Reverse Engineering of Gene Networks with LASSO and Nonlinear Basis Functions

Mika Gustafsson,[a] Michael Hörnquist,[a] Jesper Lundström,[b] Johan Björkegren,[b] and Jesper Tegnér[b,c]

[a]*Department of Science and Technology, Linköping University, Norrköping, Sweden*

[b]*Department of Medicine, Center for Molecular Medicine, Karolinska Universitetssjukhuset, Stockholm, Sweden*

[c]*Department of Physics, Linköping University, Linköping, Sweden*

The quest to determine cause from effect is often referred to as reverse engineering in the context of cellular networks. Here we propose and evaluate an algorithm for reverse engineering a gene regulatory network from time-series and steady-state data. Our algorithmic pipeline, which is rather standard in its parts but not in its integrative composition, combines ordinary differential equations, parameter estimations by least angle regression, and cross-validation procedures for determining the in-degrees and selection of nonlinear transfer functions. The result of the algorithm is a complete directed network, in which each edge has been assigned a score from a bootstrap procedure. To evaluate the performance, we submitted the outcome of the algorithm to the reverse engineering assessment competition DREAM2, where we used the data corresponding to the InSilico1 and InSilico2 networks as input. Our algorithm outperformed all other algorithms when inferring one of the directed gene-to-gene networks.

Key words: **reverse engineering; network inference; nonlinear; DREAM conference; LARS; LASSO**

Introduction

To identify cellular mechanisms of importance for a biological process or a disease is an essential endeavor for science. To accomplish this enterprise, it is central to uncover the structure and dynamics of the web of interactions between genes, proteins, and metabolites. Recent progress in high-throughput technologies has emphasized the need for developing efficient and accurate algorithms for reconstructing such networks from experimental data.[1] Here we formulate this problem using a system of ordinary differential equations (ODEs) to reverse engineer a regulatory gene-to-gene network from gene expression data sampled during steady state and time series.

Several frameworks for modeling gene regulatory networks have been proposed. Depending on size of the network and available data, these models range from Boolean networks,[2] today with thousands of nodes,[3] to detailed descriptions of the biochemical reactions with only a few units.[4] In between these two extremes, there are both graphical models (including Bayesian networks) and information-theoretic models.[5] A special class of gene regulatory network models, which has gained some popularity, comprises the linear, time-continuous models based on systems of ODEs.[5] The first study of this kind, to the best of our knowledge, was the paper from 1999 by D'haeseleer and coworkers.[6] This framework has since been revisited on numerous occasions with many variations on the ODE theme (see Ref. 7 and references therein). Here we follow this tradition with two modifications. First we use

Address for correspondence: Michael Hörnquist, Linköping University, Department of Science and Technology (ITN), 601 74 Norrköping, Sweden. Voice: +46 11363381. micho@itn.liu.se

The Challenges of Systems Biology: Ann. N.Y. Acad. Sci. 1158: 265–275 (2009).
doi: 10.1111/j.1749-6632.2008.03764.x

nonlinear transfer functions to preprocess the data before the actual network inference. Our rationale is to reduce prediction errors and thereby increase the performance of the model. Second, our algorithm can use both steady-state and time-series data.

A central problem in the field has been how to assess the quality and applicability of the plethora of reverse-engineering algorithms. Interestingly, no consensus has been obtained regarding which methods are most efficient, and not even on how their efficiency should be measured, although there are some comparison studies published.[8] It is no understatement that this state of affairs is confusing not only for the experimental biologist but also for the experienced engineer. Therefore, it was very welcome when the Dialogue for Reverse Engineering Assessments and Methods (DREAM) Initiative, resulting from a conference in 2006,[9] pushed the notion of a competition to let researchers in the field show how well their methods performed in an objective test.[10] Importantly, this exercise also addresses the fundamental problem of how to measure performance and the role of prediction error in this process.

The organization of this paper is the following: In section Data, we briefly review the conditions given by the DREAM Initiative for challenge 4 in the competition. In the following section, Method, we present our reverse-engineering algorithm, and in the Results section, the performance is presented and evaluated using the networks that generated the data and which became available after the competition. Finally, in Discussion and Outlook, we address what can be learned from a competition of this kind and emphasize some directions in which we think future research should develop.

Data

The data come from two different artificial networks, referred to as InSilico1 and InSilico2 in the competition launched by the DREAM Initiative.[9] For completeness of the paper, we quote from the web page:

> **Description:** These datasets were produced from a gene network with 50 genes, where the rate of synthesis of the mRNA of each gene is affected by the level of mRNA of other genes.
>
> **InSilico1-heterozygous.xls** contains steady state levels for the wild-type and 50 heterozygous knock-down strains for each gene (+/−). Values of gene expression are provided for a standard condition (steady states).
>
> **InSilico1-null-mutants.xls** contains steady state levels for the wild-type and 50 null mutant strains for each gene (−/−). Values of gene expression are provided for a standard condition (steady states).
>
> **InSilico1-trajectories.xls** contains time courses (trajectories) of the network recovering from several external perturbations. There are 23 different perturbations and 26 time points for each one.

