

# ROBUST IDENTIFICATION OF LARGE GENETIC NETWORKS

D. DI BERNARDO

*TIGEM, Via P Castellino 111, 80131, Naples, Italy*  
*email:dibernardo@tigem.it; Tel: +39 081 6132 229 FAX: +39 081 560 98 77*

T.S. GARDNER, J.J. COLLINS

*Center for BioDynamics and Department of Biomedical Engineering, Boston University, 44 Cummington St., Boston, MA 02215, USA*

Temporal and spatial gene expression, together with the concentration of proteins and metabolites, is tightly controlled in the cell. This is possible thanks to complex regulatory networks between these different elements. The identification of these networks would be extremely valuable. We developed a novel algorithm to identify a large genetic network, as a set of linear differential equations, starting from measurements of gene expression at steady state following transcriptional perturbations. Experimentally, it is possible to overexpress each of the genes in the network using an episomal expression plasmid and measure the change in mRNA concentration of all the genes, following the perturbation. Computationally, we reduced the identification problem to a multiple linear regression, assuming that the network is sparse. We implemented a heuristic search method in order to apply the algorithm to large networks. The algorithm can correctly identify the network, even in the presence of large noise in the data, and can be used to predict the genes that directly mediate the action of a compound. Our novel approach is experimentally feasible and it is readily applicable to large genetic networks.

## 1 Introduction

Temporal and spatial gene expression, together with the concentration of proteins and metabolites, is tightly controlled in the cell. This is possible thanks to complex regulatory networks between these different elements. The identification of these networks would be extremely valuable. Different experimental and computational methods have been proposed to tackle the network identification problem<sup>1,2,3,4,5,6</sup>. Although implemented with some success, they are data intensive and the description of the network they provide is limited. A variety of mathematical models may be used to describe genetic networks<sup>7,8,9</sup>, including Boolean logic<sup>10,11</sup>, Bayesian networks<sup>12</sup>, graph theory<sup>13</sup>, and

ordinary differential equations. We concentrated our efforts on this last model, because 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 function in the network. In addition, it can be applied to the RNA expression measurements obtained from pharmacological perturbations to identify the genes that directly mediate a compound’s bio-activity in the cell. We already developed and tested in vitro an algorithm to identify a genetic network of nine genes, as a set of linear differential equations, starting from measurements of gene expression at steady state following transcriptional perturbations<sup>14</sup>. In what follows we describe a modification of the algorithm to tackle the problem of reverse-engineering large genetic networks.

## 2 Methods

### 2.1 Network model description

A network can be described by a set of ordinary differential equations<sup>7</sup> describing the time evolution of the mRNA concentration of the genes in the network<sup>8</sup>:

$$\dot{\underline{x}} = f(\underline{x}, \underline{u}) \quad (1)$$

where  $\underline{x}$  represents the mRNA concentrations of the genes in the network, and  $\underline{u}$  is a set of transcriptional perturbations. Assuming that the cell under investigation is at equilibrium near a stable steady-state point, we can apply a small perturbation to each of its genes. A perturbation is small if it does not drive the network out of the basin of attraction of its stable steady-state point and if the stable manifold in the neighborhood of the steady-state point is approximately linear. With these assumptions, we can linearize the set of nonlinear rate equations near its stable state-steady point<sup>6</sup>. Thus, for each gene,  $i$ , in a network of  $N$  genes we can write the following equation:

$$\dot{x}_{il} = \sum_{j=1}^N a_{ij}x_{jl} + u_{il} = \underline{a}_i^T \cdot \underline{x}_l + u_{il}, \quad i = 1 \dots N, l = 1 \dots M, \quad (2)$$

where  $x_{il}$  is the mRNA concentration of gene  $i$  following the perturbation in experiment  $l$ ;  $a_{ij}$  represents the influence of gene  $j$  on gene  $i$ ;  $u_{il}$  is an

