DISCOVERY OF REGULATORY INTERACTIONS THROUGH PERTURBATION: INFERENCE AND EXPERIMENTAL DESIGN

TREY E. IDEKER[†], VESTEINN THORSSON[†]
Department of Molecular Biotechnology, University of Washington
Seattle, WA 98195-7730, USA
(trunk@u.washington.edu, thorsson@u.washington.edu)

RICHARD M. KARP
Mathematical Sciences Research Institute
International Computer Science Institute
University of California, Berkeley
Berkeley, CA 94720-1198, USA
(karp@icsi.berkeley.edu)

We present two methods to be used interactively to infer a genetic network from gene expression measurements. The *predictor* method determines the set of Boolean networks consistent with an observed set of steady-state gene expression profiles, each generated from a different perturbation to the genetic network. The *chooser* method uses an entropy-based approach to propose an additional perturbation experiment to discriminate among the set of hypothetical networks determined by the predictor. These methods may be used iteratively and interactively to successively refine the genetic network: at each iteration, the perturbation selected by the chooser is experimentally performed to generate a new gene expression profile, and the predictor is used to derive a refined set of hypothetical gene networks using the cumulative expression data. Performance of the predictor and chooser is evaluated on simulated networks with varying number of genes and number of interactions per gene.

1 Introduction

Recently a variety of experimental techniques have been developed with the ability to observe the expression of many genes simultaneously. At the forefront of these technologies lies the *DNA microarray*, commonly used to monitor gene expression at the level of mRNA abundance. Similarly, the rapid identification of proteins and protein abundances is becoming possible through methods such as 2D polyacrylamide gel electrophoresis (2D-PAGE) coupled with mass spectroscopy[1]. The principal attraction of all of these technologies is that numerous genes can be monitored in the same experiment, making it possible to perform a *global* expression analysis of the cell.

A popular application of these technologies has been to globally monitor gene expression during execution of a cellular process or pathway. For instance, a whole-genome microarray has been used to examine several yeast pathways for changes in gene expression over time, including glycolysis[2], cell cycle[3], and sporulation[4], and the expression levels of 112 genes in the rat central nervous system have been

[†] *first authors*

measured over a sequence of developmental stages[5]. In each case, changes in the expression levels of hundreds to thousands of genes were observed over 7-20 time points.

Inspired by experiments such as these, several computational methods have been proposed for analyzing a measured time series of gene expression profiles to infer the underlying *genetic network*[6]. The aim of these methods is to produce a model of the network, describing how the expression level of each gene in the network depends on external stimuli and on the expression levels of other genes. Network interactions are inferred either by identifying statistical correlations between expression levels[7, 8], by training a neural network[9], or through use of information theoretic methods[10].

Additional information about a genetic network may be gleaned experimentally by applying a directed *perturbation* to the network, and observing the steady-state expression levels of every gene in the network in the presence of the perturbation. Perturbations may be *genetic*, in which the expression levels of one or more genes are fixed by deletion or overexpression, or *biological*, in which one or more non-genetic factors are altered, such as a change in growth media, a temperature increase, or the addition of an extracellular ligand. Deletion and overexpression are readily performed on a large scale in yeast (e.g. see the *Saccharomyces Genome Deletion Project*[11]), and are also becoming feasible in other model organisms such as worm, fruit-fly, and mouse as the efficiency of cell transformation improves. Given expression data from a series of perturbation experiments, analytical methods are now needed to infer the underlying genetic network[12]. In this regard, Akutsu *et al.* have derived helpful upper and lower bounds on the number of perturbations that would be required if the network were Boolean[13].

To further address this need, we here describe two analytical methods and an explicit strategy for inferring a Boolean genetic network through perturbation. According to this strategy, the underlying network of interest is exposed to an initial series of genetic and/or biological perturbations and a steady-state gene expression profile is generated for each. Next, a method called the *predictor* is used to infer one or more hypothetical Boolean networks consistent with these profiles. When several networks are inferred, the predictor returns only the most parsimonious, as measured by those networks having the fewest number of interactions.