Hence, for each network there are $51 + 51 + 23 \times 26 = 700$ different data points for each gene, whereas the full network consists of only 50 genes. The sheer number of data points makes the reverse-engineering problem more tractable, but also less realistic, as compared with the *in vitro* or *in vivo* situation. Moreover, from the observation that all expression values are non-negative we infer that the values cannot be interpreted as log ratios. Finally, the distribution of the raw data suggests it has not been contaminated by noise or outliers.

Method

The basic assumption we utilize is that both the time-course and the steady-state data can be described by a set of N ordinary differential equations. Since all data points can be considered gene-by-gene, the network inference factorizes into N independent problems. Hence

each equation represents the dynamics of a single gene in the network. We therefore make the following ansatz:

$$\frac{dx_i}{dt}(t) = \sum_{j=1}^{\mathcal{N}} \sum_{a} w_{ija} f_a[x_j(t)] - \lambda_i x_i(t), \quad i = 1, \ldots, \mathcal{N}. \tag{1}$$

Here $x_i(t)$ denotes the expression level of gene i at time t and w_{ija} is the net effect of gene j on gene i mediated by the function f_a. These functions f_a are generally nonlinear functions of "typical" behavior, called transfer functions, to be determined later. It might be beneficial to use more than one transfer function for each gene, and therefore we index them by the letter a. The last term corresponds to degradation, which means that λ_i has to be non-negative. The left-hand side is the effective dynamics of the mRNA concentration for gene i, that is, the transcription rate minus the degradation rate.

The work flow of our reverse-engineering algorithm is presented in Figure 1. Our repeating procedure is as follows. Step 1, data are preprocessed; Step 2, pick a transfer function from a predefined list; Step 3, perform the network inference; and Step 4, compute the prediction error given the network and present transfer function. Then repeat steps 2–4 until the list of transfer functions is exhausted. In detail:

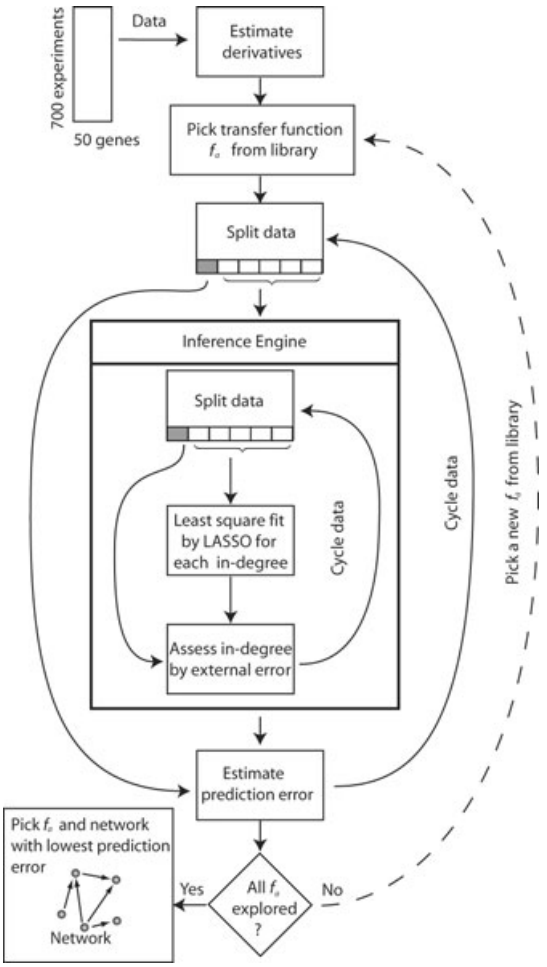

FIGURE 1. Structure of the reverse-engineering algorithm.

Step 1: First we estimate the time-derivatives using a spline approximation of the original data. Here interpolating cubic splines with no further constraints on the oscillations are utilized, since there seem to be no outliers in the trajectories.

Step 2: We pick one transfer function f_a from a predefined list. To set the stage for cross-validation, we exclude one sixth of the data for the assessment in Step 4 of the particular choice of function f_a.

Step 3: At the core of the reverse-engineering algorithm, the box in Figure 1 called the "Inference Engine," we perform the inference, described in detail below, using a six-fold cross-validation procedure for model selection, that is, for determining the in-degree, by minimizing the external error. After determination of the network structure and the actual values of the coefficients, we exit the inference engine.

Step 4: We assess the chosen transfer functions by estimating the prediction error from the hidden data. This estimation is performed as a cross-validation (six-fold) on the excluded data from Step 2, that is, we repeat Steps 2–4 six times for an estimation of the prediction error.