---

<sup>a</sup>(from now on we will use the following notation:  $\underline{x}$  represents a column vector,  $\underline{x}^T$  is a row vector,  $x$  is a scalar and  $\mathbf{A}$  is a matrix)

external perturbation to the expression of gene  $i$  in experiment  $l$ . For all  $N$  genes, Eqs. 2 can be rewritten in more compact form using matrix notation:

$$\dot{\underline{x}}_l = \mathbf{A} \cdot \underline{x}_l + \underline{u}_l, \quad l = 1 \dots M, \quad (3)$$

where  $\underline{x}_l$  is an  $N \times 1$  vector of mRNA concentrations of the  $N$  genes in experiment  $l$ ,  $\mathbf{A}$  is an  $N \times N$  connectivity matrix, composed of elements  $a_{ij}$ , and  $\underline{u}_l$  is an  $N \times 1$  vector of the perturbations applied to each of the  $N$  genes in experiment  $l$ .

## 2.2 Network Identification

To identify the network, using the model described above, means to retrieve matrix  $\mathbf{A}$ . This is possible if we measure mRNA concentration of all the  $N$  genes at steady state (i.e.,  $\dot{\underline{x}}_l = 0$ ) in  $M$  experiments and then solve the system of equations:

$$\mathbf{A} \cdot \mathbf{X} = -\mathbf{U} \quad (4)$$

where  $\mathbf{X}$  is an  $N \times M$  matrix composed of columns  $\underline{x}_l$ ;  $\mathbf{U}$  is an  $N \times M$  with each column,  $\underline{u}_l$ . Equation 4 can be solved only if  $M \geq N$ . However, the recovered weights,  $\mathbf{A}$ , will be extremely sensitive to noise both in the data,  $\mathbf{X}$ , and in the perturbations,  $\mathbf{U}$ , and thus unreliable unless we overdetermine the system of Eqs. 4. This can be accomplished either by increasing the number of experiments ( $M > N$ ), or, by assuming the maximum number of regulators acting on each gene,  $k$ , is less than  $M$  (i.e., the network is not fully connected<sup>15,6</sup>), thus reducing the number of weights  $a_{ij}$  to be recovered.

## 2.3 Experimental approach

To identify the network we need to perform transcriptional perturbations for each of the genes in the network and to measure the changes at steady state following the perturbation of the mRNA concentrations for each of the genes in the network. In each perturbation experiment, it is possible to overexpress a different one of the genes in the network using an episomal expression plasmid. Then we let the cells grow under constant physiological conditions to their steady state after the perturbation and measure the change in mRNA concentration compared to cells under the same physiological conditions but unperturbed. This can be achieved using microarrays or real time quantitative PCR.

## 2.4 Algorithm.

A genetic network can be described by the system of linear differential equations, Eqs. 2. For each gene  $i$  at steady state ( $\dot{\underline{x}}_{il} = 0$ ) in experiment  $l$ , we can therefore write:

$$-u_{il} = \underline{a}_i^T \cdot \underline{x}_l \quad (5)$$

where  $u_{il}$  is the transcriptional perturbation applied to gene  $i$  in experiment  $l$ ,  $\underline{a}_i^T$  is a row of  $\mathbf{A}$ , and  $\underline{x}_l$  ( $N \times 1$ ) are the mRNA concentrations at steady state following the perturbation in experiment  $l$ . The algorithm assumes that only  $k$  out of the  $N$  weights in  $\underline{a}_i$  for gene  $i$  are different from zero. For each possible combination of  $k$  out of  $N$  weights, the algorithm computes the solution to the following linear regression model:

$$y_{il} = \underline{b}_i^T \cdot \underline{z}_l + \epsilon_{il} \quad (6)$$

where  $y_{il} = -u_{il}$  is the perturbation applied to gene  $i$  in experiment  $l$ ;  $\underline{b}_i$  is a  $k \times 1$  vector representing one of  $\frac{N!}{k!(N-k)!}$  possible combinations of weights for gene  $i$ ;  $\epsilon_{il}$  is a scalar stochastic normal variable with zero mean and variance,  $var(\epsilon_{il})$ , representing measurement noise on the perturbation of gene  $i$  in experiment  $l$ ;  $\underline{z}_l$  is a  $k \times 1$  vector of mRNA concentrations following the perturbation in experiment  $l$ , with added uncorrelated Gaussian noise ( $\underline{\gamma}_l$ ) with zero means and variances  $var(\gamma_{jl})$ . Equation 6 represents a multiple linear regression model with noise  $\eta_{il} = \underline{b}_i^T \cdot \underline{\gamma}_l + \epsilon_{il}$ , with zero mean and variance:

