Reverse engineering gene networks using singular value decomposition and robust regression

M. K. Stephen Yeung, Jesper Tegnér[†], and James J. Collins[‡]

Center for BioDynamics and Department of Biomedical Engineering, Boston University, Boston, MA 02215

Edited by Charles S. Peskin, New York University, Hartsdale, NY, and approved March 13, 2002 (received for review October 29, 2001)

We propose a scheme to reverse-engineer gene networks on a genome-wide scale using a relatively small amount of gene expression data from microarray experiments. Our method is based on the empirical observation that such networks are typically large and sparse. It uses singular value decomposition to construct a family of candidate solutions and then uses robust regression to identify the solution with the smallest number of connections as the most likely solution. Our algorithm has $O(\log N)$ sampling complexity and $O(N^4)$ computational complexity. We test and validate our approach in a series of *in numero* experiments on model gene networks.

With recent advances in cDNA and oligonucleotide microarray technologies (1), it has become possible to measure mRNA expression levels on a genome-wide scale. Data thus collected provide valuable descriptions of gene activities under various biochemical (2) and physiological (3) circumstances and allow one to reverse-engineer the gene networks, i.e., to infer the underlying network structures from experimental measurements. However, naturally occurring gene regulatory networks are embedded in genomes that typically consist of thousands of genes. To extract the topology of such networks and hence isolate the functional subnetworks represents a computationally daunting task; it also requires a very large amount of experimental data, which are expensive to obtain.

To circumvent this problem of data deficiency, many current research efforts have focused on clustering, i.e., grouping genes into hierarchical functional units based on correlations in expression patterns (3–8). This hierarchical approach has been fruitful in identifying coregulated genes in certain functional units (3–6). It has also been generalized to self-organizing maps (7) and supervised learning schemes (8) to cope with the sensitivity to noise and other deficiencies intrinsic to hierarchical clustering (9), at the cost of increasing computational cost. However, a fundamental shortcoming of such clustering schemes is that they are based on the assumptions that: (*i*) gene regulatory networks are hierarchical in structure (3–6), and (*ii*) genes performing related biological functions exhibit similar expression patterns (and vice versa). These assumptions may not always be valid. At a structural level, there are data suggesting that gene regulatory networks are not strictly hierarchical in nature; rather, they are interwoven like a web (10), as in the cases of metabolic (11) and protein networks (12), with multiple pathways for similar functions to provide redundancy to protect against mutations and other deleterious effects (13). At a dynamical level, mRNA and protein expression levels for certain genes may not be correlated (14), suggesting a similar lack of strict correlation between gene expression and function. Therefore, although clustering is useful on a local scale to identify isolated coexpressing units, it is not suitable for large-scale reverse engineering.

Recently, there have been attempts to reconstruct models for gene regulatory networks on a global, genome-wide scale using ideas from system identification (15), such as genetic algorithms (16), neural networks (17), and Bayesian models (18). Although useful in specific contexts, these approaches are of restricted scope, as they typically require a large amount of data and computation to generate connectivity maps for large networks, such as those of genomic scales. To overcome these problems of data shortage and computational inefficiency, several researchers (19–22) have adopted a linear model and have used singular value decomposition (SVD) (23) to reverse-engineer the network architecture. As we will explain in greater detail below, although SVD provides a useful and condensed description of the data, it alone may not correctly identify the connectivity matrix and therefore may not accurately predict the behavior of the gene networks in response to novel stimuli. The method of SVD has to be supplemented by extra conditions, based on biological knowledge, to recover the network topology correctly.

Here we propose such a scheme to identify the entire network structure in gene regulatory networks on a genome-wide scale. Our approach is built on previous work using SVD (19–22) and is based on an insight provided by earlier studies on gene regulatory networks (24, 25) and bioinformatics databases (11, 12), namely, that gene networks in most biological systems are sparse. Thus, we first use SVD to construct a family of candidate networks, all being consistent with the experimental data, and then uses robust regression (26) to identify the sparsest network in this family as the most likely solution. As such, our scheme has $O(\log N)$ sampling complexity and $O(N^4)$ computational complexity. Much in the spirit of systems biology (27), our goal is to extract the gene regulatory networks on a global scale and to do so efficiently, in order to identify individual subnetworks in a first draft of the topology of the entire network, on which further, more local, analysis can be based.

Method

The method we propose to reverse-engineer gene networks consists of two steps. We first use SVD to construct a set of feasible solutions that are consistent with the measured data and then we use robust regression to select the sparsest one as the solution.

For simplicity, we will consider only systems that are operating near a steady state, so that the dynamics can be approximated by a linear system of ordinary differential equations:

$$\dot{x}_i(t) = -\lambda_i x_i(t) + \sum_{j=1}^{N} W_{ij} x_j(t) + b_i(t) + \xi_i(t) \text{ for } i = 1, 2, \ldots, N. \quad [1]$$

Here the x_is are the concentrations of the mRNAs that reflect the expression levels of the genes, the λ_is are the self-degradation rates, the b_is are the external stimuli, and the ξ_is represent noise. The matrix elements, W_{ij}s, which are real numbers, describe the type and strength of the influence of the jth gene on the ith gene, with a

This paper was submitted directly (Track II) to the PNAS office.

Abbreviation: SVD, singular value decomposition.

[†]Present address: Stockholm Bioinformatic Center, SCFAB, S-10691 Stockholm, and Department of Numerical Analysis and Computer Science, Royal Institute of Technology, S-10044 Stockholm, Sweden.

[‡]To whom reprint requests should be addressed at: Center for BioDynamics and Department of Biomedical Engineering, Boston University, 44 Cummington Street, Boston, MA 02215. E-mail: jcollins@bu.edu.

