BIOINFORMATICS ORIGINAL PAPER
Vol. 22 no. 7 2006, pages 815–822
doi:10.1093/bioinformatics/btl003

Systems biology

Inference of gene regulatory networks and compound mode of action from time course gene expression profiles

Mukesh Bansal[1,2], Giusy Della Gatta[1,3] and Diego di Bernardo[1,2,*]

[1]Telethon Institute of Genetics and Medicine, Via P. Castellino 111, 80131 Naples, Italy, [2]European School of Molecular Medicine (SEMM), Naples, Italy and [3]Seconda Universita degli Studi di Napoli, Naples, Italy

Received on June 16, 2005; revised on November 22, 2005; accepted on January 12, 2006
Advance Access publication January 17, 2006
Associate Editor: Satoru Miyano

ABSTRACT

Motivation: Time series expression experiments are an increasingly popular method for studying a wide range of biological systems. Here we developed an algorithm that can infer the local network of gene–gene interactions surrounding a gene of interest. This is achieved by a perturbation of the gene of interest and subsequently measuring the gene expression profiles at multiple time points. We applied this algorithm to computer simulated data and to experimental data on a nine gene network in *Escherichia coli*.

Results: In this paper we show that it is possible to recover the gene regulatory network from a time series data of gene expression following a perturbation to the cell. We show this both on simulated data and on a nine gene subnetwork part of the DNA-damage response pathway (SOS pathway) in the bacteria *E. coli*.

Contact: dibernardo@tigem.it

Supplementary information: Supplementary data are available at http://dibernado.tigem.it

1 INTRODUCTION

Recent developments in large-scale genomic technologies, such as DNA microarrays and mass spectroscopy have made the analysis of gene networks more feasible. However, it is not obvious how the data acquired through such methods can be assembled into unambiguous and predictive models of these networks. Different experimental and computational methods have been proposed to tackle the network identification problem (Tong *et al*., 2002; Lee *et al*., 2002; Ideker *et al*., 2001; Davidson *et al*., 2002; Arkin *et al*., 1997; Yeung *et al*., 2002). Although implemented with some success, they are data intensive and they may require a certain degree of a priori information.

A variety of mathematical models can be used to describe genetic networks (de Jong, 2002; Savageau, 2001; Levchenko and Iglesias, 2002), including Boolean logic (Shmulevich *et al*., 2002; Liang *et al*., 1998), Bayesian networks (Hartemink *et al*., 2002), graph theory (Wagner, 2001) and ordinary differential equations (Tegner *et al*., 2003). We concentrated our efforts on the last method as it offers a description of the network as a continuous time dynamical system that can be used to infer the genes with the major regulatory functions in the network.

In a recent study (Gardner *et al*., 2003), we developed an algorithm (Network Identification by multiple regression—NIR) that used a series of steady state RNA expression measurements, following transcriptional perturbations, to construct a model of a nine gene network that is a part of the larger SOS network in *E. coli* (Gardner *et al*., 2003). Though the NIR method proved highly effective in inferring small microbial gene networks, it requires prior knowledge of which genes are involved in the network of interest, and the perturbation of all the genes in the network via the construction of appropriate episomal plasmids. In addition, it requires the measurement of gene expressions at steady state (i.e. constant physiological conditions) after the perturbation. This experimental setup is challenging for large networks, it is not easily applicable to higher organisms, and, most importantly, it is not applicable if there is no prior knowledge of the genes belonging to the network.

In this paper we are presenting an algorithm TSNI (Time Series Network Identification) that can infer the local network of gene–gene interactions surrounding a gene of interest by perturbing only one of the genes in the network. To this end, we need to measure gene expression profiles at multiple time points following perturbation of the gene, or genes, of interest.

We investigated the effect of noise and a limited number of data points on the performance of the algorithm, and we devised techniques to overcome these problems.

Our algorithm is illustrated and tested *in silico* on computer simulated gene expression data and applied to an experimental gene expression data set obtained by perturbing the SOS system in the bacteria *E. coli*.

The novelty of our approach is in the idea of a gene-centric inference method that can be applied to infer the regulatory interactions of a gene of interest. State-of-the-art inference algorithms start from the assumption that a gene network is unknown and experiments are performed to perturb it. Gene expression data are then used to reconstruct the network. In a real life situation, large-scale gene expression data from a given cell type involve thousands of responsive genes and there are many different regulatory networks activated at the same time by the pertrubations. In this case, inference methods can be successful but only on a subset of the genes (i.e. a specific network) (Basso *et al*., 2005). This subset of genes (network), however, cannot be defined a priori but depends on the data set. Networks involving genes that never change in the dataset cannot be inferred. We aim at developing an integrated experimental and computational approach to infer the network of a specific gene of interest.

*To whom correspondence should be addressed.