Then we have another repetition of Steps 2–4 for the next transfer function, as illustrated in Figure 1 (dashed curve to

the right), until the list of functions is exhausted. Eventually, we choose the function that results in the least prediction error, and get the corresponding network as a final result.

We form the list from which the transfer functions are selected from a general (prior) knowledge of mathematical models within biology with growths, saturations, and so on. Explicitly, we first pick f_1 from the set:

$$\left\{ C,\, x,\, x^2,\, \frac{1}{1+x},\, \frac{1}{1+x^2},\, e^{-bx},\, \log(1+x),\, \frac{1}{1+e^{-cx}},\, \frac{1}{1+e^{-x^2}},\, \right\} \quad (2)$$

where C is a constant, $b = -1, 0.5, 1, 2, 3$ and $c = 1, 1.5, 2, 3, \ldots, 25$. We find $f_1(x) = e^{-x}$, and with this function chosen, we proceed in a greedy fashion to find f_2 from the same set (2). Eventually, we end up with the form

$$f_2(x) = \frac{1}{1+e^{-22x}}. \quad (3)$$

Note we use the same functions for all genes. Also, the function f_2 is quite close to a Heaviside function, which reflects the idea that sometimes a gene can simply be considered to be either "on" or "off" and due to its boundedness puts an upper limit on how much one gene can influence another.

The actual data fit for finding coefficients, the box called the "Inference Engine" of Figure 1, is performed as a least square problem with a certain constraint explained below. The choice of least squares is motivated by computational convenience and our observation that there seem to be no noise or outliers in the dataset. As previously noted, the reverse engineering problem factorizes and we can therefore consider the regulation of each gene independently. By indexing the times for the measurements as t_k, $k = 1, \ldots, K$, where K is the total number of measurements for all series and steady-state data, we can write the objective function as:

$$\sum_k \left[\frac{dx_i}{dt}(t_k) - \left(-\lambda_i x_i(t_k) + \sum_{j \neq i} \sum_a w_{ija} f_a[x_j(t_k)] \right) \right]^2 \quad (4)$$

That is, we search for the values of λ_i and w_{ija}, $j = 1, \ldots, N; j \neq i$ which minimize (4) for all $i = 1, \ldots, N$. However, note that when the time-derivative $\frac{dx_i}{dt}$ is zero, for example, for all steady-state data, the minimum for the corresponding term is simply obtained for all parameters λ_i and w_{ija} equal zero. In order to avoid this problem, we rewrite the objective function as:

$$\sum_k \left[x_i(t_k) - \left(-\frac{1}{\lambda_i} \frac{dx_i}{dt}(t_k) + \sum_{j \neq i} \sum_a \frac{w_{ija}}{\lambda_i} f_a[x_j(t_k)] \right) \right]^2 \quad (5)$$

The number of experiments exceeds the number of genes, which makes the minimization problem well-posed even without further constraints. However, all coefficients will be non-zero with probability one unless we perform some kind of model selection. Our choice is equivalent to using the LASSO—the least absolute selection and shrinkage operator,[13] which here means the constraint

$$\left| \frac{1}{\lambda_i} \right| + \sum_{j,a} \left| \frac{w_{ija}}{\lambda_i} \right| \leq \mu_i. \quad (6)$$

The constraint μ_i is increased from zero, where all coefficients are zero, up to a value when all of the coefficients with probability one are non-zero (in general, the number of non-zero coefficients can never exceed the number of experiments, but here we have enough experiment for every coefficient to be well determined even without any constraints). Each case of a certain number of non-zero coefficients is evaluated by estimating the prediction error by a six-fold

cross-validation. The actual values of the coefficients are found using the forward-selection method LARS, Least Angle Regression,[11] in a form implemented by Vanden Bergen.[12] LARS is an efficient implementation of LASSO,[13] which has been explored earlier for reverse engineering of gene networks.[7] Note that although LARS is a forward-selection method, it still has the ability to discover multivariable dependencies, while excluding correlated columns for stability reasons. This is of particular importance here when we are mainly interested in the presence of individual edges in the network. We pick the network that results in the least prediction error.

For some occasions, the factor λ_i^{-1} is inferred to have the value zero. That value is not valid for the model, because it corresponds to an instantaneous degradation and it also renders it impossible to determine the nominators of the coefficients $\frac{w_{ija}}{\lambda_i}$. For these rare occasions, we employ the formulation in (4).

As a final step in our reverse-engineering procedure, we apply a bootstrap procedure for obtaining a score for each edge. This is performed for the chosen transfer functions, and we only need to concentrate on the Inference Engine from Figure 1. In detail, we repeat the following procedure 10,000 times: We sample 700 experiments, with replacement, from the total set of data, and apply the Inference Engine. Each time an edge in the network is picked, that is, each time the corresponding element w_{ija} is inferred to be non-zero, we increase the score of the edge by one unit. Eventually, these scores form the basis for our submitted networks to the DREAM competition.