The publication costs of this article were defrayed in part by page charge payment. This article must therefore be hereby marked "*advertisement*" in accordance with 18 U.S.C. §1734 solely to indicate this fact.

positive sign indicating activation, a negative sign indicating repression, and a zero indicating no interaction.

In an experiment, we can apply a prescribed stimulus $(b_1, b_2, \ldots, b_N)^T$ and use a microarray to measure simultaneously the concentrations of all of the N different mRNAs, i.e., $(x_1, x_2, \ldots, x_N)^T$. Repeating this procedure M times, we get M measurements and can tabulate the results as

$$X_{N\times M} := \begin{pmatrix} x_1^1 & x_1^2 & \cdots & x_1^M \\ x_2^1 & x_2^2 & \cdots & x_2^M \\ \vdots & \vdots & \ddots & \vdots \\ x_N^1 & x_N^2 & \cdots & x_N^M \end{pmatrix}.$$

Here the subscript i indexes individual genes, and the superscript j denotes experiment number. That is, x_i^j is the concentration of the ith mRNA for the jth trial, with similar notations for $\dot{X}$ and B. Then we can rewrite Eq. **1** as

$$\dot{X}_{N\times M} = A_{N\times N}X_{N\times M} + B_{N\times M},$$

where we have neglected noise and absorbed the self-degradation rates λ_is into the coupling constants W_{ij}s to simplify notation. That is, $A_{ij} := W_{ij} - \delta_{ij}\lambda_i$.

The goal of reverse engineering is to use the measured data B, X, and $\dot{X}$ to deduce A and hence the connectivity matrix W. In this context, we may take the transpose of the system and rewrite it as

$$(X^T)_{M\times N}(A^T)_{N\times N} = [(\dot{X}^T)_{M\times N} - (B^T)_{M\times N}]$$

to emphasize the fact that A is the unknown. If $M = N$ and X is full-ranked, we can simply invert the matrix X to find A. However, typically $M \ll N$ because of the high cost of perturbations and measurements. We therefore have an underdetermined problem. One way to get around this is to use SVD (23) to decompose X^T into

$$(X^T)_{M\times N} = U_{M\times N}\cdot W_{N\times N}\cdot (V^T)_{N\times N},$$

where U and V are each orthogonal:

$$U^T\cdot U = V^T\cdot V = I_{N\times N},$$

with I being the identity matrix, and W is diagonal:

$$W = \begin{pmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_N \end{pmatrix}.$$

Without loss of generality, we may assume that all nonzero elements of w_k are listed at the end, i.e., $w_1, w_2, \ldots, w_L = 0$ and $w_{L+1}, w_{L+2}, \ldots, w_N \neq 0$, where $L := \dim(\ker(X^T))$. Then one particular solution for A is:

$$A = A_0 := (\dot{X} - B)\cdot U\cdot \mathrm{diag}\left(\frac{1}{w_j}\right)\cdot (V^T), \qquad \textbf{[2]}$$

with $1/w_j$ taken to be zero if $w_j = 0$, whereas the general solution is given by the affine space

$$A = A_0 + C\cdot V^T, \qquad \textbf{[3]}$$

with $C = (c_{ij})_{N\times N}$, where c_{ij} is zero if $j > L$ and is otherwise an arbitrary scalar coefficient. This family of solutions in Eq. **3** represents all the possible networks that are consistent with the microarray data. Among these solutions, the particular solution A_0 is the one with the smallest L_2 norm.

This idea of using SVD to reverse-engineer gene networks with a limited amount of data is not new (19–22). Nevertheless, these earlier efforts stopped at Eq. **2** and took A_0 as the solution, a choice that may not always recover the connectivity matrix correctly, as we shall see in Example 1.

Because SVD leads to nonunique solutions, we need additional constraints to isolate the true solution from the entire family in Eq. **3**. Many choices are possible, and the particular choice depends on our knowledge of the biological system. For example, if we know *a priori* that certain genes are functionally related, we may impose this as a constraint to sieve through the family of solutions given by Eq. **3**. Here we adopt the viewpoint that we have no prior knowledge of the network. In such cases, we may rely on insights provided by earlier works on gene regulatory networks (24, 25) and bioinformatics databases (11, 12), which suggest that naturally occurring gene networks are sparse, i.e., generally each gene interacts with only a small percentage of all the genes in the entire genome. It is in this global sense, on a genome-wide scale, that the entire network, which encompasses all the individual gene regulatory networks, is sparse; and it is this feature that we will exploit to resolve the ambiguity introduced by SVD.

Because the family Eq. **3** represents *all* connectivity matrices that are consistent with the measurement data, we may concentrate on this set to look for the true connectivity matrix. Imposing sparseness on the family of solutions given by Eq. **3** means that we need to choose the coefficients c_{ij} to maximize the number of zero entries in A. This is a nontrivial problem, because we do not know in advance which entries are nonzero. As we shall see in Example 1, the solution A_0 as found by SVD alone may not be close to the true solution. We therefore cannot assume that a small entry in A_0 corresponds to a zero entry in the true solution (28). It is necessary to explore the family of solutions in Eq. **3**. Nonetheless, a brute-force approach, enumerating all possibilities to see which choice will lead to a self-consistent solution (29), is computationally costly as it will take $O(N!/(k!(N-k)!))$ operations to solve a system of N equations with k nonzero entries whose locations are unknown. A more efficient method is needed for large N.