$$var(\eta_{il}) = \sum_{j=1}^k b_{ij}^2 var(\gamma_{jl}) + var(\epsilon_{il}) \quad (7)$$

(if  $\epsilon_{il}$  and  $\underline{\gamma}_l$  are uncorrelated).

If we collect data in  $M$  different experiments, then we can write Eq. 6 for each experiment and obtain the system of equations:

$$\underline{y}_i^T = \underline{b}_i^T \cdot \mathbf{Z} + \underline{\epsilon}_i^T \quad (8)$$

where  $\underline{y}_i$  is an  $M \times 1$  vector of measurements of the perturbation  $y_{il}$  to gene  $i$  in the  $M$  experiments;  $\mathbf{Z}$  is a  $K \times M$  matrix, where each column is the vector  $\underline{z}_l$  for one of the  $M$  experiments;  $\underline{\epsilon}_i$  is an  $M \times 1$  vector of noise in the  $M$  experiments. From Eqs. 8, it follows that a predictor for  $\underline{y}_i$  given the data matrix  $\mathbf{Z}$  is:

$$\hat{\underline{y}}_i^T = \underline{b}_i^T \cdot \mathbf{Z} \quad (9)$$

We chose to minimize the following cost function to find the  $k$  weights,  $\underline{b}_i$ , for gene  $i$ :

$$C_i^k = \sum_{l=1}^M (y_{il} - \widehat{y}_{il})^2 = \sum_{l=1}^M (y_{il} - \underline{b}_i^T \cdot \underline{z}_l)^2 \quad (10)$$

The solution can be obtained by computing the pseudo inverse<sup>16</sup> of  $\mathbf{Z}$ :

$$\widetilde{\underline{b}}_i = (\mathbf{Z} \cdot \mathbf{Z}^T)^{-1} \cdot \mathbf{Z} \cdot \underline{y}_i \quad (11)$$

Note that the solution,  $\widetilde{\underline{b}}_i$ , in Eq. 11 is not the maximum likelihood estimate for the parameters  $\underline{b}_i$  when the regressors  $\mathbf{Z}$  are stochastic variables<sup>17</sup>, but it nevertheless is a good estimate. We select as the best approximation of the weights in Eqs. 2 for gene  $i$ , the one with the smallest least-squares error,  $C_i^k$ , among the ( $N$  choose  $k$ ) possible solutions  $\widetilde{\underline{b}}_i$ .

### 2.5 Estimation of the variance of the parameters.

We now turn to the estimation of the variance on the estimated parameters  $\widetilde{\underline{b}}_i$  and the calculation of the goodness of fit. If, in each experiment, the noise is uncorrelated and Gaussian with zero mean and known variance, then the covariance matrix of the estimated parameters  $\widetilde{\underline{b}}_i$  is<sup>16</sup>:

$$Cov(\widetilde{\underline{b}}_i) = (\mathbf{Z} \cdot \mathbf{Z}^T)^{-1} \cdot \mathbf{Z} \cdot \Sigma_\eta \cdot \mathbf{Z}^T \cdot (\mathbf{Z} \cdot \mathbf{Z}^T)^{-1} \quad (12)$$

where  $\Sigma_\eta$  is an  $M \times M$  diagonal matrix with diagonal elements equal to the noise variance for gene  $i$  in the  $M$  experiments,  $var(\eta_{i1}) \dots var(\eta_{im})$ . We assume that we can estimate  $var(\eta_{il})$  in each experiment using the parameters  $\widetilde{\underline{b}}_i$  estimated with Eq. 11 and substituting in Eq. 7:

$$var(\eta_{il}) = \sum_{j=1}^k \widetilde{b}_{ij}^2 var(\gamma_{jl}) + var(\epsilon_{il}) \quad (13)$$