Results

After the predictions upload deadline, it was disclosed that the data originated from simulations run on the COPASI (COmplex PAthway SImulator) platform.[14] The underlying dynamical equations were of multiplicative type, where an activating factor was of the form $x_i/(x_i + C_1)$ and a repressing factor followed $C_2/(x_i + C_3)$, with C_1, C_2, and C_3 as constants. The equations also included a degradation term of the same form as we assumed. The topology of the first network turned out to be of the Erdös-Renyi (ER) type[15] with a Poissonian degree distribution, illustrated in Figure 2, while the second network was of Barabási-Albert (BA) type[15] displaying a power-law degree distribution, illustrated in Figure 3. Here we discuss our results from knowledge of these networks. This is clearly different when inferring networks from biological experiments when we do not have any true answer to evaluate our prediction.

The procedure described above resulted in two lists of predicted edges, one for each network. The lists are sorted according to our confidence in the prediction of the presence of the edge (obtained from the score of the bootstrap, described above). The scoring metrics for the DREAM competition were:[9]

> **Scoring Metrics:** We will score the results using the area under the precision versus recall curve for the whole set of predictions. For the first *k* predictions (ranked by score, and for predictions with the same score, taken in the order they were submitted in the prediction files), precision is defined as the fraction of correct predictions to *k*, and recall is the proportion of correct predictions out of all the possible true connections [. . .]. Other metrics such as precision at 1%, 10%, 50%, and 80% recall, and the area under the receiver operating characteristic (ROC) curve will also be evaluated.

The performances of our algorithm, according to the scoring functions, are shown in Figures 4 and 5 for each network, respectively. Worth noting is that our assumed functional form of the equations were quite different from the ones utilized for producing the data, but still the algorithm performs reasonably well. An effective biological regulatory gene-to-gene network is probably best described with a wider class of equations than these biochemically inspired forms utilized by COPASI. Thus, it is important that the reverse engineering works generally and not only for the same hypothesized

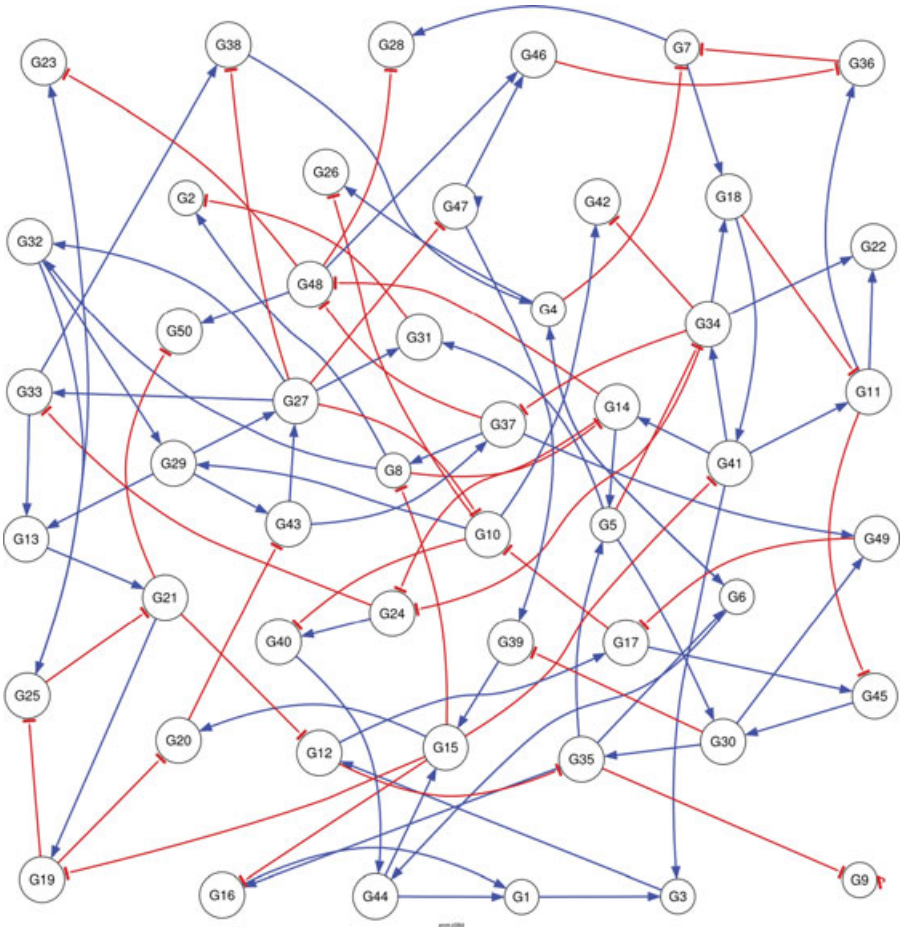

FIGURE 2. The first of the two networks from which data were provided. The degree distribution of both in-degrees and out-degrees follows a Poissonian curve, that is, the network is of the Erdös-Renyi (ER) type.