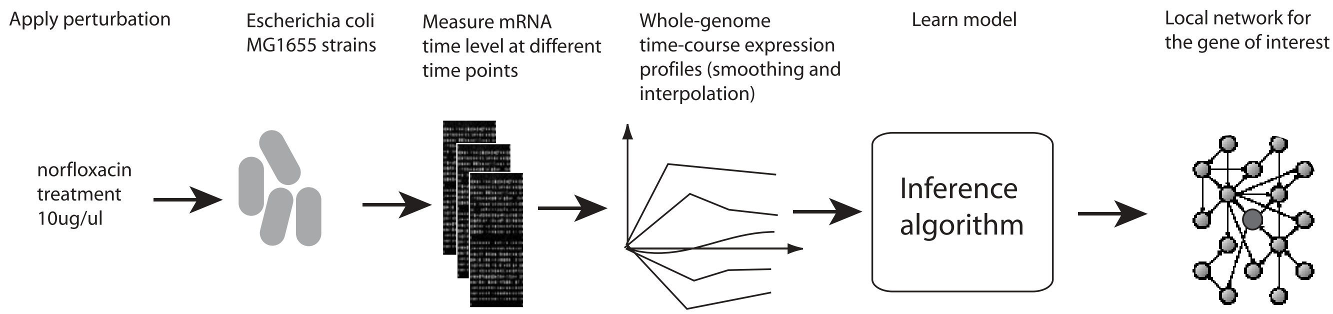

Fig. 1. Overview of the methodology used to infer the network.

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 (as shown in 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, or a genetic perturbation involving overexpression or downregulation of particular genes. We use the following system of ordinary differential equations (de Jong, 2002) 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:

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

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, i.e. 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 1th pertrubation on x_i and $u_l(t_k)$ represents the lth external perturbation at time t_k.

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

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

where $X(t_k)$ is an $N \times 1$ vector of mRNA concentration of N genes at time t_k, $\dot{X}(t_k)$ is a $N \times 1$ vector of the first derivatives of X at time t_k, A is a $N \times N$ connectivity matrix, composed of elements a_{ij}, B is a $N \times P$ matrix representing the effect of P perturbations on N genes, $U(t_k)$ 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 unknowns to calculate are the connectivity matrix A and matrix B. Element (i, l) of B will be different from zero if the i-th gene is a direct target of the l-th perturbation.

2.2 TSNI Algorithm

The TSNI algorithm identifies the network of the genes (A) as well as the direct targets of the perturbations (B). To identify the network means to retrieve A and to identify the direct targets of the drugs means to identify B, by solving Equation (2). Solving this equation is possible only if $M \geq N + P$. Since usually we have less data points (time points) than the number of genes, we cannot solve Equation (2) directly. A solution can be found either by increasing the number of data points artificially by interpolation or by dimensional reduction techniques. In our algorithm we apply both approaches to the dataset: first, we apply a cubic smoothing spline filter with an adjustable smoothing parameter (de Boor, 2001). This smoothing filter reduces the fluctuations in the data introduced by noise. Second we increase the number of time points by interpolating the smoothed data using piecewise cubic spline interpolation. Third, we apply Principle Component Analysis (PCA) to the dataset in order to reduce its dimensionality and solve the equation in the reduced dimension space as described below.

To solve Equation (2) we need the first time derivative of gene expression profile. Since the data are noisy, taking derivatives will further increase the noise level. In order to avoid this problem we convert Equation (2) to its discrete form (Ljung, 1999).

$$X(t_{k+1}) = A_d * X(t_k) + B_d * U(t_k), \quad (3)$$

where A_d is the network in the discrete space, which is different from A (Ljung, 1999) and B_d is discrete counterpart of B. Rewriting Equation (3)

$$X(t_{k+1}) = [A_d \quad B_d] * \begin{bmatrix} X(t_k) \\ U(t_k) \end{bmatrix} \quad (4)$$

which can be written for all time points;

$$X = H * Y, \quad (5)$$

where

$$H = [A_d \quad B_d]$$

and

$$Y = \begin{bmatrix} X \\ U \end{bmatrix}.$$

Dimensions of X, U, H and Y are $N \times (M - 1)$, $P \times (M - 1)$, $N \times (N + P)$ and $(N + P) \times (M - 1)$, respectively. We apply the PCA to reduce the dimension of Equation (5) by decomposing Y using singular value decomposition (Lay, 2002)

$$X = H * V * D * T', \quad (6)$$

where columns V are left singular vectors, rows of T' are right singular vectors and D is a diagonal matrix of singular values arranged in descending order. Choosing the top k singular values, we can write

$$X = Z_d * Y_{R,} \quad (7)$$

where Z_d is obtained by taking first k columns of $H * V$ and Y_R is the data in the reduced dimension obtained by taking the first k rows of $D * T'$. The solution is obtained by taking pseudo-inverse of Y to obtain

$$Z_d = X * Y_R^T * (Y_R \cdot Y_R^T)^{-1} \quad (8)$$

We then project the solution Z_d obtained in the reduced dimension space to the original dimension space using matrix V (Montgomery *et al*., 2001) to get A_d and B_d. In order to compute the continuos network model A and continuous form of B from its discretized form A_d and B_d respectively, we apply the following bilinear transformation (Ljung, 1999):

$$A = \frac{2}{\delta t} \frac{A_d - I}{A_d + I} \quad (9)$$

$$B = (Ad + I)^{-1} * A * B_d, \quad (10)$$

where I is the square identity matrix of dimension $N \times N$ and δt is the sampling interval. The transformation from discrete to continuous model is an important step. All the work presented in the literature till now is based on the discrete time model even if the dynamics of gene regulation is continuous in time.