Our idea is to consider the dual problem, where we proceed as if we could have all entries being zero, namely, setting $A = 0$ in Eq. **3** to obtain

$$C\cdot V^T = -A_0,$$

which is an overdetermined problem as there are only NL variables (c_{ij} for $j \leq L$), whereas there are N^2 equations. In this context, the solution $A = A_0$, given by Eq. **2**, is the closest solution in the L_2 sense. However, that is not what we want. Instead, we want to satisfy as many equations as possible. Viewed as such, our task is equivalent to the problem of finding the exact-fit plane (30) in robust statistics, where we try to fit a hyperplane to a set of points containing a few outliers, with the objective being to pass through as many points as possible. This is a well-studied problem, and many methods have been developed to do this, each with its merits and shortcomings (26). Here we have chosen L_1 regression (31), where the figure of merit is the minimization of the sum of the absolute values of the errors, for its efficiency. Among the many numerical algorithms for L_1 regression, we have adopted the simplex method in refs. 32 and 33 for its simplicity. This algorithm takes approximately $O(pn^2)$ computations to solve a regression problem with n data points and p parameters (31). Here we have $n = N$ and $p = L \approx N - M$ for each row in Eq. **3**. This relation implies that more experimental data points can relieve the burden on computation, whereas a small set of experimental data points will demand more computations to resolve the uncertainty. Because experiments are much more expensive than computations, it is desirable to reduce M even though that will increase p. As we shall demonstrate in the *in numero* experiments below, it is possible to reverse-engineer networks with $M = O(\log N)$ experimental measurements. In such cases, we have $M \ll N$ and so $p = O(N)$. It therefore takes $O(N^3)$ computations to reconstruct one row of the connectivity matrix, and the recovery of the entire matrix is an $O(N^4)$ process.

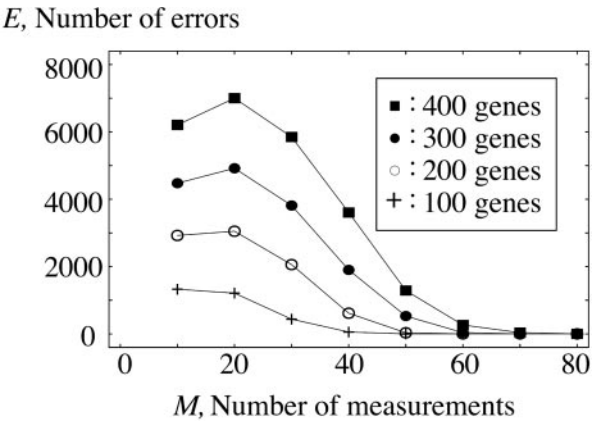

Fig. 1. Number of errors, *E*, made by the reverse engineering scheme as a function of *M*, the number of measurements, for four linear networks of the form **1** with different sizes *N*.

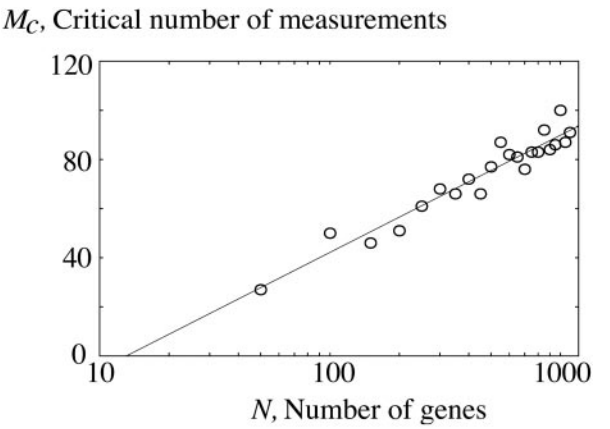

Fig. 2. Critical number of measurements, M_c, required to recover the entire connectivity matrix correctly, versus *N*, the size of the network for linear systems of the form **1**. Circles: numerical data. Line: least-squares fit of the form $M_c = a + b \log N$.

***In Numero* Experiments.** In this section, we report on three numerical experiments that we have conducted to test our reverse engineering scheme. In the first experiment, we calibrated the scheme using a large sparse linear system. In the second experiment, we applied the scheme to a nonlinear model for a cascade of repressors. In the third experiment, we tested the scheme with a large sparse nonlinear network consisting of both activators and repressors. As we shall show, in all three cases, we were able to recover the network connectivity using a small number (compared to the network size) of measurements even in the presence of noise.

Example 1: A large sparse linear network. In this first experiment, we considered a linear system governed by Eq. **1**. To generate the connectivity matrix *W*, we proceeded row by row. For each row, we randomly picked an integer *k* from a power-law distribution with a cutoff (12) at $k_{\max}$ with $k_{\max} \ll N$. We then randomly selected *k* entries and assigned each of them a nonzero value randomly chosen from a uniform distribution. The condition $k_{\max} \ll N$ ensures sparseness.

To test the reverse engineering scheme, we perturbed the system **1** with transient random perturbations $b_i(t)$. While the system was relaxing back to the steady state, we took measurements for *X* and estimated $\dot{X}$ by linear interpolation. We repeated this process *M* times to collect *M* data points and applied the reverse engineering scheme to attempt to reconstruct the connectivity matrix. The connectivity matrix thus reconstructed, denoted by A_R, was then compared with the true connectivity matrix $A_T = W - \Lambda$, entry by entry. Specifically, we measured the error by counting the number of discrepancies:

$$E := \sum_{i=1}^{N} \sum_{j=1}^{N} e_{ij}, \qquad [4]$$

where

$$e_{ij} := \begin{cases} 1 & \text{if } |A_{R,ij} - A_{T,ij}| > \delta \\ 0 & \text{otherwise;} \end{cases},$$

with δ being some prescribed small value for error tolerance, which was chosen in accordance with the noise level. We emphasize the importance of considering the difference $A_R - A_T$ directly when calibrating a reverse engineering scheme. Demonstrating that A_R, when substituted back into Eq. **1**, reproduces the experimental data closely (19–22) is insufficient, as this amounts to showing that the residual $\|A_R x - b - \dot{x}\|$ is small. But a small residual does not imply an accurate solution, especially when ill-conditioned equations are involved (34), which is the case here as we are attempting to invert nonsquare matrices by SVD. An A_R that leads to a small residual, although faithfully describing the experimental data, may not correctly predict the network's response to novel stimuli.