form as used for the data generation. Obviously, our algorithm had the best performance on the BA-network, which is promising since it is generally believed that biological networks are more similar to those than to ER-networks.[15] This difference in performance between the ER- and BA-networks is interesting and deserves further study. Right now, we mainly note that many units in the BA-network were regulated by only one other unit, that is, combinatorial regulations were quite rare, which might have been beneficial for our result. Indeed, in the DREAM competition, our algorithm was outstanding for the BA-network (the areas under the precision-recall and ROC curves were 0.26 and 0.75, respectively, compared with the second scoring algorithm, which obtained the values 0.15 and 0.66), but only scored second for the ER-network (areas under curves were 0.13 and 0.72, where the top performance gave 0.20 and 0.81).

A closer inspection reveals that we have also identified several coregulated genes as one regulating the other. For example, in the network InSilico1, we conjecture with score 1 (see Table 1) that the nodes 20 and 25 regulate each other. From Figure 2, lower left corner, we see that both these nodes are negatively regulated by node 19, but there is no direct edge between them. In other cases we have failed to identify an intermediate regulatory gene in a pathway. A typical example of what occurs is in InSilico2,

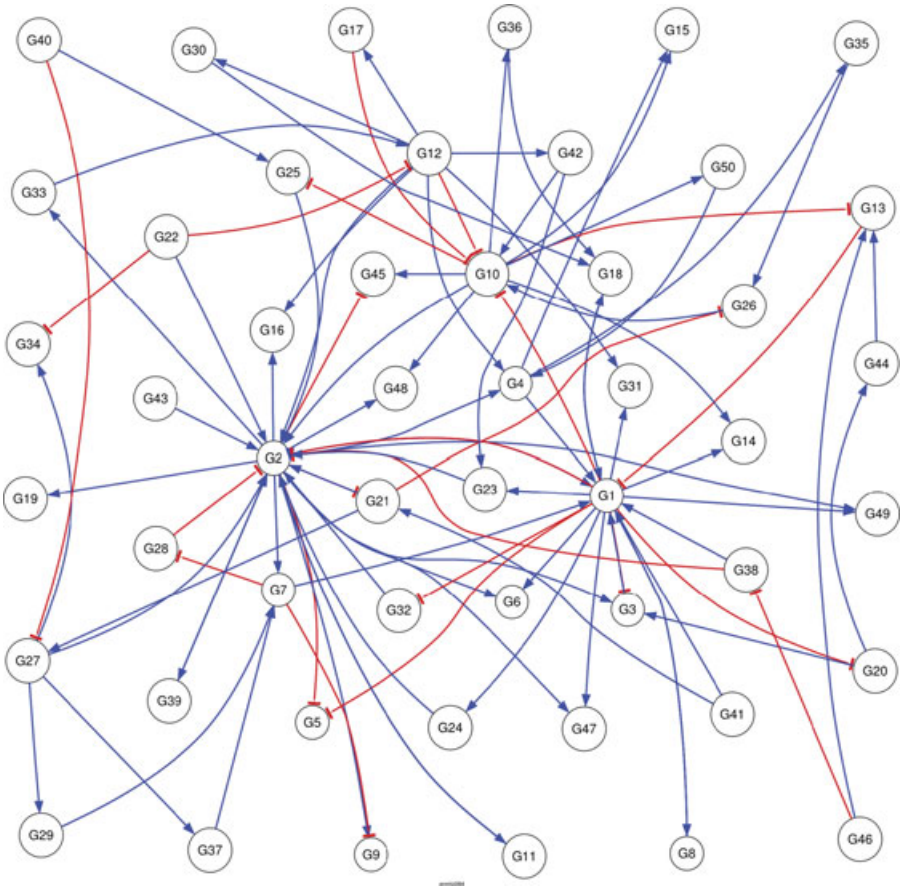

FIGURE 3. The second of the two networks from which data were provided. The degree distribution of both in-degrees and out-degrees follows a power-law, that is, the network is of the Barabási-Albert (BA) type.

when we with score 0.994 suggest that node 12 regulates node 48. This is true, as seen in Figure 3, upper middle part, but through a cascade with node 10 as an intermediate unit. In both cases, the edges are judged as simply "wrong," but still the result might be useful if the purpose of a large-scale inference is to find communities of genes involved in a certain process[16] or if we search for affected targets for some drug. The exact relationship between the genes is then a question for a more refined analysis. Therefore, we also show in Table 1 the *undirected* distance in the real net between the 24 most probable pairs of nodes from each network in our reverse engineering. To use the *directed* distance here would be misleading, since we are mainly looking for co-regulated genes, and between such the directed distance is irrelevant, and can even be infinite in case the network is not strongly connected. In the case of a perfect inference, all these distances should of course be unity, but we can see that even if we "miss," the mistake is not that big. The mean values for these 24 edges are 1.96 and 1.46 for the ER- and BA-networks, respectively. One should compare these numbers with the mean distances for the entire networks, which are 2.91 and 2.43, respectively.