We can now compute the variances of the parameters using Eq. 12, where  $\Sigma_\eta$  is computed using Eq. 13. The quantities  $var(\gamma_{jl})$  and  $var(\epsilon_{jl})$  are supposed to have been estimated experimentally. We can also compute a goodness of fit test using the Chi-squared statistic:

$$\chi_i^2 = \sum_{l=1}^M \frac{(y_{il} - \widetilde{\underline{b}}_i^T \cdot \underline{z}_l)^2}{var(\eta_{il})} \quad (14)$$

## 2.6 Modification of the algorithm for large networks.

For a network of  $N$  genes, with  $k \leq N$  connection for each gene, we need to solve Eq. 6 for all the possible  $\frac{n!}{k!(n-k)!}$  combinations of  $k$  genes and then select the one that fits the data best. For large networks, this exhaustive approach is unfeasible since there are too many combinations to test. We used a heuristic search method (Forw-TopD-reest-K<sup>18</sup>) to reduce the number of solutions to test. We first compute all the possible solutions with single connections ( $k=1$ ) as described in sec. 2.4. We then select the best  $D$  solutions (the ones with the smallest least squared error), and only for these intermediate solutions, we compute all the possible solutions with an additional connection. Then we again select the best  $D$  solutions, and so on until the number of connections found for each gene is  $k$ . We implemented this approach using a value of  $D = 5$ .

## 2.7 Target prediction.

It is possible to use the recovered network  $\tilde{\mathbf{A}}$  to deconvolve the results of an experiment, i.e., to recover the unknown perturbations  $\underline{u}_0$  in an experiment, given the measurements of the response to that perturbation,  $\underline{x}_0$ . The predicted perturbations  $\tilde{\underline{u}}_0$  can be computed from:

$$\tilde{\underline{u}}_0 = -\tilde{\mathbf{A}} \cdot \underline{x}_0. \quad (15)$$

The variance on the estimated perturbation of gene  $i$  can be computed as<sup>19</sup>:

$$var(\tilde{u}_{0_i}) = \underline{x}_0^T \cdot (\mathbf{Z} \cdot \mathbf{Z}^T)^{-1} \cdot \mathbf{Z} \cdot \Sigma_\eta \cdot \mathbf{Z}^T \cdot (\mathbf{Z} \cdot \mathbf{Z}^T)^{-1} \cdot \underline{x}_0 + \sum_{j=1}^k \tilde{b}_{ij}^2 var(x_{0_j}) \quad (16)$$

Using the variance of the estimated perturbation, we perform a  $t$  test to test the hypothesis that the predicted perturbations are significantly different from zero.

## 2.8 Simulated data

To test the algorithm on a realistic data set, we generated 10 random networks of  $N = 100$  genes with an average of  $k = 10$  connections for each gene. Each network was represented by a full rank sparse matrix  $\mathbf{A}$  ( $N \times N$ ), as described in sec.2.2. We made sure that all the eigenvalues of these random sparse matrices had a real part less than 0 to ensure that the dynamical systems described by

them were stable. The data set  $\mathbf{X}$  ( $N \times M$ ) was obtained by inverting eq.4 to obtain:

