = 20$, 40 and 100 for 20 gene network). White Gaussian noise was added to the data matrix with zero mean and standard deviation $\sigma = \alpha^* \|\boldsymbol{X}\|$ where $\|\boldsymbol{X}\|$ represents the absolute values of the elements of $\boldsymbol{X}$ and α is chosen as 0, 0.1 and 0.3. $\alpha = 0$ refers to data with no noise [10].

3.2 Experimental data set

We also tested our algorithm on a nine gene network in *Escherichia coli*. Nine genes selected are part of the DNA-damage response pathway (SOS pathway). Gene expression profiles for the nine genes were measured at six different points (with three replicates) following treatment with Norfloxacin, a known antibiotic that acts by damaging the DNA. These nine genes were selected, since the network among them is well known in the literature. For more details, refer to the work Bansal *et al.* [20].

4 Results

In order to check the performance of the algorithm for each simulated data set, we computed positive predictive value (PPV) or accuracy defined as $TP/(TP + FP)$ and sensitivity (Se) defined as $TP/(TP + FN)$ with TP: true positive; FP: false positive and FN: false negative. Before checking its performance, we rounded all the values of recovered networks to the second decimal place and removed the self-feedback (diagonal elements from $\boldsymbol{A}$) from all recovered and original networks. The reason being not to give an advantage to our algorithm, since it always recovers diagonal elements. Our algorithm infers the network as a signed directed graph (Fig. 2), but we checked its performance on recovering also the directed graph (no sign). We computed: a signed directed graph PPV^s by defining a TP when $\text{sign}(\hat{a}_{ij}) = \text{sign}(a_{ij})$ and $a_{ij} \neq 0$, $\hat{a}_{ij} \neq 0$, that is the sign of the recovered parameter equals the sign of the original parameter and both are different from zero; a directed graph PPV^d by defining a TP when $a_{ij} \neq 0$, $\hat{a}_{ij} \neq 0$; that is both are different from 0 and similarly we computed Se^s and Se^d [14].

We also computed a Random PPV, that is the PPV obtained by randomly selecting the same number of genes as in the original network. To check if any algorithm is significantly better than random on a given data or not, we computed the p-value using a binomial distribution.

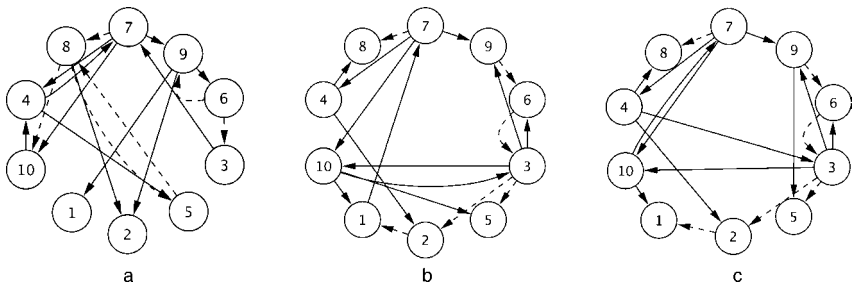

Fig. 2 *Example of signed network*

Dashed line represents negative connection and straight line represents positive connection
a Original network
b Network recovered by TSNIB for 10% noise level in the data
c Network recovered by TSNIF for 10% noise level in the data

4.1 Simulated data

We applied time-series network identification using bootstrapping (TSNIB) and TSNIF on all data sets using the information of perturbation ($\boldsymbol{U}$) and which gene is perturbed ($\boldsymbol{B}$). For forward stepwise regression step, we iterated till $K = 2$ and $K = 5$ for 10 and 20 gene networks, respectively. We computed the final PPV and Se for each data set by taking the mean and standard error mean (SEM) of PPV and Se of 20 networks for each data set. Results for the 10 gene network data set are shown in Table 1 and for 20 gene network data set in Table 2. SEM for PPV and Se is not shown in the table for clarity but is always found to be in range 0.00 and 0.04 for all data sets. Bold numbers in table are the results that are significantly better than random PPV with P-value less than 0.1.