Finally, we can also study the signs of our inferred edges in the networks and compare them with the true models. The correspondence turns out to be less satisfactory, which probably is a result of our ansatz, where terms could be negative indicating repression of the

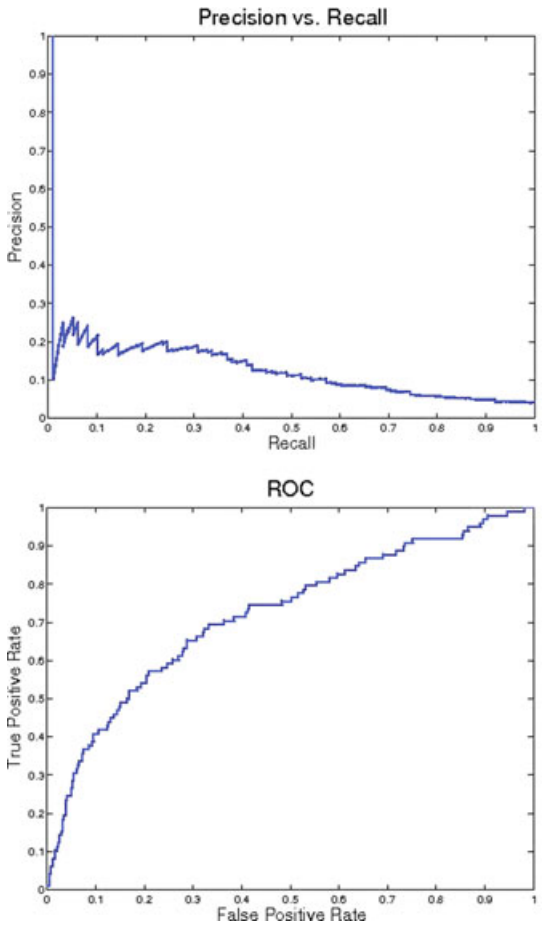

FIGURE 4. Precision versus recall and ROC curves for our reversed-engineered network for In-Silico1, the ER-network. The area under the former curve is 0.129 and under the latter 0.722.

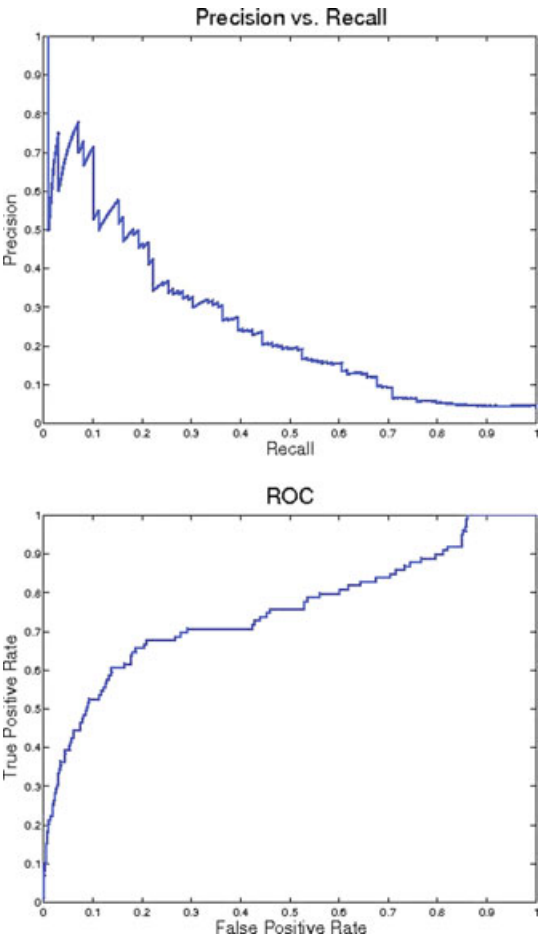

FIGURE 5. Precision versus recall and ROC curves for our reversed-engineered network for In-Silico2, the BA-network. The area under the former curve is 0.256 and under the latter 0.753.

gene, while for COPASI all factors except the self-degradation were positive, and the ones corresponding to repression just had a negative derivative.

Discussion and Outlook

Normally, one of the great challenges for reverse engineering is to find suitable ways to assess the quality of an algorithm. There is no generally accepted way of doing this, but this time, though, we have a "gold standard" against which we can measure the precision of the inference.

In contrast to the case of real data, here the number of experiments (time-points at which we have data) exceeds the number of genes, thereby rendering the problem well-posed even if all coefficients w_{ija} should turn out to be non-zero. Nevertheless, this is a useful exercise, since this kind of "best-case scenario" gives us a first quality judgement. If an algorithm does not even work properly here, it should be practically useless for real data since the inference problem then is less tractable.