2.3 Simulated gene expression data

Before applying the algorithm to the real dataset, we tested its performance on simulated datasets. We tested the performance on two different simulated datasets, one corresponding to a set of small gene networks with 10 genes and another one corresponding to larger networks with 1000 genes. To test the performance of TSNI on both these networks with 10 and 1000 genes, we generated 100 random networks with an average of 5 and 100 connections per gene, respectively. Each network was represented by a full rank sparse matrix A ($N \times N$) with eigenvalues with a real part less than 0 (Ljung, 1999) to ensure the stability of the dynamical systems, i.e. all the gene mRNAs reach an equilibrium between their transcription rate and degradation rate after a given time period. We applied $P = 1$ perturbations to each network. For networks of 10 genes, we perturbed 1 gene, while for networks of 1000 genes, we performed two sets of perturbations, one in which we perturbed only 1 gene and the other in which we perturbed 100 genes simultaneously. The information of which gene is perturbed is contained in B ($N \times 1$). B has all its elements equal to 0 except for the genes that are the direct target of the perturbation. U ($1 \times M$) contains the information about what kind of perturbation is applied. In our simulation we applied a constant perturbation, so all elements of U are kept constant (1 in our simulation). The simulated gene expression profile dataset $X = |X(t_1), \ldots, X(t_k)|$ was obtained using *lsim* command in MATLAB [see supplementary of Gardner *et al*. (2003) for more details of how to obtain simulated gene expression profile] by solving

$$\dot{X} = AX + BU, \tag{11}$$

where X ($N \times M$) is the response of the N genes at M time points following the perturbation. The end time t_e of the simulated time series was chosen equal to four times the inverse of the real part of the smallest eigen value of A (Ljung, 1999). This ensures that at time t_e all the genes are close to their steady-state values. We then selected 5 and 10 time points for the 10 and 1000 gene networks, respectively. These time points were equally spaced from the start time t_s to the end time t_e. White Gaussian noise was added to the data matrix with zero mean and varying the standard deviation from $\sigma = 0 * \|X\|$ to $\sigma = .50 * \|X\|$, with an interval of 0.1, where $\|X\|$ represents the absolute values of the elements of X (Gardner *et al*., 2003). The simulated gene expression time courses are then filtered using the smoothing algorithm described in (de Boor, 2001) with a default parameter of 0.8. Smoothing is widely used in signal processing to remove outliers, if any, from the time course, to reduce the measurement noise, and, to increase the number of data points via interpolation. However, to our knowledge, it has never been applied on time series gene expression data.

2.3.1 *Assessing the performance of the algorithm.*

- Gene regulatory network: Matrix $\hat{A}$ inferred by the TSNI algorithm has $N \times N$ elements describing the regulatory influences among the N genes in the network. In the recovered network all the elements are non-zero. To make the network sparse (Gardner *et al*., 2003), we set the smallest h elements in $\hat{A}$ to zero. We calculated the ratio, r_z, of number of correctly identified zero coefficients in the recovered $\hat{A}$ to the number of zero elements in the original A and the ratio, r_{nz}, of total number of non-zero elements in the recovered $\hat{A}$ whose signs are in agreement with the signs of non zero elements in the original A. We varied h from 0 to the number of elements in $\hat{A}$, and calculated r_z and r_{nz} for each h.
- Direct targets of the perturbations: Matrix $\hat{B}$ inferred by the algorithm has $N \times 1$ elements that describe the direct targets of the perturbation. In the recovered $\hat{B}$, all the elements are non zero. We sort all the elements of $\hat{B}$ according to their absolute values and selected the top h largest elements and set remaining $N - h$ elements to zero. Once the N − h elements of $\hat{B}$ are set to 0, to assess how well the algorithm can infer the direct targets of perturbation, we defined as True Positives (TP) those elements $\hat{b}_i$ of the inferred $\hat{B}$ that are different from 0 and that are non-zero in the original B. Similarly False Positives (FP) are all the elements $\hat{b}_i$ that are different from 0 while the original b_i are 0. Analogously, we defined the False Negatives (FN) and True Negatives. We measured the overall performance by computing the positive predictive value (PPV) $\frac{TP}{(TP+FP)}$ and sensitivity $\frac{TP}{(TP+FN)}$ by varying h from 1 to N.

2.4 Experimental methods

2.4.1 *Growth conditions and E.coli treatment.*

The bacterial strain MG1655 was grown over night in 5 ml LB amp 100 μg/ml with shaking (300 r.p.m.) at 37°C. The time course experiment consisted in the induction with 10 μg/ml of Norfloxacin and extraction of the total RNA at the following time points: 0, 12, 24, 36, 48 and 60 min from the drug treatment. Each experiment was done in triplicate; positive controls were done at 24 and 60 min from the induction with Norfloxacin.