Result shows that both TSNIB and TSNIF method can partially recover the real network. Their accuracy (PPV) decreases with increasing noise in the data, but increasing the number of time points used to infer the network can compensate for this. This result is consistent for both 10 and 20 gene network data sets. Performance of TSNIB and TSIF are similar for 10 gene network but for 20 gene network, for noisy data set, PPV of TSNIF is higher than the PPV of TSNIB method but Se is lower and vice versa. So TSNIF is better in getting small number of connection with higher accuracy but TSNIB recovers more connections with less accuracy compared to TSNIF. Also we observed that both these methods are better in recovering signed networks than directed networks. This can be explained as both these algorithms are designed to recover signed networks. So whenever they recover any correct connection, they usually recover it with correct sign. Time needed on INTEL XEON 3 GHz processor for TSNIB for 10 gene network ranges from 1 to 10 min for increasing number of data points and for 20 gene network it is 30 min to 2 h. For TSNIF, it is almost 1 s for 10 gene network and 5 s for 20 gene network for all number of data points. TSNIB method is computational intensive and is not scalable. TSNIF is scalable and is possible to apply on large networks of order 1000 genes.

4.2 Experimental data

We applied both TSNIB and TSNIF on a nine gene subnetwork, part of SOS network in *E. coli*. To this end, we computed the average of the three replicates for each time point following treatment with Norfloxacin. As the algorithm requires the first time point to be zero, we computed the log2 (log base 2) ratio of all time points with respect to first time point for each gene. We assumed the concentration of Norfloxacin to be constant over time. Norfloxacin is a member of fluoroquinolone class of antimicrobial agents that target the prokaryotic type II topoisomerase type II (DNA gyrase) and topoisomerase IV inducing the formation of single-stranded DNA and thus activating the SOS pathway via activation of the recA protein. So we considered $B = 1$ only for the element corresponding

Table 1: Results of the application of network inference algorithms on 20 networks of 10 genes