$$\mathbf{X} = -\mathbf{A}^{-1} \cdot \mathbf{U} \quad (17)$$

where  $\mathbf{U}$ , ( $N \times M$ ), were the perturbations in  $M = 100$  experiments. We chose  $\mathbf{U}$  to be a diagonal identity matrix. This is equivalent to say that in each experiment only 1 out of the 100 genes was perturbed by increasing its transcription rate by 1. The data the algorithm needs to identify the network  $\mathbf{A}$  are the gene expression data matrix  $\mathbf{X}$  and the perturbation matrix  $\mathbf{U}$ . We added white gaussian noise to each data matrix. For the perturbation matrix  $\mathbf{U}$ , the standard deviation of the noise was fixed to  $\sigma_u = 0.3$  (i.e. 30% of the magnitude of the perturbation), while for the gene expression data matrix it varied from  $\sigma_x = 0.1 * \bar{X}$  to  $\sigma_x = 0.5 * \bar{X}$  where  $\bar{X}$  represents the average of the absolute values of the elements in  $\mathbf{X}$ . The performance of the algorithm was tested using these data with the different noise levels in order to identify the network  $\mathbf{A}$ . We used two measures of performance: coverage (correct connections in the recovered network model / total connections in the true network) and false positives (incorrect connections in the recovered model / total number of recovered connections).

In order to test the ability of the identified network to predict unknown perturbations given the gene expression data, for each random network, we generated 10 additional experiments in which 3 genes, randomly chosen out of 100 genes, were perturbed simultaneously. We computed the ability of the recovered network to predict which genes had been perturbed, using the method described in 2.7.

The algorithm described in this section was fully implemented in MATLAB environment. For a network of 100 genes, the algorithm took 50s to run on a Pentium III with a clock speed of 1.2 Ghz.

### 3 Results

#### 3.1 Identification of networks

Figure 1 shows the average performance of the algorithm across the 10 random networks described in sec. 2.8 for noise levels ranging from 10% to 50%. Since the algorithm reports also the variance of the identified elements in matrix  $\mathbf{A}$ , it is possible to compute a  $p$  value for each of its elements  $a_{ij}$ . We used a Student  $t$  distribution to test the hypothesis that the element  $a_{ij}$  identified

by the algorithm is significantly different from 0. This is equivalent to test whether gene  $i$  is significantly regulated by gene  $j$ . Figure 1 reports also the coverage and false positives in the case we consider significantly different from 0 only those elements with a  $p$ -value  $\leq 0.05$  (dashed lines).

### 3.2 Target prediction

Figure 2 shows the coverage (genes correctly identified as perturbed by the network model / total number of perturbed genes) and the percentage of false positives (genes wrongly identified as perturbed by the network model / total number of genes identified as perturbed by the network model) for noise levels ranging from 10% to 50% averaged across the 10 random networks and across 10 perturbation experiments, as described in sec.2.8.

In Figure 2, open bars show coverage and false positives considering the predicted perturbations correct only if they have a  $p$ -value  $\leq 0.01$ , black bars show the same quantities for a  $p$ -value  $\leq 0.1$ .

## 4 Discussion

The algorithm we propose requires only measurements of mRNA concentrations at steady state following transcriptional perturbations. Therefore, the experimental time and costs involved in the procedure are affordable. This is a very useful feature of our approach. Another essential feature is its robustness to measurement noise. Measurements of mRNA concentration using microarrays are noisy, and therefore an algorithm to identify networks is useful only if it is robust to such noise. We showed that the recovered network can be used for target prediction, this can be very useful for drug discovery. Using measurements of mRNA concentration changes at steady state following the application of a compound to a cell population, we can predict which are the direct targets of that drug in a large gene network using the recovered network model.

The recovered network model,  $\mathbf{A}$ , is a linear representation of a nonlinear system. Nonlinear behaviours that are sometimes exhibited by gene, protein, and metabolite networks, including bifurcations, thresholds, and multistability, cannot be described by  $\mathbf{A}$ . Nevertheless, the linear approximation is topologically equivalent to the nonlinear system near a steady-state point. Therefore, to apply our algorithm, it is necessary to remain near a single steady state during the course of all experiments. From a practical perspective, this means that cells must be maintained under consistent and constant environmental

and physiological conditions, and the applied perturbations must be relatively small. If these conditions are not met, the recovered model may contain a certain degree of nonlinear error, or, in the extreme, it may not be possible to adequately fit a linear model.

In practice, it is generally straightforward to keep the cells in a constant environmental and physiological state, but due to the presence of measurement noise, it can be challenging to meet the condition of small perturbations. For errors due to noise, we can improve the signal-to-noise ratio ( $S/N$ ) by boosting the size of the perturbations. However, larger perturbations can lead to larger nonlinear errors. Thus, the experimenter must identify an acceptable balance between noise and nonlinear error.

