INFLUENCE OF NETWORK TOPOLOGY AND DATA COLLECTION ON NETWORK INFERENCE

V. ANNE SMITH, ERICH D. JARVIS
Department of Neurobiology, Duke University Medical Center, Box 3209
Durham, NC 27710, USA

ALEXANDER J. HARTEMINK
Department of Computer Science, Duke University, Box 90129
Durham, NC 27708, USA

We recently developed an approach for testing the accuracy of network inference algorithms by applying them to biologically realistic simulations with known network topology. Here, we seek to determine the degree to which the network topology and data sampling regime influence the ability of our Bayesian network inference algorithm, NETWORKINFERENCE, to recover gene regulatory networks. NETWORKINFERENCE performed well at recovering feedback loops and multiple targets of a regulator with small amounts of data, but required more data to recover multiple regulators of a gene. When collecting the same number of data samples at different intervals from the system, the best recovery was produced by sampling intervals long enough such that sampling covered propagation of regulation through the network but not so long such that intervals missed internal dynamics. These results further elucidate the possibilities and limitations of network inference based on biological data.

1 Introduction

Large amounts of data on the expression of genes have recently become available due to advances in gene expression array analysis. Functional network inference algorithms have begun to be used to draw causal information from this correlational data, producing networks representing putative regulatory interactions of hundreds to thousands of genes (for an early review, see [1]). However, a difficulty arises in evaluating the success of these algorithms, because no high throughput method of assessing genetic regulatory influences yet exists. To experimentally validate all links in even a single putative network, using antisense blocking or other methods, could take impractically long given current technology.

In Smith et al. [2] and Jarvis et al. [3] we developed an approach to circumvent this problem, by creating a biologically realistic computer simulation of a system in which we make and know all the rules. We run the simulation representing the behaving organism and sample data from it as one would in a real biological experiment. We then present this sampled data to network inference algorithms. We compare putative networks produced by the algorithms to the known truth underlying the simulation, and thus evaluate the success of the algorithms. Additionally, because we sample data from the simulation, we can also use our framework to evaluate the effect of differing sampling regimes on the output of the

inference algorithms. This ability has implications for designing biological experiments for use with network inference algorithms[1].

In this paper, we use our approach to evaluate the degree to which the recovery performance of our Bayesian network inference algorithm depends on the topology of the network being recovered. In particular, we examine various network topologies exhibiting features commonly found in genetic regulatory pathways like feedback loops, multiple regulators of a single gene, and multiple targets of a single regulator. We also evaluate its success on two complex pathways of biologically realistic topology, a gene cascade and a convergence network. In addition, we use our approach to further examine the effect of sampling regime on network inference results. Previously, we showed that sampling at an interval that matches underlying system dynamics produces considerably better results than sampling with an interval twice as long [2]. Here, we investigate the issue more closely to determine the extent of the usable range of sampling intervals.

2 Approach and Methods

2.1 Simulator

Our simulator, BRAINSIM, is based on the vocal communication system of the songbird brain [3]. Songbirds are vocal learners, along with only five other animal groups, including humans, and thus provide a model system for understanding human language [5]. Songbird vocal communication has been studied extensively at the behavioral, electrophysiological, neuroanatomical, and molecular levels of organization [6]. This simulator is part of an integrative project that aims to synthesize information across multiple levels of biological organization [3]. As vocal learning represents an integration of motor (singing) and sensory (hearing) systems, such synthesis in the songbird is expected to have implications for a wide variety of brain systems.

BRAINSIM is programmed in C++ and simulates behavior, electrophysiological activity in distinct brain regions, and gene expression in these brain regions for six subject birds (Fig. 1). A detailed account of BRAINSIM's implementation is found in Smith et al. [2]. Briefly, the simulation progresses in discrete time steps, each step representing approximately one minute. Behavior of the animal takes one of two values, 0 or 1, which correspond to off-on states of a behavior, such as silence and singing. At each time step, activity in four brain regions ranges from 0-400 Hz to simulate recordings from extracellular multi-unit electrodes [7], and is correlated

[1] After this paper was submitted, an independent generation of a similar approach by Zak et al. [4] was brought to our attention, using a different type of simulation. Zak et al. evaluate linear, linear-log, and linear-squashing network inference algorithms.