Number of data points	α	TSNIB		TSNIF		TSNI[a]		Banjo		Random
		PPV	Se	PPV	Se	PPV	Se	PPV	Se	PPV
10	0	**0.39**^{d}	**0.36**^{d}	**0.39**^{d}	**0.36**^{d}	0.26^d	0.28^d	0^d	0^d	0.20^d
		0.37^s	**0.34**^{s}	**0.37**^{s}	**0.34**^{s}	**0.24**^{s}	**0.26**^{s}	0^s	0^s	0.10^s
10	0.1	**0.29**^{d}	**0.27**^{d}	**0.32**^{d}	**0.26**^{d}	0.25^d	0.27^d	0.31^d	0.05^d	0.20^d
		0.25^s	**0.23**^{s}	**0.29**^{s}	**0.23**^{s}	**0.23**^{s}	**0.25**^{s}	0.19^s	0.04^s	0.10^s
10	0.3	0.26^d	0.24^d	0.32^d	0.24^d	0.23^d	0.25^d	0.30^d	0.09^d	0.20^d
		0.18^s	**0.17**^{s}	**0.26**^{s}	**0.19**^{s}	**0.19**^{s}	**0.21**^{s}	0.20^s	0.06^s	0.10^s
20	0	**0.50**^{d}	**0.47**^{d}	**0.51**^{d}	**0.47**^{d}	0.23^d	0.24^d	0.10^d	0.02^d	0.20^d
		0.48^s	**0.45**^{s}	**0.50**^{s}	**0.46**^{s}	**0.21**^{s}	**0.22**^{s}	0.08^s	0.01^s	0.10^s
20	0.1	**0.34**^{d}	**0.32**^{d}	**0.35**^{d}	**0.28**^{d}	0.25^d	0.26^d	0.23^d	0.09^d	0.20^d
		0.26^s	**0.26**^{s}	**0.29**^{s}	**0.24**^{s}	**0.22**^{s}	**0.23**^{s}	0.19^s	0.07^s	0.10^s
20	0.3	**0.32**^{d}	**0.29**^{d}	**0.31**^{d}	**0.25**^{d}	0.24^d	0.25^d	0.22^d	0.17^d	0.20^d
		0.22^s	**0.20**^{s}	**0.24**^{s}	**0.19**^{s}	**0.21**^{s}	**0.22**^{s}	0.14^s	0.11^s	0.10^s
50	0	**0.62**^{d}	**0.61**^{d}	**0.61**^{d}	**0.60**^{d}	**0.29**^{d}	**0.28**^{d}	0.06^d	0.01^d	0.20^d
		0.60^s	**0.59**^{s}	**0.60**^{s}	**0.59**^{s}	**0.28**^{s}	**0.26**^{s}	0.05^s	0.00^s	0.10^s
50	0.1	**0.37**^{d}	**0.35**^{d}	**0.36**^{d}	**0.30**^{d}	**0.29**^{d}	**0.28**^{d}	0.27^d	0.15^d	0.20^d
		0.30^s	**0.28**^{s}	**0.31**^{s}	**0.26**^{s}	**0.28**^{s}	**0.26**^{s}	0.20^s	0.11^s	0.10^s
50	0.3	**0.30**^{d}	**0.29**^{d}	**0.33**^{d}	**0.25**^{d}	**0.28**^{d}	**0.27**^{d}	0.15^d	0.13^d	0.20^d
		0.23^s	**0.22**^{s}	**0.27**^{s}	**0.21**^{s}	**0.26**^{s}	**0.25**^{s}	0.12^s	0.10^s	0.10^s
100	0	**0.70**^{d}	**0.70**^{d}	**0.69**^{d}	**0.69**^{d}	**0.32**^{d}	**0.30**^{d}	0.05^d	0.00^d	0.20^d
		0.69^s	**0.69**^{s}	**0.69**^{s}	**0.68**^{s}	**0.31**^{s}	**0.29**^{s}	0.05^s	0.00^s	0.10^s
100	0.1	**0.34**^{d}	**0.33**^{d}	**0.37**^{d}	**0.31**^{d}	**0.33**^{d}	**0.31**^{d}	0.23^d	0.11^d	0.20^d
		0.30^s	**0.28**^{s}	**0.34**^{s}	**0.28**^{s}	**0.31**^{s}	**0.29**^{s}	0.17^s	0.08^s	0.10^s
100	0.3	**0.31**^{d}	**0.30**^{d}	**0.36**^{d}	**0.29**^{d}	**0.32**^{d}	**0.30**^{d}	0.16^d	0.04^d	0.20^d
		0.25^s	**0.24**^{s}	**0.29**^{s}	**0.24**^{s}	**0.30**^{s}	**0.28**^{s}	0.12^s	0.03^s	0.10^s

PPV, positive predicted value; Se, sensitivity, α, noise level. In bold are the results that performed significantly better than random, using as a random model a binomial distribution.[a]corresponds to performance of TSNI by considering top K(=sparsity of network) elements in each row not equal to zero and setting rest to zero

Table 2: Results of the application of network inference algorithms on 20 networks of 20 genes