The network should be sparse for the method to work. Our algorithm can be successfully applied as long as the real connectivity of the network (i.e. number of connections per gene) is less than the number of perturbation experiments. An exact threshold for the maximum number of connections that can be recovered correctly with this algorithm cannot be computed because this will depend on the noise level of the data. For noise-free data, the maximum connectivity will be equal to the number of perturbations experiments performed.

Our approach to inferring genetic networks has been shown to work in vivo for small networks<sup>14</sup>. The computer simulations here described suggest that a modified version of the algorithm will work also for large genetic networks. We showed that even with considerable noise, it is still possible to recover 60% of the real network with less than 10% of wrongly identified connections. This is important in biological research because it can provide a first draft of the map of interaction among hundreds of genes whose function or regulation is partly or completely unknown. Also the network recovered with the algorithm can predict the direct targets of an unknown perturbation with a specificity of approx. 80%, even in the presence of large noise. This would greatly help in the identification of the real targets of a novel molecule in a large network, by greatly reducing the targets to be tested experimentally. In addition, the experiments required to generate the data needed by the algorithm are feasible and economically affordable also for large networks.

1. A. H. Y. Tong, B. Drees, Nardelli G., G. D. Bader, B. Brannetti, L. Castagnoli, M. Evangelista, S. Ferracuti, B. Nelson, S. Paoluzi, M. Quondam, A. Zucconi, C. W. V. Hogue, S. Fields, C. Boone, and G. Cesareni, *Science* **295**, 321–324 (2002).
2. T. I. Lee, N. J. Rinaldi, F. Robert, D. T. Odom, Z. Bar-Joseph, G. K. Gerber, N. M. Hannett, C. T. Harbison, C. M. Thompson, I. Simon, J. Zeitlinger, E. G. Jennings, H. L. Murray, D. B. Gordon, B. Ren, J. J. Wyrick, J. B. Tagne, T. L. Volkert, E. Fraenkel, D. K. Gifford, and R. A. Young. *Science* **298**, 799–804 (2002).
3. T. Ideker, V. Thorsson, J. A. Ranish, R. Christmas, J. Buhler, J. K. Eng, R. Bumgarner, D. R. Goodlett, R. Aebersold, and L. Hood, *Science* **292**, 929–934 (2001).
4. E. H. Davidson, J. P. Rast, P. Oliveri, A. Ransick, C. Calestani, et al. *Science* **295**, 1669–1678 (2002).
5. A. Arkin, P. D. Shen, and J. Ross, *Science* **277**, 1275–1279 (1997).
6. M. K. S. Yeung, J. Tegnér, and J. J. Collins, *Proc. Natl. Acad. Sci. U.S.A.* **99**, 6163–6168 (2002).
7. H. de Jong, *J. Comp. Biol.* **9**, 67–103 (2002).
8. M. A. Savageau, *Chaos*, 142–159 (2001).
9. A. Levchenko, and A. Iglesias, *Biophys J*, 50–63 (2002).
10. I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang, *Bioinformatics* **18**, 261–274 (2002).
11. S. Liang, S. Fuhrman, and R. Somogyi, *Proc. Pacific Symp. Biocomp.* **3**, 18–29 (1998).
12. A. J. Hartemink, D. K. Gifford, T. S. Jaakkola, and R. A. Young, *Proc. Pacific Symp. Biocomp.* **7**, 437–449 (2002).
13. A. Wagner, *Bioinformatics* **17**, 1183–1197 (2001).
14. T. S. Gardner, D. di Bernardo, D. Lorenz, and J. J. Collins, *Science* **301**, 102–105 (2003).
15. Z. N. Oltvai, and A. L. Barabási, *Science* **298**, 763–764 (2002).
16. L. Ljung, *System Identification: Theory for the User*. Prentice Hall, Upper Saddle River, NJ, (1999).
17. W. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK, (1993).
18. E. P. van Someren, L. F. A. Wessels, M. J. T. Reinders, and E. Backer, *Proc. 2<sup>nd</sup> Int. Conf. Systems Biol.*, 222–230 (2001).
19. D. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*. John Wiley & Sons, Inc., New York, (2001).