Of course, a man-made network cannot contain all features of a biological network, which has been under evolutionary pressure for billions of years. This is partly because we

TABLE 1. Scores of the 24 top-ranked interactions from the reverse engineering, together with their undirected distance in the true networks and the nodes they connect

Network InSilico1				Network InSilico2			
Score	Dist	From	To	Score	Dist	From	To
1	1	1	3	1	1	10	36
1	2	20	25	1.000	1	45	10
1	2	25	20	0.999	1	12	42
0.999	1	3	1	0.999	1	27	37
0.998	3	46	15	0.999	2	50	36
0.996	1	36	11	0.999	1	10	45
0.995	3	37	40	0.998	1	20	44
0.991	3	4	15	0.998	1	10	50
0.991	3	18	46	0.997	1	27	29
0.990	5	23	26	0.994	2	12	48
0.989	1	8	32	0.986	1	12	17
0.988	1	43	27	0.982	2	36	50
0.987	2	19	43	0.971	1	2	47
0.986	3	15	4	0.966	1	10	25
0.982	1	32	27	0.964	3	34	48
0.982	1	34	41	0.959	2	45	36
0.978	1	11	36	0.942	2	47	3
0.977	3	40	37	0.940	2	5	21
0.977	1	38	4	0.917	2	50	25
0.974	2	43	19	0.912	1	1	47
0.974	3	46	18	0.909	2	10	41
0.973	1	19	21	0.908	2	50	45
0.970	2	19	8	0.903	1	2	21
0.970	1	41	34	0.899	1	1	31

probably have not uncovered all features of the biological systems, and partly for purely practical reasons—it is hard to pay attention to all features. For instance, for a long time it has been acknowledged that real biological networks are both modular[17] and contain motifs.[18] Nevertheless, although such structures can both be an obstacle for the reverse engineering as well as something one might take advantage of, the assessment of the algorithms has to start somewhere.

Another aspect, not considered here, but of utmost importance for any large-scale problem, is that of computational efficiency. Although the CPU-power available is always increasing, the need for analyzing data is always larger, and proper choices of algorithms and implementations remain important parts for any computational science, as discussed further in Ref. 19.

In retrospect, we notice that our nonlinear function f_1 probably was too linear for the range within which most of the data were provided. This resulted in the objective function (5) to pick coregulated genes as one (or both) being regulated by the other, especially when the derivative term was zero or close to zero. Thus, our desire to utilize the steady-state data as well gave us many false positives. There are some suggestions on how to tackle this problem,[21,22] but these were not utilized here. Again, we come to a conclusion well-known among theorists, that data in time-series are superior to steady-state data. Unfortunately, time-series data are hard to obtain for experimental biologists, and we need to continue the exploration of how the steady-state data can also be utilized.

Another issue, not addressed here, is the performance of our algorithm on data contaminated with noise and outliers. This was not part of the DREAM challenge this time, but we intend to get back to this very important question in the future.

In summary, the present algorithm has shown considerable promise, but still there are at least four directions within the ODE concept for reverse engineering of directed networks.

- First, we need a more systematic way to find a suitable basis set of functions. Now, we utilized a form of greedy approach, but when the size of the data set increases, not even this will be possible.
- Second, the huge amount of expression data available consist mainly of steady-state data, that is, not much is in the form of time-series. This has to be taken into account, if possible, when formulating an effective algorithm for reverse engineering. To simply discard all these data would be too much waste.
- Third, tightly connected to the second point, is the fact that often the number of genes/units in the network greatly

exceeds the number of measurements. In this perspective, the DREAM competition was an unrealistic dream for everyone working with reverse engineering. Even if we also consider steady-state data, we have to find suitable ways to tackle this situation since it is highly unlikely the situation will change in the near future. This includes the issue of integration of various types of data. Although this was not the issue for the present two networks for the DREAM competition, we believe this is very important in the future development of systems biology. Especially when dealing with large-scale genome-wide reverse engineering, this has to be considered to help overcome the lack of expression data.

- Last, but not least, we find it essential to reflect upon the impact of the objective function on the design of a reverse-engineering algorithm. Here we have relied on the notion of prediction accuracy. However, it has become increasingly clear within the machine-learning community that a reduction of the prediction error does not imply a control of the false discovery rate, FDR.[20] Different algorithms, each having a small prediction error, are as a rule expected to have different FDRs, and different sets of edges will therefore be selected. This is a problem that will be put in the forefront thanks to the DREAM initiative.

Finally, we would like to thank the organizers of DREAM for providing one good forum for assessment of algorithms. Despite all kinds of criticisms that can be directed against a competition of this sort, and despite all shortcomings one can see in the presented in-silico networks, we still hold this has been an excellent service to the whole community of systems biology. We are looking forward to the next round.