2.4.2 *Preparation of E. coli for hybridization to Affymetrix Chips*

- RNA extraction: Cultures were centrifuged at 3000 r.p.m. for 5 min at 4°C, the pellets were re-suspended in RNA protect and incubated for 10 min at room temperature. The RNA protect is completely poured off and the cells pellets were frozen at −80°C. RNA was prepared using the spin protocol for the RNAeasy 96 kit (Qiagen on Column Dnase digestion).
- cDNA synthesis: For each sample reverse transcription of RNA was performed using First Strand cDNA kit according to manufacter's instructions. The RNA was degraded by the addition of 1 M NaOH and heating to 65°C for 30 min. The reactions were neutralized with 1 M HCl. The reactions were purified using Qiagen QIAquick columns following the manufacturer's protocol. The DNA was eluted from the columns with 40 μl of EB buffer. To digest any genomic DNA present in the samples, 3 μg of each cDNA was fragmented by combining with the following 10× One-Phor-All Buffer, 1 U/μl DNase I, in a final volume of 50 μl H2O. The reactions were incubated in a thermocycler at 37°C for 10 min without the hot top, followed by inactivation of the DNase I at 98°C for 10 min.
- cDNA labeling and hybridization: The fragmented cDNAs were end-labeled using an Enzo BioArray Terminal Labeling Kit with Biotin-ddUTP. To each tube of fragmented cDNA the following was added: 5× reaction buffer, 10× CoCl2, 100× Biotin-ddUTP, 50× Terminal Deoxynucleotide Transferase in a final volume of 100 μl H2O. The reactions were incubated in a heatblock at 37°C for 1 h 15 min. The reactions were quenched with the addition of 2 μl 0.5 M EDTA, pH 8.0, according to manufacturers instruction. A 3 μl aliquot was taken from each tube and dried. A 2 mg/mL NeutrAvidin was added to each tube and incubated at room temp for 5 min. The conjugates along with an additional 3 μl of the labled cDNA were electrophoresed on 3% non-denaturing agarose, 1× TAE gels at 250 V for 20 min. The gels were stained with a solution of 0.1% SYBR Gold in 1× TEA for 25 min then imaged. The fragmented, labeled cDNAs were prepared for hybridization by combining with the following: cDNA, 2× Hybridization buffer, 50 mg/ml acetylated BSA, 10 mg/ml Herring Sperm DNA, 3 nM Control Oligo B2, Agent-X in a total volume of 200 μl. The mixtures were loaded on each chip and hybridized overnight at 45°C and 60 r.p.m. Following a

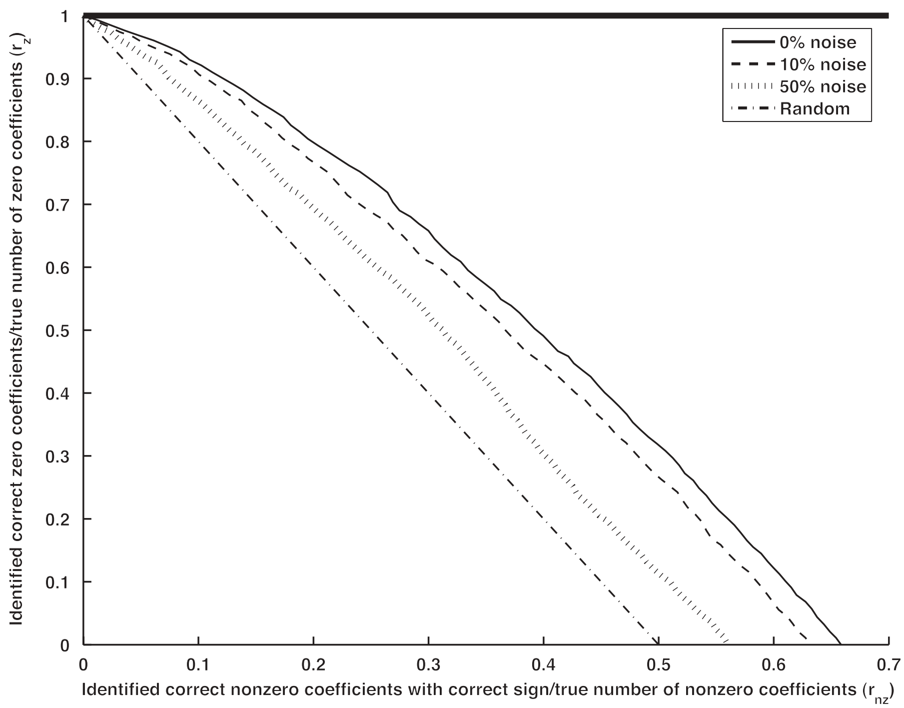

Fig. 2. Plot of average of r_{nz} vs r_z across 100 random system for double interpolation obtained the simulation. Different curves are different noise levels. Dash dotted curve is the performance which is obtained by building the connections in the network randomly. $r_z = 1$ line corresponds to the ideal curve.

minimum of 15 h of hybridization, the chips were stained and scanned according to Affymetrix protocols.

2.4.3 Microarray data analysis. We processed the microarray data using the *rma* command of the Bioconductor package that performs normalization and gene expression estimation from replicates using the algorithm described by Gautier *et al*. (2004). By calculating the mean (μ) and standard deviation (σ) on the three replicates of each time point, we found that the noise level in our experiment is $\sim$13% ($\frac{\mu}{\sigma} = 0.13$).

3 RESULTS

3.1 Choosing the parameters of the model

In order to set the best value for the number of interpolated points and the number of principle components, we applied the TSNI algorithm to each simulated dataset and varied: (1a) the number of interpolated data point ranging from 0 (no interpolation) to $10 \times M$ (10 times the number of experimental points in the time series); (2b) the number of principle components from 1 to min (N, M) (minimum of the number of genes in the network or the number of time points). For each of these parameters, we calculated the average of r_{nz} vs r_z across 100 random systems by varying h from 0 to the number of elements in $\hat{A}$ and plotted r_{nz} vs r_z. Ideally, when we increase the number of nonzero elements in $\hat{A}$ by varying h, we should start identifying non-zero elements correctly with zero false positives, which will keep the value of r_z equal to 1 and increase r_{nz} from 0 to 1. The best value of h will correspond to that value where both r_z and r_{nz} equal one. At the point when $\hat{A}$ is fully connected we should have r_{nz} equals 1 and r_z equals zero. We selected as the best set of parameters, the ones that gave the maximum area under the r_{nz} versaus r_z curve (since we have one of such curves for each parameter value that we explored).