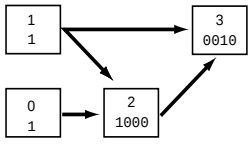
Depending on the complexity of the genetic network and the number of initial perturbations, numerous hypothetical networks may exist. Accordingly, a second method called the *chooser* is used to propose an additional perturbation experiment to discriminate among the set of hypothetical networks determined by the predictor. Since perturbations can be costly in terms of experimental time and money, the chooser uses a function based on entropy to optimally reduce the number of hypothetical networks remaining after the additional perturbation has been performed. The predictor and chooser may be used iteratively and interactively to

successively refine the genetic network: at each iteration, the perturbation selected by the chooser is experimentally performed to generate a new gene expression profile, and the predictor is used to derive a refined set of hypothetical gene networks using the cumulative expression data.

In the sections that follow we review the genetic network model used, describe our implementation of the predictor and chooser, and discuss simulation results exploring the performance of both methods.

2 Model Definition

We implement a genetic network using a deterministic Boolean model similar to several that have been previously described[13, 14]. Briefly, a network is represented as a graph consisting of N numbered nodes a_n ($0 \le n < N$), a topology of directed edges (arrows) between nodes, and a function f_n for each node. A node may represent either a *gene* or a biological *stimulus*, where a stimulus is any relevant physical or chemical factor which influences the network and is itself not a gene or gene product. A node has an associated steady-state expression level x_n, representing the amount of gene product (in the case of a gene) or the amount of stimulus present in the cell. This level is approximated as high or low and represented by the binary value 1 or 0, respectively. An edge directed from one node to another represents the influence of the first gene or stimulus on that of the second, so that the expression level of a node a_n is a Boolean function f_n of the levels of the nodes in the network which connect (have a directed edge) to a_n.



A *A directed graph structure with numbered nodes connected by edges*

x1	1	0	1	0
x2	1	1	0	0
x3	**0**	**0**	**1**	**0**

B *The truth table (shown for node 3 only)*

```
x0 := 1
x1 := 1
x2 := x0 and x1
x3 := x1 and not x2
```

C *The logic equations for each node*

Figure 1: Example of the Boolean steady-state network model

An example of a small network of four nodes is shown in figure 1. In figure 1A, each node a_n is represented by a box, with the node number n given in the upper half of the box. In the lower half of the box is a numerical code representing the function f_n associated with the node. This code is extracted from the output row of a *truth table*, an example of which is shown in figure 1B. A truth table provides a unique description of the Boolean function for a node, and the code (shown in bold) represents the output level x_n corresponding to each possible combination of input

levels. In the example, the level of gene a_3 is determined by the levels of genes a_1 and a_2 according to the associated function represented by the binary code 0010. The steady-state expression level x_3 will be high only when x_1 is high and x_2 is low. The equations in figure 1C show an alternative representation of the topology of the network and its functions.

In order to infer a genetic network of this type, a population of cells containing a target genetic network T is monitored in the steady state over a series of M experimental perturbations. In each perturbation p_m ($0 \leq m < M$) any number of nodes may be forced to a low or high level. Genes may be perturbed through laboratory methods for gene deletion or overexpression, while stimuli are perturbed by altering the environment outside the cell. The observed steady-state expression levels for all genes and stimuli over all experimental perturbations are represented by the expression matrix $\boldsymbol{E}$. Figure 2 shows an expression matrix generated by several illustrative perturbations to the example genetic network of figure 1. Rows of $\boldsymbol{E}$ represent perturbation conditions while columns represent node values in each steady-state condition, such that matrix entry $\boldsymbol{E}_{mn}$ is the expression level of node a_n in the presence of perturbation p_m. The symbols + and – are used to show that a node has been forced to a high or low value, respectively. For example, in perturbation p_3, node a_2 has been forced low so that $x_2 = 0$. The set of all expression levels of all nodes for perturbation p_m is termed a *network state*. Note that perturbation p_0 is what is known to a geneticist as a *wild-type state*, because no nodes have been forced high or low.

$\boldsymbol{E} =$

x_0	x_1	x_2	x_3	
1	1	1	0	p_0
-	1	0	1	p_1
1	-	0	0	p_2
1	1	-	1	p_3
1	1	1	+	p_4

Figure 2: Example expression matrix generated from the genetic network in fig. 1.