The results from one such experiment are depicted in Fig. 1. For small *M*, there are many errors in the reconstruction, but the number of errors *E* drops rapidly as *M* increases. A more detailed study where we kept track of the e_{ij}s row by row revealed (data not shown) that our algorithm recovers sparser rows earlier than less sparse rows as the number of measurements *M* increases. This result suggests the possibility of partially recovering gene networks even with a very small amount of data; we shall not pursue this idea here and shall concentrate on the recovery of entire networks.

We have also plotted in Fig. 2 the smallest number of measurements, M_c, that are needed to recover the entire $N \times N$ matrix correctly, i.e., with $E = 0$. It reveals that $M_c = O(\log N)$. This data requirement is maximally efficient, as the minimum number of measurements needed to recover the entire matrix is bounded by the information content in the matrix, which is conjectured (17) to be $M_c = \Omega(k \log(N/k))$.

For comparison, we attempted to reverse-engineer the network using SVD alone (19–22), without imposing sparseness by robust regression, i.e., by taking A_0 in Eq. **2** as the solution. We found that we could recover the connectivity matrix only when we had a large number of measurements ($M \gtrsim 0.99N$) (Fig. 3). For small *M*, we

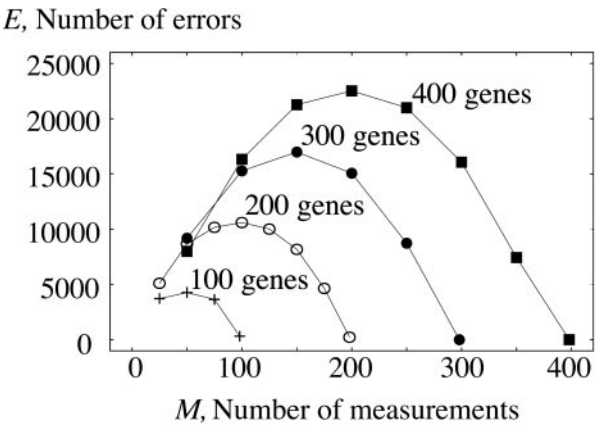

Fig. 3. Number of errors, *E*, made by SVD alone (without the imposition of sparseness) as a function of *M*, the number of measurements, for four linear networks of the form **1** with different sizes *N*.

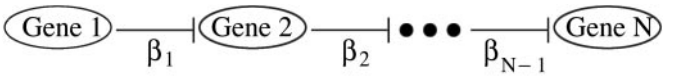

Fig. 4. Schematic of a one-dimensional gene network with a cascade structure.

found that A_R was far from A_T, although the A_R thus found could reproduce the time series data.

Example 2: A repressing cascade. In this second *in numero* experiment, we considered a one-dimensional cascade of genes, each of which is induced by an external stimulus and repressed by an immediate neighbor, as illustrated in Fig. 4. This system can be modeled by the following nonlinear system of ordinary differential equations (35, 36):

$$\frac{d}{dt}u_i = -\lambda_i u_i + \frac{\alpha_i}{1 + u_{i-1}{}^{\beta_i}} + \xi_i(t), \text{ for } i = 1, 2, \ldots, N, \quad \textbf{[5]}$$

where u_i is the concentration of the ith mRNA, λ_i is the degradation rate of the ith mRNA, α_i is the synthesis rate of the ith repressor, β_i is the repression cooperativity of the ith repressor, and $\xi_i(t)$ represents noise. Here we adopt the convention that $u_0 \equiv 0$.

In the absence of noise, Eq. **5** has a unique stable fixed point, given recursively by

$$u_i^* = \frac{1}{\lambda_i}\left(\frac{\alpha_i}{1 + u_{i-1}^{*\,\beta_i}}\right), \text{ for } i = 1, 2, \ldots, N. \quad \textbf{[6]}$$

Near this steady state, the dynamics is governed by the linearization **1** with $x_i = u_i - u_i^*$, $b_i = 0$, and

$$W_{ij} = \frac{-\alpha_i \beta_i u_{i-1}^{*\,\beta_i - 1}}{(1 + u_{i-1}^{*\,\beta_i})^2}\delta_{i-1,j}.$$

As in Example 1, to test the reverse engineering scheme, we repeatedly perturbed the system **5** from the steady state **6** with transient small random perturbations and took measurements while the system was relaxing back to the steady state. We iterated this process M times to collect M data points and applied our reverse engineering scheme to attempt to reconstruct the connectivity matrix. For small M, there are many errors in the reconstruction, but the number of errors E, as defined in Eq. **4**, drops rapidly as M increases (Fig. 5). In particular, with the parameter values chosen, we could recover the entire connectivity matrix correctly with $M \geq M_c = 70 \ll N = 400$.