Number of data points	α	TSNIB		TSNIF		TSNI[a]		BANJO		Random
		PPV	Se	PPV	Se	PPV	Se	PPV	Se	PPV
20	0	**0.33**^{d}	**0.28**^{d}	**0.31**^{d}	**0.28**^{d}	0.28^d	0.30^d	0.16^d	0.02^d	0.25^d
		0.23^s	**0.19**^{s}	**0.21**^{s}	**0.19**^{s}	**0.21**^{s}	**0.22**^{s}	0.16^s	0.02^s	0.12^s
20	0.1	0.29^d	0.25^d	**0.32**^{d}	**0.19**^{d}	0.26^d	0.28^d	0.29^d	0.07^d	0.25^d
		0.17^s	**0.15**^{s}	**0.22**^{s}	**0.13**^{s}	**0.20**^{s}	**0.21**^{s}	0.19^s	0.05^s	0.12^s
20	0.3	0.27^d	0.24^d	0.28^d	0.19^d	0.25^d	0.26^d	0.24^d	0.07^d	0.25^d
		0.14^s	0.13^s	0.16^s	0.11^s	**0.17**^{s}	**0.17**^{s}	0.16^s	0.04^s	0.12^s
40	0	**0.35**^{d}	**0.31**^{d}	**0.35**^{d}	**0.32**^{d}	0.26^d	0.28^d	0.16^d	0.01^d	0.25^d
		0.27^s	**0.24**^{s}	**0.24**^{s}	**0.22**^{s}	**0.21**^{s}	**0.22**^{s}	0.14^s	0.01^s	0.12^s
40	0.1	**0.30**^{d}	**0.27**^{d}	**0.35**^{d}	**0.16**^{d}	0.26^d	0.27^d	0.26^d	0.09^d	0.25^d
		0.17^s	**0.16**^{s}	**0.24**^{s}	**0.11**^{s}	**0.21**^{s}	**0.22**^{s}	0.17^s	0.06^s	0.12^s
40	0.3	0.25^d	0.22^d	**0.31**^{d}	**0.16**^{d}	0.27^d	0.28^d	0.29^d	0.14^d	0.25^d
		0.18^s	**0.16**^{s}	**0.20**^{s}	**0.11**^{s}	**0.21**^{s}	**0.22**^{s}	0.18^s	0.09^s	0.12^s
100	0	**0.35**^{d}	**0.32**^{d}	**0.34**^{d}	**0.31**^{d}	0.28^d	0.25^d	0.34^d	0.02^d	0.25^d
		0.28^s	**0.25**^{s}	**0.28**^{s}	**0.25**^{s}	**0.22**^{s}	**0.20**^{s}	0.26^s	0.01^s	0.12^s
100	0.1	**0.31**^{d}	**0.28**^{d}	**0.37**^{d}	**0.17**^{d}	0.27^d	0.25^d	0.23^d	0.05^d	0.25^d
		0.20^s	**0.18**^{s}	**0.29**^{s}	**0.13**^{s}	**0.21**^{s}	**0.20**^{s}	0.13^s	0.03^s	0.12^s
100	0.3	**0.30**^{d}	**0.27**^{d}	**0.34**^{d}	**0.16**^{d}	0.27^d	0.25^d	0.18^d	0.02^d	0.25^d
		0.18^s	**0.16**^{s}	**0.25**^{s}	**0.11**^{s}	**0.22**^{s}	**0.20**^{s}	0.15^s	0.01^s	0.12^s

PPV, positive predicted value; Se, sensitivity, α, noise level. In bold are the results that performed significantly better than random, using as a random model a Binomial distribution.[a] Corresponds to performance of TSNI by considering top K(=sparsity of network) elements in each row not equal to zero and setting rest to zero

to recA gene and set the other elements to zero, while using a constant value of 1 for $\boldsymbol{U}(t_k)$. For the forward stepwise regression step, we used the information from [10] that each gene is connected to five other genes, hence we set $K = 5$.

Before computing PPV and Se, we removed the self-feedback and rounded the recovered network to second place of decimal. Results are shown in Table 3. In the literature, 43 connections (apart from self-feedback) are known among the nine genes of SOS pathway. TSNIB and TSNIF recovered 12 and 7 connections with correct sign with an accuracy of 52 and 50%, respectively. The real network is highly connected; therefore the random PPV for directed graph is very high (e.g. for fully connected network the random PPV is 1), so our method failed to give better performance than random PPV for directed graph.