3 Methods for Inference and Experimental Design

3.1 Inference of Genetic Networks Using the Predictor

We now describe the *predictor*, a method for inferring Boolean networks using the expression data contained in the matrix $\boldsymbol{E}$. In order to arrive at a Boolean function f_n independently for each node a_n, we determine a minimum set of nodes whose levels must be included as input variables of f_n to explain the observed data in $\boldsymbol{E}$, then construct a truth table using these nodes as inputs. Specifically, the function for node a_n is determined according to the following procedure:

(1) Consider all pairs of rows (i, j) in $\boldsymbol{E}$ in which the expression level of a_n differs, excluding rows in which a_n was itself forced to a high or low value. For each pair, find the set S_{ij} of all other nodes whose expression level *also*

differs between the two rows (i, j). *Because the network is self-contained, a change in at least one of these genes or stimuli must have caused the corresponding difference in* a_n. Therefore, at least one node in this set must be included as a variable in f_n.

(2) Identify the smallest set of nodes S_{min} required to explain the observed differences over all pairs of rows (i, j), so that at least one node in S_{min} is present in each set S_{ij}. This task is a classic combinatorial problem called *minimum set covering* which can be solved by the *branch and bound* technique[15]. More than one smallest set S_{min} may be found, in which case a separate function f_n is inferred and reported for each set.

(3) Once S_{min} has been determined for node a_n using the branch and bound procedure, a truth table is determined for f_n in terms of the levels of genes and/or stimuli in S_{min} by taking relevant levels directly from $\boldsymbol{E}$. If all combinations of input levels are not present in $\boldsymbol{E}$, the corresponding output level for gene a_n cannot be determined and is represented by the symbol "*" in the truth table.

As an example, consider how this method is used to infer the function for a_3 from the expression matrix $\boldsymbol{E}$ shown in figure 2. Since the expression level of a_3 is equal to 0 in rows p_0 and p_2 and equal to 1 in rows p_1 and p_3, in step (1) we examine the row pairs *(0,1), (0,3), (1,2),* and *(2,3),* all of which differ in the expression level of a_3. We exclude row p_4, in which node a_3 is itself perturbed. Next, nodes that are in the set S_{ij} are determined for each pair. The difference in level of a_3 between rows p_0 and p_1 could have been caused by a change in level of a_0, a_2, or both, thus $S_{01} = \{a_0, a_2\}$. In a similar fashion, we find that set $S_{03} = \{a_2\}$, set $S_{12} = \{a_0, a_1\}$ and set $S_{23} = \{a_1\}$. In step (2), the minimum set which covers all four sets is found to be $S_{min} = \{a_1, a_2\}$.

For step (3), we determine the truth table for x_3 in terms of x_1 and x_2. We begin by searching rows of $\boldsymbol{E}$ for the four combinations of binary values for x_1 and x_2. In row p_0 and p_1, we find the pair $\{x_1, x_2\}$ equals {1, 1} and {1, 0}, respectively. In row p_2 we find that $\{x_1, x_2\}$ = {-, 0}, but since the symbol "-" means that x_1 has been forced low, we take $x_1 = 0$. Likewise, in row p_3, we find $\{x_1, x_2\}$ = {1, -} = {1, 0}. As before, row p_4 is excluded from the analysis. Since the remaining state $\{x_1, x_2\}$ = {0, 1} is never observed in $\boldsymbol{E}$, we don't know its effect on x_3 and represent this lack of knowledge by inserting the symbol "*" in the truth table. Thus, the final function code for x_3 is 0*10. Except for our lack of knowledge about the value of x_3 when $\{x_1, x_2\} = \{0, 1\}$, our truth table exactly reproduces the original shown in figure 1B.

If a node a_n in the network is found to have more than one smallest set S_{min}, the algorithm infers several networks, each with a distinct function f_n corresponding to each S_{min}. If several such nodes exist, a separate network *hypothesis* is returned for each distinct combination of functions at each node. As an example, if nodes a_2 and a_4 each have two possible functions, a total of $2 \times 2 = 4$ networks will be returned.

The minimum set cover also ensures that only the most *parsimonious* networks will be returned, in the sense that they are consistent with ***E*** *and* have the fewest edges possible. If desired, the algorithm may easily be modified to produce networks whose number of edges is not strictly minimal.

When selecting among several functions for each node, we have chosen to restrict consideration to *acyclic* network models. This has the technical advantage that the steady-state behavior is uniquely determined independent of assumptions about the time delays of the components. For genetic networks that involve a sequential progression of biochemical interactions with few feedback loops (e.g. see Hereford *et al.*[16]), an acyclic network description may be biologically informative even in cases where it is not strictly accurate.