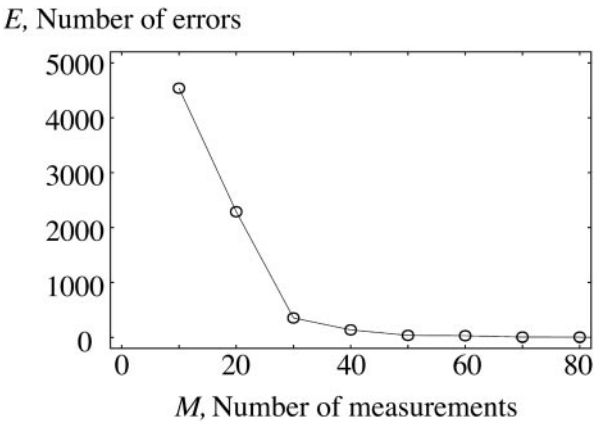

Fig. 5. Number of errors, E, made by the reverse engineering scheme as a function of M, the number of measurements, for the repressing cascade **5** with $N = 400$ genes.

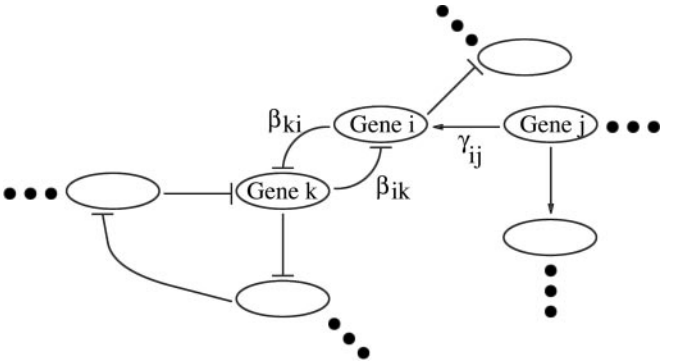

Fig. 6. Schematic of a nonlinear gene network with a random structure.

We emphasize the importance of keeping the system close to a steady state for the reverse engineering scheme presented here to be applicable. We have tried applying large perturbations to the system **5** so that the dynamics is far from equilibrium, i.e., u is no longer close to u^*, as may occur during development, disease, injury, or certain genetic or biochemical perturbations. In these cases, we could not recover the connectivity matrix even with a large ($>N$) number of measurements. This failure is not surprising, given that under such conditions the linearization **1** is no longer valid. In such cases, we need a nonlinear model to capture the dynamics and to recover the connectivity topology.

Example 3: A large sparse gene network. In this third *in numero* experiment, we considered a random network of genes. Each gene is induced by an external stimulus while also activated and repressed by other genes that are randomly chosen from a power-law distribution with a cutoff $k_{\max} \ll N$, as in Example 1. This system is illustrated in Fig. 6. Assuming that different regulatory effects do not interfere with one another, we can model the system by the following nonlinear system of ordinary differential equations (35, 36):

$$\frac{d}{dt}u_i = -\lambda_i u_i + \frac{\alpha_i + \sum_{j \in \mathcal{A}_i} u_j^{\gamma_{ij}}}{1 + \sum_{j \in \mathcal{A}_i} u_j^{\gamma_{ij}} + \sum_{k \in \mathcal{R}_i} u_k^{\beta_{ik}}} + \xi_i(t),$$

$$\text{for } i = 1, 2, \ldots, N, \quad \textbf{[7]}$$

where u_i is the concentration of the ith mRNA, λ_i is the degradation rate of the ith mRNA, α_i is the synthesis rate of the ith mRNA, γ_{ij} is the activation cooperativity of the jth gene on the ith gene, β_{ik} is the repression cooperativity of the kth gene on the ith gene, and $\xi_i(t)$ represents noise. The sets $\mathcal{A}_i$ and $\mathcal{R}_i$, whose cardinalities obey a power-law distribution, specify the genes that activate or repress, respectively, the ith gene.

Unlike Example 2, here we were unable to find the steady states analytically; we could not even ascertain analytically whether a steady state exists. We therefore resorted to numerics. By following the dynamics directly with numerical integration, we found that a stable fixed point exists for certain choices of parameters. Near a steady state u^*, the dynamics is governed by the linearization **1** with $x_i = u_i - u_i^*$, $b_i = 0$, and

$$W_{ij} = \frac{\gamma_{ij} u_j^{*\gamma_{ij}-1}}{1 + \sum_{l \in \mathcal{A}_i} u_l^{*\gamma_{il}} + \sum_{k \in \mathcal{R}_i} u_k^{*\beta_{ik}}} - \frac{\left(\alpha_i + \sum_{l \in \mathcal{A}_i} u_l^{*\gamma_{il}}\right)(\gamma_{ij} u_j^{*\gamma_{ij}-1} + \beta_{ij} u_j^{*\beta_{ij}-1})}{\left(1 + \sum_{l \in \mathcal{A}_i} u_l^{*\gamma_{il}} + \sum_{k \in \mathcal{R}_i} u_k^{*\beta_{ik}}\right)^2}.$$

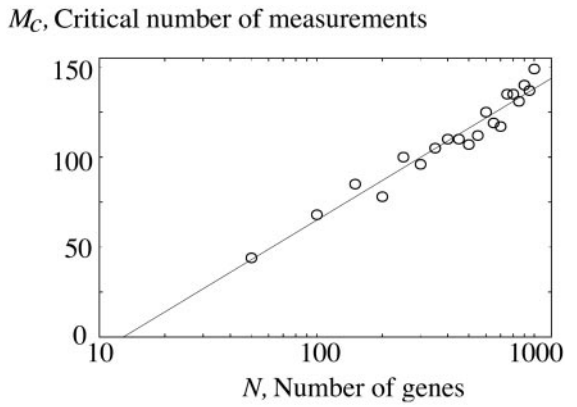

Fig. 7. Critical number of measurements, M_c, required to recover the entire connectivity matrix correctly, versus N, the size of the network for nonlinear systems of the form **7**. Circles: numerical data. Line: least-squares fit of the form $M_c = a + b \log N$.