4.3 Comparison with other inference methods

State-of-the-art inference methods (for a review, see [21]) are designed to infer regulatory interactions among genes/proteins, that is the gene network, from gene expression data measured after genetic perturbations. One of the most successful inference method for time-series gene expression data is dynamic Bayesian networks (DBN). Yu *et al.* [5] developed a generalisation of the DBN approach to infer signed directed networks with feed-back loops from time-series data. They developed a software, Bayesian network inference with Java objects (BANJO), implementing their novel approach. We ran BANJO on each simulated data set and compared its performance to our approach (Tables 1 and 2). BANJO recovers many connections with influence score equal to zero. We excluded all these connections and other connections whose predicted influence score was less than 0.01 (i.e rounded to second place of decimal). We ran BANJO using the default parameters (For details refer to [14]). We found that BANJO was unable to recover edges significantly in all data sets, and TSNIB and TSNIF methods outperformed BANJO consistently. The Bayesian approach is a probabilistic approach requiring a large number of data points to correctly estimate conditional probability density distributions. In addition, for simulating gene expression profiles we used the same model as used by our algorithm for inference. This could give an advantage to our algorithm over BANJO. For more detail on performance of Dynamic Bayesian Network and other algorithms for network inference using gene expression profiles, refer to Bansal *et al.* [14].

We also compared TSNIB and TSNIF method with our previously published algorithm TSNI [20]. While using TSNI, we did not use the information of which gene was perturbed in the network ($\boldsymbol{B}$ matrix) as TSNI was designed to recover it as well [20]. We found that both these methods are better or in worst case performs similar to our previously developed algorithm, TSNI.

Table 3: Results on *E. coli* data set

TSNIB		TSNIF		TSNI[a]		Banjo		Random
PPV	Se	PPV	Se	PPV	Se	PPV	Se	PPV
0.65^d	0.35^d	0.64^d	0.21^d	0.60^d	0.63^d	0^d	0^d	0.6^d
0.52^s	**0.28**^{s}	**0.50**^{s}	**0.16**^{s}	**0.42**^{s}	**0.44**^{s}	0^s	0^s	0.3^s

PPV, positive predicted value; Se, sensitivity, α, noise level. In bold are the results that performed significantly better than random, using as a random model a Binomial distribution.
[a]Corresponds to the performance of TSNI when we used the information from [10] that each gene is connected with other five genes ($K = 5$) and set the four weakest connection in each row of $\boldsymbol{A}$ to zero

5 Discussion

In this work, we present a new gene network inference approach to partially recover a gene network from few and noisy measurements. Our approach is based on the assumption that gene expressions are the outcome of a linear model. Even if biological systems are non-linear, this assumption still holds if the perturbation is small. Small perturbations will keep the system close to its stable state thus behaving linearly, but larger (or sometime even complex) perturbations can lead to nonlinear behaviour. In that case, our method will not be appropriate for inference of gene networks. If the system is nonlinear and if there are lot of datapoints then Bayesian approach may be an appropriate algorithm to apply on that data set. To improve the performance of our algorithm, we should perturb more than one gene at a time with a more complex dynamics. This, however, is difficult to implement experimentally, since already a single perturbation requires considerable experimental effort. More complex perturbations are necessary for improving inference performance of any algorithm based on time-series gene expression data, but at this stage it is still not practical in experimental molecular biology. This situation may soon change, thanks to the progress in synthetic biology that will allow construction of inducible systems with more complex dynamics [22].

6 Acknowledgment

This work was supported by 'Fondazione Telethon' Grant TDDP17TELB and the 'Scuola Europea di Medicina Molecolare-SEMM' Grant for M.B.