The analysis of cyclic networks is complicated by the possibility of oscillatory behavior. For cyclic networks, one may adopt either a *synchronous* model in which each component has a fixed, known delay, or an *asynchronous* model in which the delays are unknown and even nondeterministic. The methods of this paper can easily be extended to apply to the synchronous model. The inherent nondeterminism of the asynchronous model introduces the further complication that many possible steady states may exist, and we are currently exploring the extent to which our methods can be extended to handle this case.

3.2 Design of Experiments with the Chooser

The *chooser* uses an entropy-based procedure to select an additional perturbation experiment to discriminate among hypothetical networks generated by the predictor (Section 3.1). Assume that the predictor uses an initial expression matrix ***E*** to generate L equally parsimonious, equiprobable networks and that these are hypotheses as to the real underlying target network T. The problem is now to choose a new perturbation experiment p, from a supplied set P of perturbation experiments under consideration, which will best discriminate between the L hypothetical networks[17]:

1) For each perturbation p in P, compute the network state resulting from p for each of the L networks. A given perturbation will result in a total of S distinct states s over the L networks ($1 \le S \le L$, $1 \le s \le S$). Denoting the number of networks giving state s by l_s, evaluate the entropy score H_p according to the expression:

$$H_p = -\sum_{s=1}^{S} \frac{l_s}{L} \log_2\left(\frac{l_s}{L}\right)$$

2) Choose the perturbation p giving the maximum value of H_p as the next experiment.

H_p may be interpreted as the expected decrease in entropy (uncertainty as to which of the L networks is the true network T) for perturbation p, and is therefore a measure of the expected information gained in performing p. Complete reduction in uncertainty is obtained under perturbation p if all L networks produce L distinct states ($H_p = \log_2 L$), while no information is gained if all L networks produce an identical state ($H_p = 0$). Note that this approach is one of many which could be used to score a given perturbation p. For example, since the number of nodes forced to a high or low level for a given perturbation p may impact the difficulty of performing the perturbation in the laboratory, a cost function which penalizes those p having large numbers of forced nodes may be appropriate.

Once chosen, the new perturbation is performed experimentally on T and the measured gene expression values added to $\boldsymbol{E}$. A new set of parsimonious networks is then inferred with the predictor, another perturbation experiment to discriminate between these is selected by the chooser, and so on. This design process proceeds iteratively, choosing a new perturbation experiment in each iteration, until either a single parsimonious network remains ($L = 1$) or no perturbation in P can discriminate between any of the L networks ($H_p = 0$).

The L hypothetical networks are not always completely specified, sometimes having a "*" symbol in the function of one or more nodes (see section 3.1). In order to complete the entropy calculation for these networks, we randomly choose a 0 or 1 to replace the undetermined "*" value. In addition, when L is large it may be infeasible to calculate entropy over all hypothetical networks. In this case, we randomly sample a smaller number of these networks for the entropy calculation.

4 Results

4.1 Evaluation of the Predictor Using Simulated Networks

To evaluate the performance of the predictor, a series of target genetic networks and network perturbations were simulated to produce test sets of gene expression data. These data were analyzed by the predictor and the inferred networks were compared to the original simulated target networks. To construct a target network T of size N and maximum in-degree k (where the in-degree of a node is its number of incoming edges), network edges were chosen randomly so that the indegree of each node was distributed uniformly between 1 and k, and restricted so that the network contained no cycles. Functions were then chosen for each node by randomly generating an output value (0 or 1, low or high) for each possible combination of input expression levels, but restricted to ensure that the output expression level was dependent on each input. The wild-type perturbation (with no nodes fixed high or low) and all N single perturbations of T, one for each node, were simulated on each network. Note that deleting a gene which already has a value of 0 in the wild-type state provides no

new information, nor does over-expressing a gene with a value of 1 in the wild-type state. Therefore, each node was perturbed away from the wild-type state.