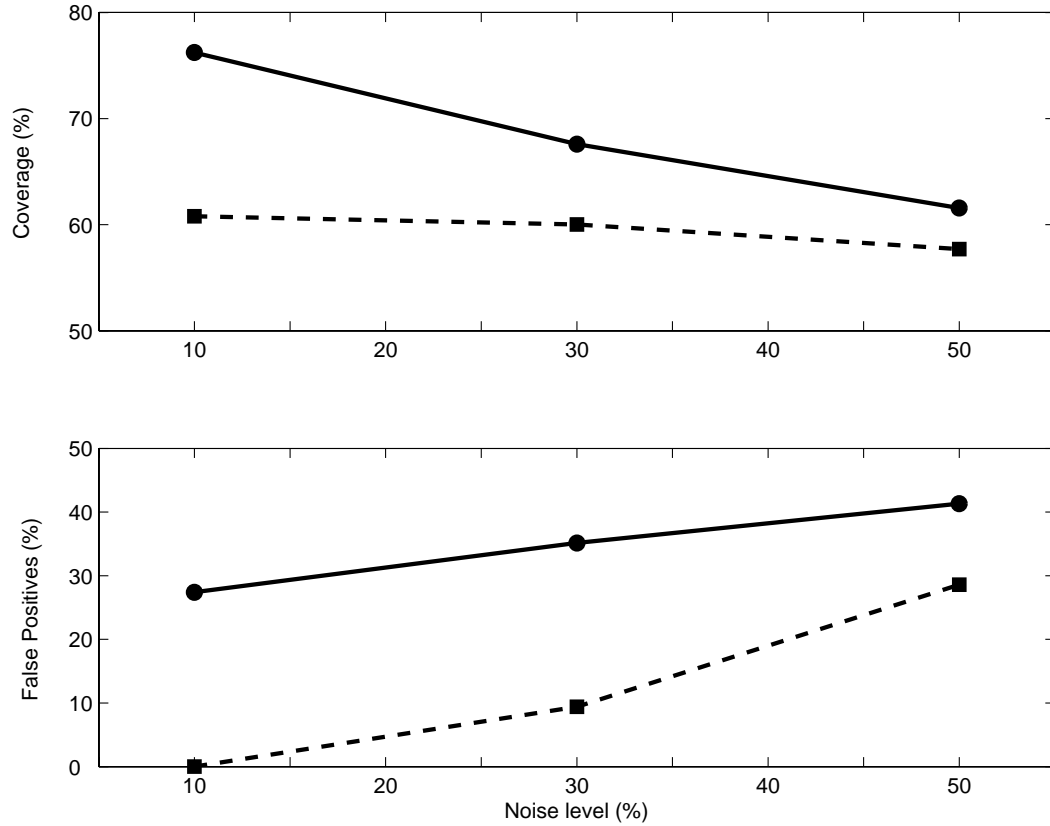


Figure 1: Model recovery performance for simulations. Perturbations of magnitude  $u_i = 1$  (arbitrary units) were applied to ten randomly connected networks of one hundred genes with an average of ten regulatory inputs per gene. For each perturbation to each random network, the mRNA concentrations at steady state were calculated, and normally-distributed, uncorrelated noise was added both to the mRNA concentrations and to the perturbations to represent measurement error. The noise (noise =  $S_x/\mu_x$ , where  $S_x$  is the standard deviation of the mean of  $x$ ,  $\mu_x$ ) on the perturbations was set to 30%. The noise on the mRNA concentrations was varied from 10% to 50%. The average coverage, top panel, (correct connections in the recovered network model / total connections in the true network) and average false positives, bottom panel, (incorrect connections in the recovered model / total number of recovered connections) were calculated across all the models recovered. Filled circles: All the recovered connections were included in the computation of coverage and false positives. Filled squares: Only the recovered connections with a  $p$ -value  $\leq 0.05$  were included in the computation.