3.2 Result on simulated 10 gene network

To select the best set of parameters, we plotted the area under r_{nz} versus r_z curve for all the set of parameters. First we plotted the area under r_{nz} versus r_z curve for different interpolation levels and different principle components for various noise levels (see Figure 1S for 0% noise level and Figure 2S for 10% noise level in supplementary). These plots shows that double interpolation (two times the number of data points in the time series) works best. Once the interpolation is decided, we then plotted the area under r_{nz} versus r_z curve for different noise level and different principle components (see fig 3S in supplementary) after fixing the interpolation to two times. From this we found that if the noise level is very low, then three principle components work best. At 10% noise level, two principle components work better, whereas at higher noise levels only one principle component works well. This is to be expected, since higher principle components capture also the noise signal, whereas most of the information is captured in lower components. We therefore selected double interpolation and two and three principle components for our study, as we know that the noise level in our real data is $\sim$13% (see Section 2.4.3).

Figure 2 shows the plot for average of r_{nz} versus r_z across 100 random networks for double interpolation and three principle components at various noise level. The dash dotted line shows the performance when we select the connections in the network randomly. The performance of the algorithm decreases clearly

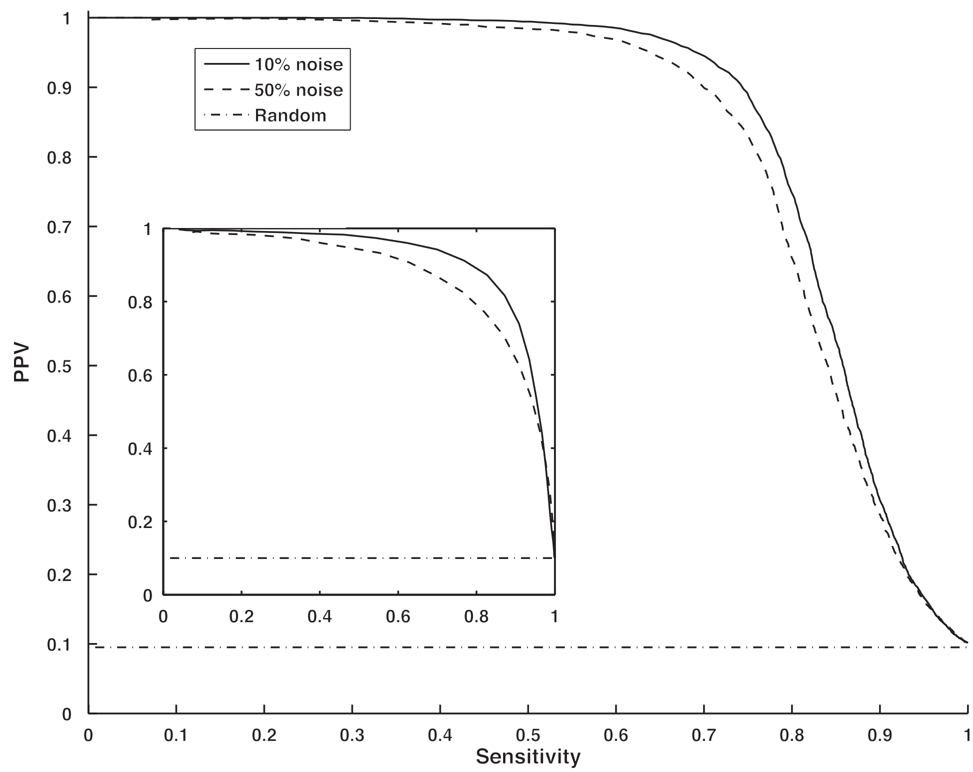

Fig. 3. Plot of PPV versus sensitivity for 1000 genes network when predicting genes directly regulated by the perturbed gene (column of $\hat{A}$). Two different curves for 10 and 50% noise level are shown. The dash dotted horizontal line at the bottom shows the curve when we select the genes randomly. In the inset, is the PPV versus sensitivity curve for the prediction of targets of perturbation for 1000 gene network ($\hat{B}$).

with increasing noise levels. We also checked how well we can recover $\hat{B}$ (graph not shown here) and the result shows that we can predict the targets of the perturbation with a sensitivity of 99% with 96% PPV, i.e. 96% of the times we were able to tell correctly the correct target without any false positives.

3.3 Result on simulated 1000 gene network

3.3.1 Results on inferring network ($\hat{A}$) On larger networks of 1000 genes the performance of our algorithm in recovering the $\hat{A}$ matrix is not very good and infact r_{nz} versus r_z curve overlapped the random curve. This happened because the network is not fully observable and one experiment does not yield sufficient information to infer the network. To check whether we can infer the local network around a gene of interest, we selected the column corresponding to the perturbed gene in the simulation in which we perturbed only one gene, and compared it with the corresponding column in the original network A. This corresponds to infer the genes that are directly regulated by the perturbed gene. In Figure 3 we computed the PPV and sensitivity in the same way as described above when assessing the performance in getting the direct targets of the perturbation (Section 2.3.1). We found that for double interpolation and 1 principle components (which is found again by the checking the set of parameters which gives the maximum area under ppv vs sensitivity curve for all set of parameters) at 10% noise we get 75% sensitivity with almost 100% PPV. PPV decreases to 90% at 50% noise level, i.e. if the gene regulates 100 different genes and algorithm predicts 75 genes as direct target then all of them are real at 10% noise and 68 of them are real at 50% noise. The horizontal dash dotted line at the bottom of Figure 3 shows the performance if we select the genes randomly without using any prediction algorithm.