Target networks were simulated in this manner over a range of network sizes N and maximum in-degrees k, and the most parsimonious networks were inferred in each case. The similarity between each inferred network and its target was evaluated with regard to *sensitivity*, defined as the percentage of edges in the target network that were also present in the inferred network, and *specificity*, defined as the percentage of edges in the inferred network that were also present in the target network. Results from a series of simulations over a range of N and k are shown in Table 1. For each choice of network size, 200 target networks were generated. Note that specificity was always significantly higher than sensitivity, and that both steadily decreased with increasing N and k. Also, the number of nodes whose functions had only a single minimal solution (see step 2 in section 3.1) was approximately 90% for $k = 2$, independent of N. Thus although the number of inferred networks grew exponentially with N, this number was consistently due to ambiguities at just 10% of the nodes.

A number of non-random target network topologies modeled after known biological networks were also simulated and inferred. The well-studied networks responsible for bacterial chemotaxis[18], galactose induction in yeast[19], and yeast meiosis[20] were encoded using our network representation, and expression levels from the wild-type state and all single perturbations were simulated as before. In all

TABLE 1: Run statistics for the *predictor* method over a range of N and k. Each measurement is an average over 200 simulated target networks, with the standard error given in parentheses. Standard error for columns G and H was always less than 1% and is not listed. For each simulation of size N and k (given in columns A and B), the number of edges in the target network was recorded along with the number of parsimonious inferred networks returned (columns C and D). The number of edges in the parsimonious inferred networks was also calculated (E), as well as the avg. number of edges shared between the target network and each inferred network (F). These figures were used to calculate sensitivity (G) and specificity (H), also shown. Finally, the avg. number of nodes whose functions have only a single minimal solution was recorded (I), as well as the avg. CPU time required to complete the inference procedure (J). CPU time was computed using a 500 MHz Pentium III processor.

A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*
N	***k***	**Total Sim. Edges**	**Num. Inferred Networks**	**Total Inferred Edges**	**Num. Shared Edges**	**Sens-itivity**	**Spec-ificity**	**Num. Nodes w/ 1 Soln.**	**CPU Time (sec)**
5	2	4 (0.1)	1 (.02)	3 (0.1)	3 (0.1)	77%	99%	5 (0.0)	0.1 (0.0)
10	2	12 (0.1)	60 (50)	9 (0.1)	9 (0.1)	71%	95%	9 (0.1)	0.1 (0.0)
20	2	27 (0.2)	3×10^{7} (10^{7})	21 (0.2)	19 (0.1)	71%	92%	18 (0.1)	0.2 (0.0)
50	2	72 (0.2)	1×10^{12} (10^{12})	57 (0.3)	51 (0.3)	71%	90%	45 (0.2)	0.8 (0.0)
100	2	146 (0.7)	3×10^{26} (10^{26})	119 (0.9)	104 (0.7)	70%	88%	89 (0.5)	6.6 (0.3)
20	4	44 (0.3)	2×10^{6} (10^{6})	28 (0.3)	23 (0.2)	51%	84%	16 (0.1)	0.2 (0.0)
20	6	57 (0.5)	2×10^{7} (10^{7})	33 (0.3)	27 (0.2)	42%	82%	14 (0.2)	0.2 (0.0)
20	8	69 (0.7)	9×10^{7} (10^{8})	38 (0.4)	31 (0.3)	35%	82%	13 (0.2)	0.2 (0.0)

three cases several parsimonious networks were inferred, some of which resembled or corresponded exactly to the original simulated network. Detailed data and results from these simulations will be provided in a future publication.

4.2 Evaluation of the Chooser

In order to evaluate the proposed method for experimental design, a randomly generated target network T (N=20, k=4, 24 edges) was simulated in the *wild-type* state and under N single-node perturbations (as in section 4.1) to produce an expression matrix $\boldsymbol{E}$. Eight parsimonious networks, all with 21 edges, were consistent with $\boldsymbol{E}$. To discriminate between these networks, the chooser was used to select, *from the set of all double perturbations*, a double perturbation which had maximum score H_p over the eight networks. This process of choosing double perturbations was repeated iteratively until only a single network was inferred (a total of 16 iterations), identical to T in this case. Note that in general, the number of distinct double perturbations evaluated at each iteration was equal to $4 \cdot (N \textit{ choose } 2)$, because each node may be forced high or low.

Figure 3 tracks changes in the number of inferred networks, the number of edges in each inferred network, average sensitivity, and average specificity as new double perturbations were added to $\boldsymbol{E}$ over each iteration of the design process. Note that on the third, fifth, and tenth double perturbations performed, the number of inferred networks grew, since it was found that none of the original solutions with 21 edges were consistent with the new data. This characteristic pattern of jumps and decays in the number of network solutions, correlated with a monotonic increase in the number of inferred edges, was observed consistently over many other simulations using a wide range of different target networks T (data not shown).