7 References

1 Akutsu, T., Miyano, S., and Kuhara, S.: 'Identification of genetic networks from a small number of gene expression patterns under the Boolean network model', *Pac. Symp. Biocomput.*, 1999, pp. 17–28.
2 Akutsu, T., Miyano, S., and Kuhara, S.: 'Inferring qualitative relations in genetic networks and metabolic pathways', *Bioinformatics*, 2000, **16**, (8), pp. 727–734
3 Liang, S., Fuhrman, S., and Somogyi, R.: 'Reveal, a general reverse engineering algorithm for inference of genetic network architectures', *Pac. Symp. Biocomput.*, 1998, pp. 18–29.
4 Dojer, N., Gambin, A., Mizera, A., Wilczynski, B., and Tiuryn, J.: 'Applying dynamic Bayesian networks to perturbed gene expression data', *BMC Bioinformatics*, 2006, **7**, p. 249
5 Yu, J., Anne Smith, V., Wang, P.P., Hartemink, A.J., and Jarvis, E.D.: 'Advances to bayesian network inference for generating causal networks from observational biological data', *Bioinformatics*, 2004, **20**, (18), pp. 3594–3603
6 Basso, K., Margolin, A.A., Stolovitzky, G., Klein, U., Dalla-Favera, R., and Califano, A.: 'Reverse engineering of regulatory networks in human B cells', *Nat. Genet.*, 2005, **37**, (4), pp. 382–390
7 Butte, A.J., and Kohane, I.S.: 'Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements', *Pac. Symp. Biocomput.*, 2000, pp. 418–429.
8 Faith, J.J., Hayete, B., Thaden, J.T., Mogno, I., Wierzbowski, J., Cottarel, G., Kasif, S., Collins, J.J., and Gardner, T.S.: 'Large-scale mapping and validation of *Escherichia coli* transcriptional regulation from a compendium of expression profiles', *PLoS Biol.*, 2007, **5**, (1), p. e8
9 Bonneau, R., Reiss, D.J., Shannon, P., Facciotti, M., Hood, L., Baliga, N.S., and Thorsson, V.: 'The inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo (evaluation studies)', *Genome. Biol.*, 2006, **7**, (5), p. R36
10 Gardner, T.S., di Bernardo, D., Lorenz, D., and Collins, J.J.: 'Inferring genetic networks and identifying compound mode of action via expression profiling', *Science*, 2003, **301**, pp. 102–105
11 van Someren, E.P., Wessels, L.F., and Reinders, M.J.: 'Linear modeling of genetic networks from experimental data', *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 2000, **8**, pp. 355–366
12 Yeung, M.K.S., Tegnfier, J., and Collins, J.J.: 'Reverse engineering gene networks using singular value decomposition and robust regression', *Proc. Natl Acad. Sci. USA*, 2002, **99**, pp. 6163–6168
13 van Someren, E.P., Vaes, B.L.T., Steegenga, W.T., Sijbers, A.M., Dechering, K.J., and Reinders, M.J.T.: 'Least absolute regression network analysis of the murine osteoblast difierentiation network', *Bioinformatics*, 2006, **22**, (4), pp. 477–484
14 Bansal, M., Belcastro, V., Ambesi-Impiombato, A., and di Bernardo, D.: 'How to infer gene networks from expression profiles (comparative study)', *Mol. Syst. Biol.*, 2007, **3**, p. 78
15 de Jong, H.: 'Modeling and simulation of genetic regulatory systems: A literature review', *J. Comp. Biol.*, 2002, **9**, pp. 67–103
16 Hastie, T.J., Tibshirani, R.J., and Friedman, J.: 'The elements of statistical learning' (Springer, 2001)
17 van Someren, E.P., Wessels, L.F.A., Reinders, M.J.T., and Backer, E.: 'Searching for limited connectivity in genetic network models'. Proc. 2nd Int. Conf. Systems Biol, 2001, pp. 222–230
18 Di Bernardo, D., Gardner, T.S., and Collins, J.J.: 'Robust identification of large genetic networks', *Proc. Pacific Symp. Biocomp.*, 2004, pp. 486–497.
19 Ljung, L.: 'System identification: theory for the user' (Upper Saddle River, Prentice-Hall, NY, 1999)
20 Bansal, M., Gatta, G.D., and di Bernardo, D.: 'Inference of gene regulatory networks and compound mode of action from time course gene expression profiles', *Bioinformatics*, 2006, **22**, (7), pp. 815–822
21 Faith, j., and Gardner, T.S.: 'Reverse-engineering transcription control networks', *Phys. Life Rev.*, 2005, **2**, pp. 65–88
22 Hasty, J., McMillen, D., and Collins, J.J.: 'Engineered gene circuits', *Nature*, 2002, **420**, (6912), pp. 224–230