3.3.2 Results on inferring the targets of perturbation ($\hat{B}$) We then checked also how good is algorithm in predicting the targets of a perturbation (B). To this end, we performed the simulation by perturbing 100 genes simultaneously (see Section 2.3). We found again the best set of parameters using ppv vs sensitivity plot by comparing original B and the recovered $\hat{B}$. We found that double interpolation and one principle component work best. The result shows (Fig. 3 inset) that up to 50% sensitivity we predict the targets with almost 100% PPV at 10% noise, and 95% PPV at 50% noise.

This shows that TSNI is very good in predicting the local network around the perturbed gene, if we perturb only one gene. If we treat the cells with some drug, which has multiple gene targets, then TSNI can also find them with a very good ppv.

3.4 Results on *E.coli*

3.4.1 Results on inferring the network ($\hat{A}$) We applied our TSNI algorithm to a nine gene network, part of SOS network in *E. coli*. Genes are the same as the ones we used in our previous work (Gardner *et al.*, 2003) in order to see how well we can replicate the results. To this end we computed the average of the three replicates for each time point following treatment with Norfloxacin, a

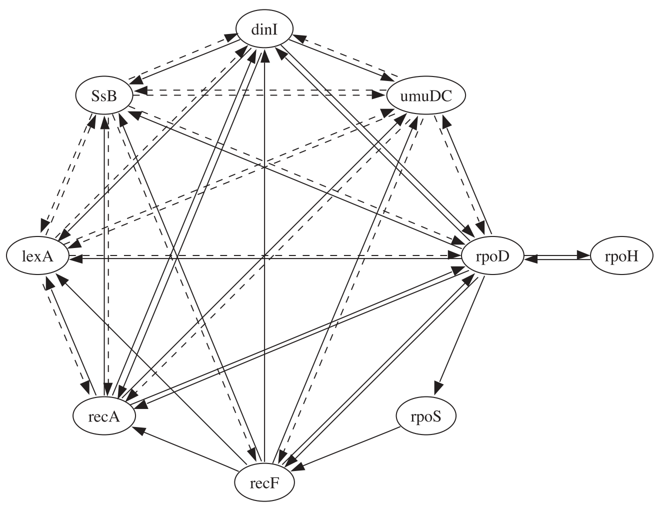

Fig. 4. Gene-gene interaction between the nine genes of SOS network in *E.coli* known in literature. Positive interactions are shown as line, and negative interactions are shown as dotted line.

known antibiotic that acts by damaging the DNA. In order to assess the performance of the algorithm on this experimental data, we compared the inferred network with the one we identified in our previous work (Gardner *et al*., 2003) and with a literature survey of the known interactions among these nine genes (Fig. 4). We found 43 connections, apart from the self-feedback, between these genes that are known in literature. The network obtained by the algorithm for the *E. coli* time-series data for three principle components and double interpolation is shown in Table 1. We compared this predicted network with known connections from the literature and plotted r_{nz} versus r_z (Fig. 5). The cross on the plot shows the value of r_{nz} and r_z which is obtained by comparing the network predicted in our previous work (Gardner *et al*., 2003) with the network from the literature. NIR found 22 connections correctly out of 43 known connections. The result of our present study is similar to our previous work, even if we used only a single perturbation experiment and 5 time points as compared to our previous work in which we used nine different perturbation experiments and we also assumed the matrix B to be known.

When we used the information that there should be five connections for each gene, [from our previous work (Gardner *et al.* (2003))], and set four elements in each row of the inferred matrix $\hat{A}$ to zero, then our algorithm finds 20 connections correctly (diamond in Fig. 5).

3.4.2 Results on inferring the targets ($\hat{B}$) To check the prediction of B, we considered the treatment of *E.coli* with Norfloxacin equivalent to the a perturbation to recA. 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 recAp protein. Quinolones have been previously demonstrated to induce recA and other SOS-responsive genes in *E.coli*. (Phillips *et al*., 1987). We checked that we can get recA as the strongest target. This gives 100% value for both positive predicted value and sensitivity, which shows that TSNI algorithm is very good in predicting drug target, at least for small network.

We then checked how well we can do if we select a larger dataset of genes in *E.coli*. We applied our algorithm to the 300 genes which statistically responded the to the Norfloxacin treatment. We ordered the absolute value of all elements in recovered $\hat{B}$ by TSNI and looked at the top 50 genes (Table 1S in supplementary). We found that recA was ranked 14 and there were 9 genes which belong to SOS pathway in the top 50 genes.

3.5 Comparison with Dynamic Bayesian Network

We then compared our algorithm with Dynamic Bayesian Network (DBN), one of the most successful algorithms available now for time series. The standard DBN (Murphy, 2001) cannot be compared with our algorithm directly, since DBN infers an undirected network (i.e. it does not give the sign of the connection in the network), and it is not able to infer feedback loops. We therefore decided to compare our work with the work by Yu *et al*. (2004). In this paper, the authors developed a generalization of the DBN algorithm to infer directed networks with feedback loops. The authors then applied their algorithm on a network of 20 genes and tested its performance by using different number of data points, ranging from 25 to 5000. Figure 5a in their paper shows that for high number of data points (>1000), they are able to recover 98% of the connections in the network correctly, but using lower number number of data points, their performance quickly decreases. In addition, their performance varies a lot, depending on the number of parents of each node. When the network is very sparse, i.e. each node has few parents (1 or 2) it works well, but when the network becomes dense, with three or more parents per node then the performance decreases a lot. In our dataset, we assumed only 5 data points for a 10 gene network. In addition, we assume that the network is not very sparse and has five parents for each node (Gardner *et al*., 2003). For this dataset, therefore, the performance of the algorithm of Yu *et al*. (2004) is equivalent to the random algorithm.