It is interesting to study global features of the design process as a function of N

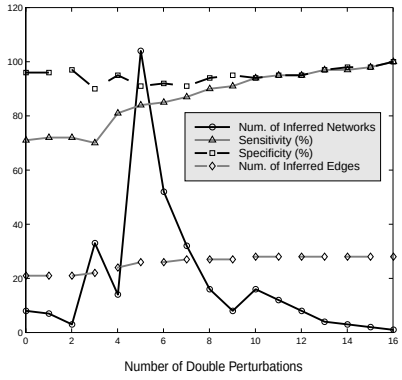


Figure 3: Progress through experimental design

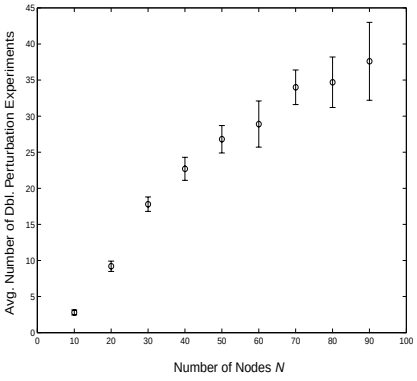


Figure 4: Average number of perturbation experiments vs. network size N for k = 2, with bars indicating standard error.

and k. A theoretical lower bound on the number of gene expression profiles which must be observed to uniquely specify a genetic network has been reported[21] to be $k \cdot \log_2(N / k)$ for $N >> k$. In order to characterize the performance of our methods in relation to this lower bound, we generated 50 target networks T for each of several values of N with $k = 2$. The wild-type state and all single perturbations were simulated on each T, and as before the chooser was used iteratively in conjunction with the predictor to select a series of double perturbation experiments to refine the network hypotheses until the iteration terminated (see section 3.2). The average number of double perturbation experiments required for each N is shown in figure 4. These preliminary results show evidence of logarithmic behavior.

5 Discussion

We have demonstrated a new method for inferring genetic networks from gene expression data monitored over a series of perturbations. We have also described a design process by which specific perturbation experiments may be chosen to further reduce a given set of inferred networks or to increase their specificity and sensitivity. This process is automatable and involves an experimentally tractable number of perturbations. Although the minimum set covering problem (see section 3.1) is known to be NP-complete, it may be solved in polynomial time if the maximum-indegree k is fixed. In this regard, there may exist biological rules governing the range of values for k, as suggested by the recent observation that in general, the cis-regulatory regions of genes are organized into modules bound by four to eight transcription factors each[22].

Our representation of a genetic network, while simple and abstract, provides a flexible description of a biochemical pathway. The level associated with each node could potentially refer to mRNA abundances, protein activities, or concentrations of other macro-molecules. Different levels of regulation, including transcription, translation, and protein modification, could be included in the model by defining several nodes per gene, each corresponding to a different level of regulation. Molecular compartmentalization may be treated analogously.

The predictor and chooser methods presented here are preliminary: a number of extensions, most of them straightforward, will likely be required to make the methods useful to the laboratory biologist. First of all, in nearly all cases of practical interest *some* knowledge of network genes and interactions is available. In this regard, pre-existing information about the network may be incorporated during the inference process. For instance, the user might require that all inferred networks contain certain edges or node functions, or might *disallow* certain node functions as being biologically infeasible. Furthermore, the genes known to be involved in a network could be deleted or overexpressed to generate the initial set of perturbation experiments analyzed by the predictor.

Second, the Boolean representation of gene expression assumed here is merely a starting point. Although arbitrary data may be expressed in Boolean form through the use of thresholding, the observed levels of gene products and other macromolecules may be such that a two-level description misses important features of the network. In these cases, a multi-level description (greater than two) may be adequate to describe the data. For example, transcript levels could be described by one of three states: absent, basal, or induced. Our proposed methods can be extended to multi-level data with relatively little modification. It may also be possible to extend the method for use with continuous (rather than discrete Boolean or multi-level) gene expression data, but this is an open problem.