Acknowledgments

We acknowledge Pedro Mendes and Gustavo Stolovitzky for letting us use Figures 2 and 3 (PM) and 4 and 5 (GS), respectively. We also acknowledge financial support from the Centre for Industrial Information Technology at Linköping Institute of Technology, Sweden (MG and MH) and from the PhD school in medical bioinformatics (FMB) (JL).

Conflicts of Interest

The authors declare no conflicts of interest.

References

1. Tegnér, J. & J. Björkegren. 2007. Perturbations to uncover gene networks. *Trends Genet.* **23:** 34–41.
2. Kauffman, S.A. 1969. Metabolic stability and epigenesis in randomly constructed genetic nets. *Theor. Biol.* **22:** 437–467.
3. Karlsson, F. & M. Hörnquist. 2007. Order or chaos in Boolean gene networks depends on the mean fraction of canalizing functions. *Physica A* **384:** 747–757.
4. Werner, M., I. Ernberg, J. Zou, *et al*. 2007. Epstein-Barr virus latency switch in human B-cells : a physicochemical model. *BMC Syst. Biol.* **1:** 40.
5. Margolin, A.A. & A. Califano. 2007. Theory and limitations of genetic network inference from microarray data. *Ann. N. Y. Acad. Sci.* **1115:** 51–72.
6. D'haeseleer, P., X. Wen, S. Fuhrman & R. Somogyi. 1999. Linear modeling of mRNA expression levels during CNS development and injury. In *Pacific Symposium on Biocomputing*, Vol. 4. R.B. Altman, A.K. Dunker, L. Hunter, *et al*., Eds.: 41–52. World Scientific Publishing Co. Singapore.
7. Gustafsson, M., M. Hörnquist & A. Lombardi. 2005. Constructing and analyzing a large-scale gene-to-gene regulatory network-lasso-constrained inference and biological validation. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2:** 254–261.
8. Bansal, M., V. Belcastro, A. Ambesi-Impiombato & D. di Bernardo. 2007. How to infer gene networks from expression profiles. *Mol. Syst. Biol.* **3:** 78.
9. Stolovitzky, G., D. Monroe & A. Califano. 2007. Dialogue on reverse-engineering assessment and methods. The dream of high-throughput pathway inference. *Ann. N. Y. Acad. Sci.* **1115:** 1–22.
10. DREAM, Dialogue on Reverse-Engineering Assessment and Methods (2007), project webpage: http://wiki.c2b2.columbia.edu/dream/index.php/The_DREAM_Project.
11. Efron, B., T. Hastie, I. Johnstone & R. Tibshirani. 2004. Least angle regression. *Ann. Stat.* **32:** 407–499.

12. Vanden Berghen, F. 2005. LARS Library: Least Angle Regression Stagewise Library. The MATLAB implementation is utilized here. Unpublished.
13. Tibshirani, R. 1996. Regression Shrinkage and selection via the Lasso. *J. Roy. Stat. Soc., Ser. B* **58:** 267–288.
14. Hoops, S., S. Sahle, R. Gauges, *et al*. 2006. COPASI — a complex pathway simulator. *Bioinformatics* **22:** 3067–3074.
15. Barabási, A.-L. & Z. Oltvai. 2004. Network biology: Understanding the cell's functional organization. *Nat. Rev. Genet*. **5:** 101–113.
16. Gustafsson, M., M. Hörnquist & A. Lombardi. 2006. Comparison and validation of community structures in complex networks. *Physica A: Stat. Mech. Appl.* **367:** 559–576.
17. Hartwell, L.H., J.J. Hopfield, S. Leibler & A.W. Murray. 1999. From molecular to modular cell biology. *Nature* **402:** C47–C52.
18. Milo, R., S. Shen-Orr, S. Itzkovitz, *et al*. 2002. Network motifs: simple building blocks of complex networks. *Science* **298:** 824–827.
19. Gustafsson, M. & M. Hörnquist. 2008. Integrating expression data, protein-protein interaction data and TF-binding data for improved quality in reverse engineering of gene regulatory networks. In *Computational Methods in Gene Regulatory Networks*. S. Das, D. Caragea, S. Welch & W. Hsu, Eds. IGI Global, Hershey, PA.
20. Nilsson, R., J.M. Peña, J. Björkegren & J. Tegnér. 2007. Consistent feature selection for pattern recognition in polynomial time. *J. Mach. Learn. Res.* **8:** 589–612.
21. Rice, J.J., Y. Tu & G. Stolovitzky. 2005. Reconstructing biological networks using conditional correlation analysis. *Bioinformatics* **21:** 765–773.
22. Basso, K., A.A. Margolin, G. Stolovitzky, *et al*. 2005. Reverse engineering of regulatory networks in human B cells. *Nat. Genet*. **37:** 382–390.