4 DISCUSSION

In this pilot study we investigated the possibility of inferring the local network of regulatory interactions surrounding a gene of interest when there is no a priori knowledge of the genes belonging to network, nor about the structure of the network. We found that by perturbing a gene of interest and measuring the response of the genes following the perturbation, it is possible to partially reconstruct the regulatory network. Our approach confirmed many of the known information from literature about different interactions in the SOS network. We propose a robust method that allows to infer a continuous-time model of a gene network from time series data without requiring the estimation of the first derivatives, thanks to the use of the bilinear transformation. We also show that smoothing is a very powerful technique to reduce the noise in the data and should be used prior to interpolation.

The TSNI algorithm could be a powerful methodology for the drug discovery process since it would be able to identify the compound mode of action via a time-course gene expression profile. We have already shown in a recent study (di Bernardo *et al*., 2005), using a different approach that requires large collection of whole-genome gene expression profiles in yeast, that it is possible to infer compound mode of action by analyzing transcriptional response

Table 1. The SOS network for *E.coli* obtained by TSNI algorithm

	recA	lexA	Ssb	recF	dinI	umuDC	rpoD	rpoH	rpoS
recA	−1.2436	−0.2937	−1.3958	1.7862	0.1864	−0.3994	1.0770	0.3188	−1.7990
lexA	0.5260	−3.8494	0.1855	0.1814	0.4144	0.1139	0.3016	0.0215	0.0030
Ssb	0.3046	0.3180	−3.2623	0.2029	0.4980	0.2223	0.7421	0.2047	0.7331
recF	0.8962	−0.4283	−0.9027	−2.5483	−0.9168	−0.5607	1.2184	0.6924	−0.3771
dinI	1.8592	0.2694	−0.0876	0.4730	−2.7953	0.2360	0.3258	−0.1914	−0.8924
umuDC	0.4951	0.0905	−0.0058	0.0599	0.3861	−3.9094	0.0095	−0.0987	−0.2706
rpoD	1.0374	−0.3098	−0.4930	1.8567	−0.9117	−0.5429	−1.9702	1.0397	0.2742
rpoH	0.0162	−0.2580	−0.3403	0.7112	−0.7695	−0.3402	0.7250	−3.5191	0.1869
rpoS	−0.6059	0.0649	0.6377	0.3339	−0.4751	−0.0594	0.9743	0.5517	−2.6977

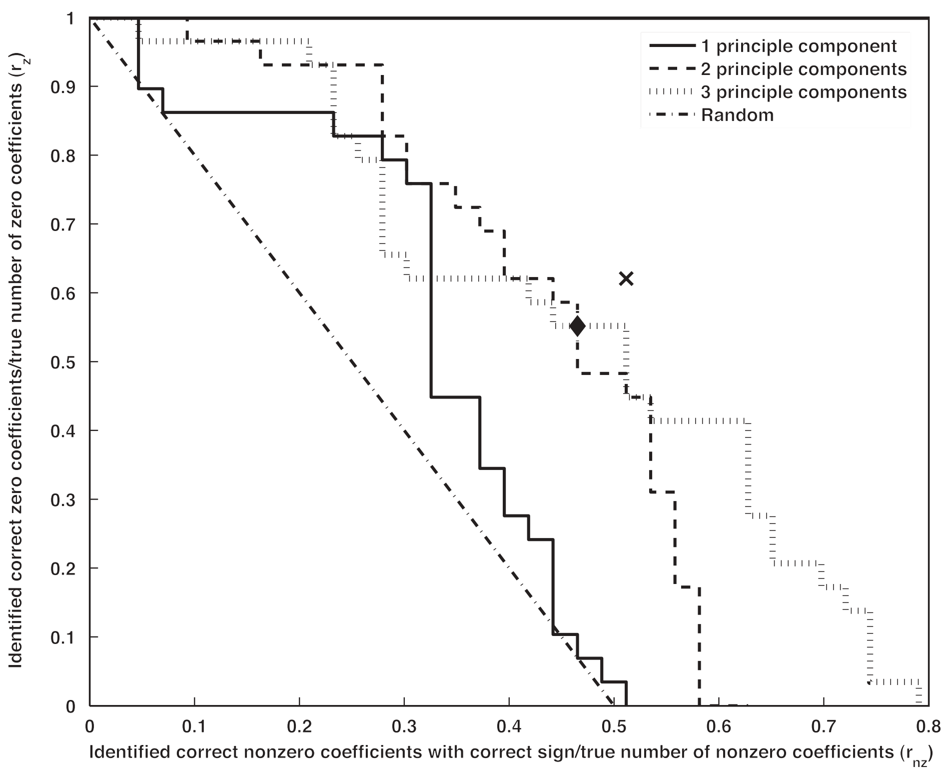

Fig. 5. Plot of average of r_{nz} versus r_z obtained by comparing the predicted network of nine genes in SOS network and the network obtained from literature. Cross (X) corresponds to the value of r_z and r_{nz} obtained by comparing network obtained in our previous work with the network obtained from literatue. Diamond () corresponds to the performance when we used the information from our previous work (Gardner *et al*., 2003) that each gene is connected with other five genes and set four weakest connection in each row of $\hat{A}$ to zero. $r_z = 1$ line corresponds to the ideal curve.

even when the compound does not directly affect the transcriptional response.