Third, in this treatment we have only considered genetic networks which do not contain cycles. This restriction may be sufficient to describe certain biochemical networks, but biological examples of cyclic gene networks are also known (e.g. see McAdams *et al.*[23]). Therefore, another future direction is to allow cyclic solutions in the inference procedure, but this will require treatment of the additional complication of oscillatory or non-deterministic behavior.

Fourth, we currently do not allow for noise or other imperfections in the gene expression data sets used for network inference. Gene expression levels measured with DNA microarrays or other technologies are subject to an appreciable amount of experimental variability, and the impact of this variability on our method should be evaluated. It seems plausible that the inference method could be modified to account for noisy data, but this also remains an open problem.

Finally, we expect that the proposed methods may be most effective when used in conjunction with existing software for grouping genes. For instance, a clustering algorithm might be used to reduce the apparent size of the network by grouping genes according to similar expression level over the series of perturbations performed, then one representative from each cluster could be supplied to the network inference method. Alternatively, the set of all genes could be partitioned into strongly interacting subsets, with a single subset processed as a separate network by the predictor. Genes could also be grouped according to similarities in nucleotide or amino acid sequence annotation, for example clustering genes with common transcription factor binding sites.

Acknowledgments

We wish to thank Jeremy Buhler, Rimli Sengupta, David Haynor, and Leroy Hood for numerous helpful suggestions during the development of this research. The work of T.I. and V.T. was supported in part by a Univ. of Washington Training Grant in Interdisciplinary Genome Sciences. In addition, the work of V.T. was supported by a Sloan Foundation/DOE Fellowship in Computational Molecular Biology, and the work of T.I. was supported by a fellowship from the ARCS Foundation.

References and Notes

1. S. P. Gygi, Y. Rochon, B. R. Franza, R. Aebersold, *Mol Cell Biol* **19**, 1720-30 (1999).
2. J. L. DeRisi, V. R. Iyer, P. O. Brown, *Science* **278**, 680-6 (1997).
3. P. T. Spellman, et al., *Mol Biol Cell* **9**, 3273-97 (1998).
4. S. Chu, et al., *Science* **282**, 699-705 (1998).
5. X. Wen, et al., *Proc Natl Acad Sci U S A* **95**, 334-9 (1998).
6. A genetic network may be described as a collection of molecular components, such as genes, and interactions between them that collectively carry out a cellular function.
7. A. Arkin, J. Ross, *J. Phys. Chem.* **99**, 970 (1995).
8. N. Friedman, I. Nachman, D. Peer, ISMB99, Heidelberg (AAAI Press, 1999).
9. D. C. Weaver, C. T. Workman, G. D. Stormo, Pacific Symposium on Biocomputing, Hawaii, Hawaii (1999).
10. S. Liang, S. Fuhrman, S. Somogyi, Pacific Symposium on Biocomputing, Maui, Hawaii (1998).
11. E. A. Winzeler, et al., *Science* **285**, 901-6 (1999).
12. Because steady-state data are analyzed, such methods do not require assumptions about the time behavior of genetic networks, such as the assumption that all network nodes update synchronously (e.g. see ref. 14).
13. T. Akutsu, S. Kuhara, O. Maruyama, S. Miyano, Proc of the 9th ACM-SIAM Symposium on Discrete Algorithms, (ACM Press, 1998).
14. R. Somogyi, C. Sniegoski, *Complexity* **1**, 45-63 (1996).
15. G. L. Nemhauser, *Integer and combinatorial optimization* (Wiley, New York, 1988).
16. L. M. Hereford, J. H. Hartwell, *J Mol Biol* **85**, 445-61 (1974).
17. This formulation of the problem results in a greedy algorithm, which chooses the single best experiment to perform next, rather than an optimal sequence of multiple experiments.
18. U. Alon, M. G. Surette, N. Barkai, S. Leibler, *Nature* **397**, 168-71 (1999).
19. D. Lohr, P. Venkov, J. Zlatanova, *Faseb Journal* **9**, 777-87 (1995).
20. A. P. Mitchell, *Microbiol Rev* **58**, 56-70 (1994).
21. J. Hertz, http://www.nordita.dk/~hertz/projects.html, Pacific Symposium on Biocomputing, Maui, Hawaii (1998).
22. M. I. Arnone, E. H. Davidson, *Development* **124**, 1851-64 (1997).
23. H. H. McAdams, L. Shapiro, *Science* **269**, 650-6 (1995).