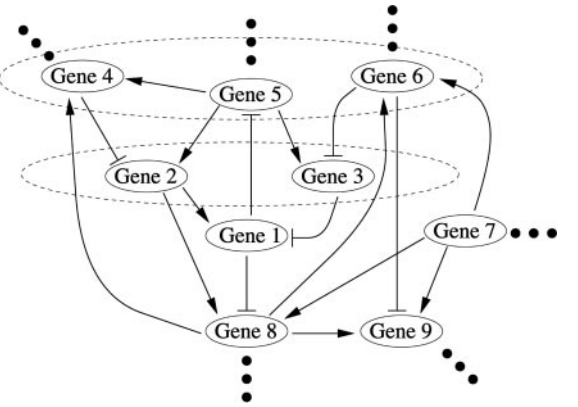

Fig. 8. Layer-by-layer recovery of network topology. Focusing on Gene 1, we can identify Genes 2 and 3 as the immediate upstream elements directly regulating Gene 1, and then Genes 4, 5, and 6 as next-immediate upstream elements indirectly regulating Gene 1.

As before, to test the reverse engineering scheme, we repeatedly perturbed the system **7** from a steady state with transient small random perturbations and took measurements while the system was relaxing back to the steady state. Repeating this process M times, we obtained M data points and applied our reverse engineering scheme to attempt to reconstruct the connectivity matrix. For small M, there are many errors in the reconstruction, but the number of errors E, as defined in Eq. **4**, drops rapidly to zero as M increases (data not shown), indicating that we can elucidate the whole network correctly with a small number of measurements. As Fig. 7 shows, the smallest number of measurements, M_c, that are needed to recover the entire $N \times N$ matrix again scales logarithmically with N, i.e., $M_c = O(\log N)$.

Discussion

We have proposed a reverse engineering scheme suitable for global reconstruction of gene networks, which are large and sparsely connected on a genome-wide scale. Our algorithm requires $O(\log N)$ sample points and $O(N^4)$ computations. We have tested and validated our scheme in three *in numero* experiments. In this section, we compare our method with other reverse engineering schemes. We also discuss how our scheme can be improved and generalized.

One method to reverse-engineer gene networks is to use genetic algorithms (16), which allow network models to evolve under selective pressure in an attempt to fit the data. A related idea is to train neural networks (17) to learn the network topology. However, these approaches are of restricted scope, as they typically require an unrealistically large amount of data and computation to generate connectivity maps for large networks. Similarly, Bayesian models (18), although useful for evaluating the likelihood of a particular hypothesis and for identifying the most likely network from a small set of competing candidate models, are inefficient if used to reconstruct network architectures *de novo* for networks of genomic scales. In comparison, adopting a linear model and then using SVD to reverse-engineer the network architecture (19–22) is more efficient in terms of both data requirement and computation. The SVD method attempts to reconstruct the networks directly from experimental data without prior knowledge of their structures and is therefore useful in generating a first draft of the network topology in novel situations. Nonetheless, one has to use SVD with care to get biologically meaningful results. What SVD does is to provide a family of candidate networks that are consistent with the microarray data. It does not identify which one of those candidates is the correct solution. The solution with the smallest L_2 norm may not correspond to the real structure, as discussed in Example 1. We therefore have to amend the SVD method with some additional criteria to select the most likely solution. We have proposed using the sparseness of the networks as one such criterion. Our numerical experiments have shown that this amended method can recover the network topology correctly, using $O(\log N)$ sample points and $O(N^4)$ computations.

One attractive feature of our reverse engineering scheme is that it can be easily parallelized, as it recovers the connectivity matrix row by row, with mutually independent operations. Moreover, if we focus on a particular gene, it takes $O(NM^2)$ computations to perform SVD and then $O(N^3)$ computations to impose sparseness to find the elements that are immediately regulating this particular gene. Successive iterations then identify upstream genes layer by layer (Fig. 8) without solving for the entire network, which requires $O(N^4)$ computations. Such a recovery process allows rapid elucidation of pathways that lead to the particular gene under study, identifying its regulating genes as potential drug targets for pharmaceutical purposes.

Nevertheless, our reverse engineering scheme has a counterintuitive feature. Although efficient in recovering the architectures of large networks, it is less efficient for small networks, requiring almost as many measurements as the number of genes to reconstruct the network topology, because the notion of sparseness is relative and is ill-defined for small networks. In a network with only a few genes, even if each gene is interacting with a small number of genes, it is still interacting with a significant portion of the network. Thus, the network may no longer be regarded as sparse. We have been unable to quantify this notion of sparseness and to pinpoint the critical size of a network as a function of the average number of connections. This issue is related to the concept of exact-fit point (30) in L_1 regression and has been solved only under very specific conditions governing the sample points. As a rule of thumb, our numerical experiments suggest that if the average number of connections is fewer than 10, which is a reasonable estimate for biological systems (24, 25), then our algorithm starts to show its efficiency when the network size is larger than 200 genes or so. As a consequence, our method is useful for rapidly furnishing on a global scale a first draft of the topology of the entire network that encompasses all of the gene regulatory networks in naturally occurring genomes but is not suitable for fine-tuning to improve the local resolution of small subnetworks that govern individual biological functions. To extract these local features to build biologically meaningful models for the various genetic and biochemical pathways, we need to integrate our method with other statistical methods and experimental data (27). Such methods may include using Bayesian networks to verify the likelihood of the paths (18)

or probing the networks iteratively to gain more information (J.T., M.K.S.Y., J. Hasty, and J.J.C., unpublished work).