Our model is scalable to large networks, thanks to the dimensional reduction and smoothing and interpolation, and to the reduced number of experiments it requires, since only one perturbation experiment is necessary. We are currently applying the algorithm to infer a network from whole genome time series microarrays in mammalian primary cells.

There are two main innovative aspects in our algorithm: (1) we propose an experimental and computational methodology to infer the gene regulatory network from time expression data in which a specific gene of interest is present. This can be achieved either by directly perturbing the gene using an inducible vector for its overexpression or downregulation, or through a compound known to activate the pathway of interest (i.e. Norfloxacin to activate the SOS-pathway as shown in this paper); (2) the approach can be used to speed up the drug discovery process, in that, it is able to predict for an unknown compound, its direct molecular targets from gene expression data following treatment. Importantly, our algorithm requires a limited amount of data as compared with Dynamic Bayesian network and Bayesian network. These methods are very powerful when large number of data points are available. In addition, to our knowledge, DBN and BN have never been applied to infer the targets of a compound or of perturbation from gene expression data.

ACKNOWLEDGEMENTS

We wish to acknowledge Prof. James Collins at the Department of Biomedical Engineering, Boston University, for proposing the idea of single perturbations to infer networks, Prof. Timothy S Gardner at Boston University and Dr Jamey Wierzbowski at Cellicon Biotechnologus, Inc. for discussion and providing the experimental data. This work was supported by a 'FondazioneTelethon' Grant TDDP17TELB and the 'Scuola Europea di Medicina Molecolare—SEMM' Grant for Mr Mukesh Bansal.

Conflict of Interest: none declared.

REFERENCES

Arkin,A. *et al*. (1997) A test case of correlation metric construction of a reaction pathway from measurements. *Science*, **277**, 1275–1279.

Basso,K. *et al*. (2005) Reverse engineering of regulatory networks in human B cells. *Nat. Genet*., **37** (4), 382–390.

Davidson,E.H. *et al*. (2002) A genomic regulatory network for development. *Science*, **295**, 1669–1678.

de Boor,C. (2001) *A practical Guide to Splines*. Springer, NY.

de Jong,H. (2002) Modeling and simulation of genetic regulatory systems: a literature review. *J. Comp. Biol*., **9**, 67–103.

di Bernardo,D. *et al*. (2005) Chemogenomic profiling on a genome-wide scale using reverse-engineered gene networks. *Nat. Biotechnol*., **23**, 377–383.

Gardner,T. *et al*. (2003) Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, **301**, 102–105.

Gautier,L. *et al*. (2004) Affy–analysis of Affymetrix GeneChip. data at the probe level. *Bioinformatics*, **20**, 307–315.

Hartemink,A.J. *et al*. (2002) Combining location and expression data for principled discovery of genetic regulatory network. *Pacific Symp. Biocomput*., **7**, 437–449.

Ideker,T. *et al*. (2001) Integrated genomic and proteomic analyses of a systematically perturbed metabolic network. *Science*, **292**, 929–934.

Lay,D.C. (2002) *Linear Algebra and Its Applications*. Addison Wesley, NY.

Lee,T.I. *et al*. (2002) Transcriptional regulatory networks in *Saccharomyces cerevisiae*. *Science*, **298**, 799–804.

Levchenko, A., and Iglesias, P.A. *et al*. (2002) Models of eukaryotic gradient sensing: application to chemotaxis of amoebae and neutrophils. *Biophys. J*., **82**, 50–63.

Liang,S. *et al*. (1998) REVEAL, a general reverse engienering algorithm for inference of genetic network architectures. *Pac. Symp. Biocomput*., **3**, 18–29.

Ljung,L. (1999) *System Identification: Theory for the User*. Prentice Hall, Upper Saddle River, NJ.

Montgomery,D., Peck,E.A. and Vining,G.G. (2001) *Introduction to Linear Regression Analysis*. John Wiley & Sons, Inc., NY.

Murphy,K. (2001) The bayes net toolbox for matlab. *Computing Science and Statistics*, **33**.

Phillips,I. *et al*. (1987) Induction of the SOS response by new 4-quinolones. *J. Antimicrob. Chemother*., **20**, 631–638.

Savageau,M. (2001) Design principles for elementary gene circuits: elements, methods, and examples. *Chaos*, **11**, 142–159.

Shmulevich,I. *et al*. (2002) Probabilistic boolean networks: a rule.based uncertainty model for gene regulatory networks. *Bioinformatics*, **18**, 261–274.

Tegner,J. *et al*. (2003) Reverse engineering gene networks: integrating genetic perturbations with dynamical modeling. *Proc. Natl Acad. Sci. USA*, **100**, 5944–5949.

Tong,A.H.Y. *et al*. (2002) A combined experimental and computational strategy to define protein interaction networks for peptide recognition modules. *Science*, **295**, 321–324.

Wagner,A. (2001) How to reconstruct a large genetic network from n gene perurbations in fewer that n^2 easy steps. *Bioinformatics*, **17**, 1183–1197.

Yeung,M.K.S. *et al*. (2002) Reverse engineering gene networks using singular value decomposition and robust regression. *Proc. Natl Acad. Sci. USA*, **99**, 6163–6168.

Yu,J. *et al*. (2004) Advances to Bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, **20**, 3594–3603.