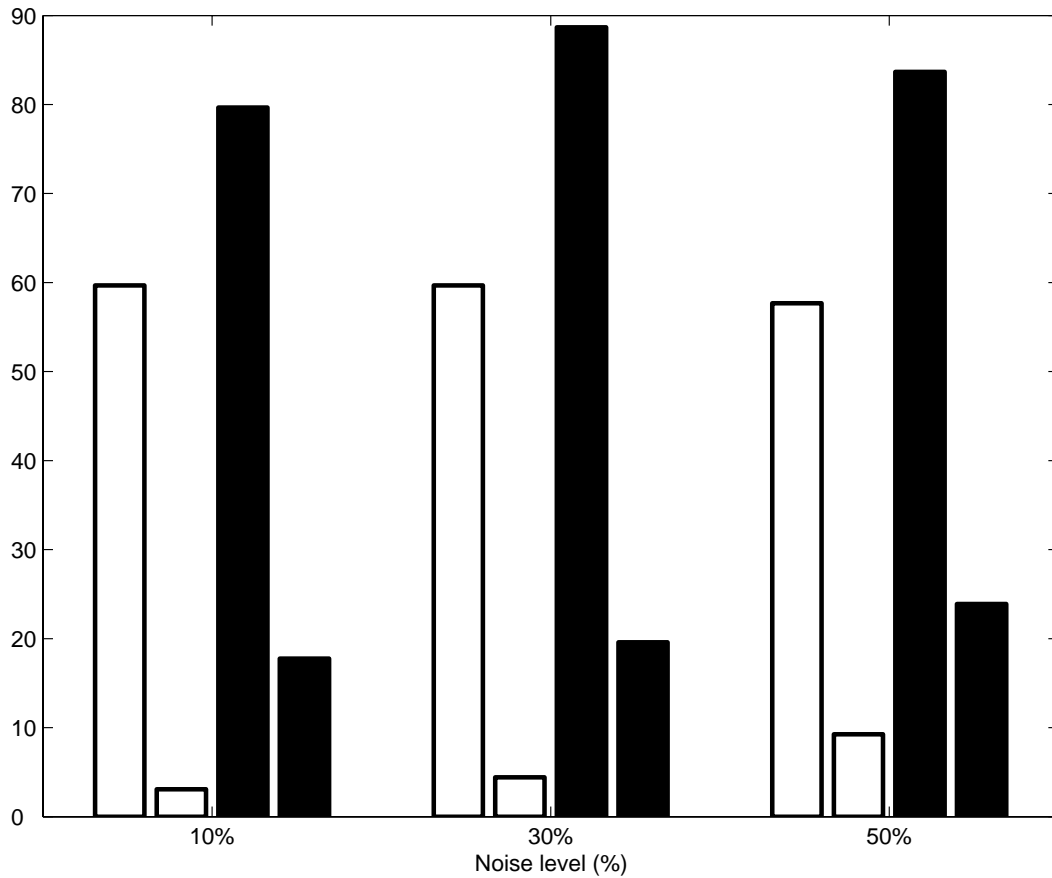


Figure 2: Perturbation prediction performance for simulations.

Three genes were randomly and simultaneously perturbed. Using the steady state measurements following the perturbation, the network model was used to predict which genes had been perturbed. This experiment was repeated ten times for each one of ten different random networks of one hundred genes with an average of ten regulators per gene.

Coverage (genes correctly identified as perturbed by the network model / total number of perturbed genes) and the percentage of false positives (genes wrongly identified as perturbed by the network model / total number of genes identified as perturbed by the network model) for noise levels ranging from 10% to 50% averaged across the ten random networks and the ten perturbation experiments. Open bars: Coverage (tall) and false positives (short) considering correct only predictions with a  $p$ -value  $\leq 0.01$ . Filled bars: Coverage (tall) and false positives (short) considering correct only predictions with a  $p$ -value  $\leq 0.1$ .