As for the data, we may need to incorporate the activities of proteins (14, 37, 38) in addition to the mRNA expression levels. Our scheme is applicable to protein networks as well, because they are also sparsely connected large networks (38). Advances in experimental techniques have provided proteomic data on a massive scale and have sparked attempts to reverse-engineer protein networks (37–39). Most of these reverse engineering methods are based on clustering (40) and suffer from the same drawbacks as the clustering schemes for reverse engineering gene networks, as noted earlier.

One potential drawback of our scheme is that it requires data on time derivatives ($\dot{X}$ in Eq. **2**), which can be difficult to obtain especially in the presence of noise. However, with careful instrumentation (41) and statistics (42), it is possible to estimate the gene expressions relatively accurately by repeating microarray measurements only a small number of times. The large number of gene expressions per microarray provides the large number of samples to avoid small-sample biases for reliable estimation of the noise parameters. We conjecture that a similar approach can be used to obtain reasonable estimates of the time derivatives of the gene expressions. In our *in numero* experiments, we adopted an unsophisticated approach, where we measured only the gene expressions X, and then estimated the time derivatives $\dot{X}$ by linear interpolation. Even with such naïveté, we could correctly reverse-engineer the network architectures, as demonstrated in the *in numero* experiments.

There is considerable theoretical work that can be done to improve our method, both algorithmically and in terms of modeling. On the algorithmic side, there is much to investigate regarding how to impose sparseness. For convenience we have adopted the simplex method from refs. 32 and 33 to solve the L_1 regression problem. This computational algorithm may not be the most efficient one given our unconventional situation, with the number of regression parameters being approximately $N - M$, which increases linearly with the number of data points N. Other methods, such as interior point methods (43) may be worth considering. Indeed, minimizing the L_1 norm may not be the optimal way to unmask the outliers. Many different methods, such as least median of squares and least trimmed squares, have been proposed as alternatives (26). It is as yet unclear which of these methods is best suited for the task at hand. This promises to be a rich problem in robust statistics. Another idea that we may borrow from the study of robust statistics is to use designed experiments rather than random sampling to achieve higher exact-fit points (30), which translate into smaller numbers of measurements to recover the connectivity matrix. This construction can be experimentally realized by using genetic toggle switches (44) to perturb the gene networks systematically. A similar idea has been exploited (in J.T., M.K.S.Y., J. Hasty, and J.J.C., unpublished work) to reverse-engineer gene networks.

On the modeling side, for simplicity we have adopted Eq. **1**, a deterministic linear system of ordinary differential equation with constant coefficients, to model gene networks. We have neglected the effects of nonlinearities, noise, time delays (45), and combinatorial effects (46). Another complication is that network connectivity can change dynamically in response to changes in the experimental conditions, as proteins and metabolites are synthesized or destroyed to create or block pathways. It is futile to try to capture such a dynamically changing network by using a static model. Instead, we have to adopt a dynamical model with time-dependent matrix elements $W_{ij}(t)$ in Eq. **1**. Similarly, we can combine time series data from different experiments only if the data are obtained under comparable conditions, and in particular, with the system operating near the same steady state. Otherwise, the system may be governed by $\dot{x} = Cx$ in one case and by $\dot{x} = Dx$ in another, with $C \neq D$, so that we cannot capture all the data by using one model $\dot{x} = Ax$.

We expect that our reverse engineering scheme will be useful for reconstructing gene networks on a genome-wide scale when more experimental data become available. Because the number of sample points that are needed to recover the network scales logarithmically with the size of the network, we expect to be able to recover the network topology of real genomes, which consist of $O(10^4)$ genes, with several hundred measurements instead of tens of thousands of measurements. This amount of data should be obtainable in the near future as the cost of experimental data collection drops (1, 47).

We thank the referees for carefully reading an earlier draft of this manuscript and for their comments, which have significantly improved the quality of this manuscript. This work was supported by the National Science Foundation through the Bio-QuBIC Program (Grant no. EIA-0130331) and by the Defense Advanced Research Projects Agency (Grant no. F30602-01-2-0579).

1. Lockhart, D. J. & Winzeler, E. A. (2000) *Nature (London)* **405,** 827–836.
2. DeRisi, J. L., Iyer, V. R. & Brown, P. O. (1997) *Science* **278,** 680–686.
3. Wen, X., Fuhrman, S., Michaels, G. S., Carr, D. B., Smith, S., Barker, J. L. & Somogyi, R. (1998) *Proc. Natl. Acad. Sci. USA* **95,** 334–339.
4. Spellman, P. T., Sherlock, G., Zhang, M. Q., Iyer, V. R., Anders, K., Eisen, M. B., Brown, P. O., Botstein, D. & Futcher, B. (1998) *Mol. Biol. Cell.* **9,** 3273–3297.
5. Eisen, M. B., Spellman, P. T., Brown, P. O. & Botstein, D. (1998) *Proc. Natl. Acad. Sci. USA* **95,** 14863–14868.
6. Tavazoie, S., Hughes, J. D., Campbell, M. J., Cho, R. J. & Church, G. M. (1999) *Nat. Genet.* **22,** 281–285.
7. Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E. S. & Golub, T. R. (1999) *Proc. Natl. Acad. Sci. USA* **96,** 2907–2912.
8. Brown, M. P., Grundy, W. N., Lin, D., Cristianini, N., Sugnet, C. W., Furey, T. S., Ares, M., Jr., & Haussler, D. (2000) *Proc. Natl. Acad. Sci. USA* **97,** 262–267.
9. Morgan, B. J. T. & Ray, A. P. G. (1995) *Appl. Statist.* **44,** 117–134.
10. Marcotte, E. M. (2001) *Nat. Biotechnol.* **19,** 626–627.
11. Jeong, H., Tombor, B., Albert, R., Oltvai, Z. N. & Barabási, A.-L. (2000) *Nature (London)* **407,** 651–654.
12. Jeong, H., Mason, S. P., Barabási, A.-L. & Oltvai, Z. N. (2001) *Nature (London)* **411,** 41–42.
13. Wagner, A. (2000) *Nat. Genet.* **24,** 355–361.
14. Gygi, S. P., Rochon, Y., Franza, B. R. & Aebersold, R. (1999) *Mol. Cell. Biol.* **19,** 1720–1730.
15. Ljung, L. (1999) *System Identification: Theory for the User* (Prentice–Hall, Englewood Cliffs, N.J.), 2nd Ed.
16. Wahde, M. & Hertz, J. (2000) *BioSystems* **55,** 129–136.
17. D'haeseleer, P., Liang, S. & Somogyi, R. (2000) *Bioinformatics* **16,** 707–726.
18. Hartemink, A. J., Gifford, D. K., Jaakkola, T. S. & Young, R. A. (2001) in *Pac. Symp. Biocomputing*, Vol. 6, 422–433.
19. D'haeseleer, P., Wen, X., Fuhrman, S. & Somogyi, R. (1999) in *Pac. Symp. Biocomputing*, Vol. **4,** 41–52.
20. Alter, O., Brown, P. O. & Botstein, D. (2000) *Proc. Natl. Acad. Sci. USA* **97,** 10101–10106.
21. Raychaudhuri, S., Stuart, J. M. & Altman, R. B. (2000) in *Pac. Symp. Biocomputing*, Vol. 5, 452–463.
22. Holter, N. S., Maritan, A., Cieplak, M., Fedoroff, N. V. & Banavar, J. R. (2001) *Proc. Natl. Acad. Sci. USA* **98,** 1693–1698.
23. Press, W. H., Teukolsky, S. A., Vetterling, W. T. & Flannery, B. P. (1992) *Numerical Recipes in Fortran: The Art of Scientific Computing* (Cambridge Univ. Press, New York), 2nd Ed.
24. Arnone, M. I. & Davidson, E. H. (1997) *Development* (Cambridge, U.K.) **124,** 1851–1864.
25. Thieffry, D., Huerta, A. M., Pérez-Rueda, E. & Collado-Vides, J. (1998) *BioEssays* **20,** 433–440.
26. Hampel, F. R., Ranchetti, E. M., Rousseeuw, P. J. & Stahel, W. A. (1986) *Robust Statistics: The Approach Based on Influence Functions* (Wiley, New York).
27. Ideker, T., Galitski, T. & Hood, L. (2001) *Annu. Rev. Genom. Hum. Genet.* **2,** 343–372.
28. Weaver, D. C., Workman, C. T. & Stormo, G. D. (1999) in *Pac. Symp. Biocomputing*, 112–123.
29. van Someren, E. P., Wessels, L. F. A. & Reinders, M. J. T. (2000) in *Proc. of the 21st Symp. on Information Theory in the Benelux*, pp. 215–222.
30. Ellis, S. P. & Morgenthaler, S. (1992) *J. Am. Stat. Assoc.* **87,** 143–148.
31. Bloomfield, P. & Steiger, W. L. (1983) *Least Absolute Deviations: Theory, Applications, and Algorithms* (Birkhäuser, Boston).
32. Barrodale, I. & Roberts, F. D. K. (1973) *SIAM J. Numer. Anal.* **10,** 839–848.
33. Barrodale, I. & Roberts, F. D. K. (1974) *Comm. ACM* **17,** 319–320.
34. Noble, B. (1969) *Applied Linear Algebra* (Prentice–Hall, Englewood Cliffs, NJ).
35. Yagil, G. & Yagil, E. (1971) *Biophys. J.* **11,** 11–27.
36. Edelstein-Keshet, L. (1988) *Mathematical Models in Biology* (McGraw–Hill, New York).
37. Marcotte, E. M., Pellegrini, M., Thompson, M. J., Yeates, T. O. & Eisenberg, D. (1999) *Nature (London)* **402,** 83–86.
38. Uetz, P., Giot, L., Cagney, G., Mansfield, T. A., Judson, R. S., Knight, J. R., Lockshon, D., Narayan, V., Srinivasan, M., Pochart, P., *et al.* (2000) *Nature (London)* **403,** 623–627.
39. Zhu, H., Bilgin, M., Bangham, R., Hall, D., Casamayor, A., Bertone, P., Lan, N., Jansen, R., Bidlingmaier, S., Houfek, T., *et al.* (2001) *Science* **293,** 2101–2105.
40. Eisenberg, D., Marcotte, E. M., Xenarios, I. & Yeates, T. O. (2000) *Nature (London)* **405,** 823–826.
41. Eisen, M. B. & Brown, P. O. (1999) *Methods Enzymol.* **303,** 179–205.
42. Ideker, T., Thorsson, V., Siegel, A. F. & Hood, L. E. (2000) *J. Comput. Biol.* **7,** 805–817.
43. Wright, S. J. (1996) *Primal-Dual Interior-Point Methods* (SIAM, Philadelphia).
44. Gardner, T. S., Cantor, C. R. & Collins, J. J. (2000) *Nature (London)* **403,** 339–342.
45. McAdams, H. H. & Arkin, A. (1997) *Proc. Natl. Acad. Sci. USA* **94,** 814–819.
46. Pilpel, Y., Sudarsanam, P. & Church, G. M. (2001) *Nat. Genet.* **29,** 153–159.
47. Kalir, S., McClure, J., Pabbaraju, K., Southward, C., Ronen, M., Leibler, S., Surette, M. G. & Alon, U. (2001) *Science* **292,** 2080–2083.