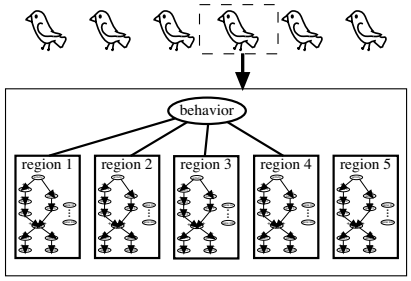

Figure 1. Overview of BRAINSIM. Each simulated data set consists of behavior, activity in five brain regions, and expression levels of 100 genes in each region, for six subjects. A small fraction of the 100 genes are regulated as part of a pathway; the remainder wander randomly. Specific pathway topologies we considered are found in Fig. 2. Figure modified from [2].

either positively or negatively with behavior. Activity in the fifth region wanders randomly in the same range. Three factors influence gene expression level, whose regulatory network is assumed to be the same in all brain regions. First, a returning function moves each gene towards its constitutive expression level, representing degradation of mRNA or return to transcription after suppression. Second, for genes in a pathway, gene regulation is implemented by adding (for up-regulation) or subtracting (for down-regulation) a proportion of the regulator's expression level in the previous time step. This proportion is set at 0.2 but can be varied, and is the **influence** of a regulator on its target. Genes not in a pathway add or subtract a random amount from their previous expression level to simulate influence from other unmeasured processes. Third, a final random amount is added or subtracted from all genes to simulate stochasticity in gene expression [8]. All genes have a minimum possible expression level of 0 and a maximum of 50, in arbitrary units.

For this paper, the simulation began with behavior at 0 and was run for 250 time steps to stabilize the system. Behavior then switched to 1 for 50 time steps, and back to 0 for 100 more time steps. This represents a likely experimental situation where an animal would perform a behavior, such as singing, for a set amount of time in response to some stimulus, such as presentation of a potential mate.

2.2 Network inference algorithm

Our inference algorithm, NETWORKINFERENCE, described in detail previously [2,9], is written in C and uses simulated annealing to identify a high scoring network from data using a discrete Bayesian network framework. A Bayesian network is a graph-based representation of a joint probability distribution, with nodes connected by links that represent conditional dependence. A link from one node to another indicates that the child node is conditionally dependent on the parent node. Unlike many other network inference frameworks, Bayesian networks are capable of representing combinatorial, nonlinear, and stochastic relationships, such as are often found in biological systems. Additionally, due to their probabilistic nature, Bayesian networks can handle noisy data [10]. While static Bayesian networks are limited to

having no cycles, regulatory cycles like those found in biological systems can be handled using dynamic Bayesian networks (DBN). A DBN represents all elements at time t and $t+\Delta t$, with controlling elements at time t having links to controlled elements at time $t+\Delta t$. For example, a cyclic interaction in time between nodes A and B can be represented acyclically in a DBN using links from $A(t) \rightarrow B(t+\Delta t)$ and $B(t) \rightarrow A(t+\Delta t)$ [11]. NETWORKINFERENCE is designed to recover DBNs from pairs of data points, disallowing links that flow backwards in time, i.e., from time $t+\Delta t$ to time t [2].

2.3 Genetic regulatory network topology

We wished to examine the performance of our algorithm on three types of topologies found in genetic regulatory pathways: (1) multiple regulator genes of a single target gene, (2) regulatory loops, and (3) multiple targets of a single regulator. We simulated networks to test each topology, and also tested multiple regulators and multiple targets in two large, more biologically realistic topologies.

We considered two types of multiple regulators, one where two inputs to one gene were governed by pathways originating from a common variable and were thus **correlated**, and another where the two inputs were governed by **uncorrelated**

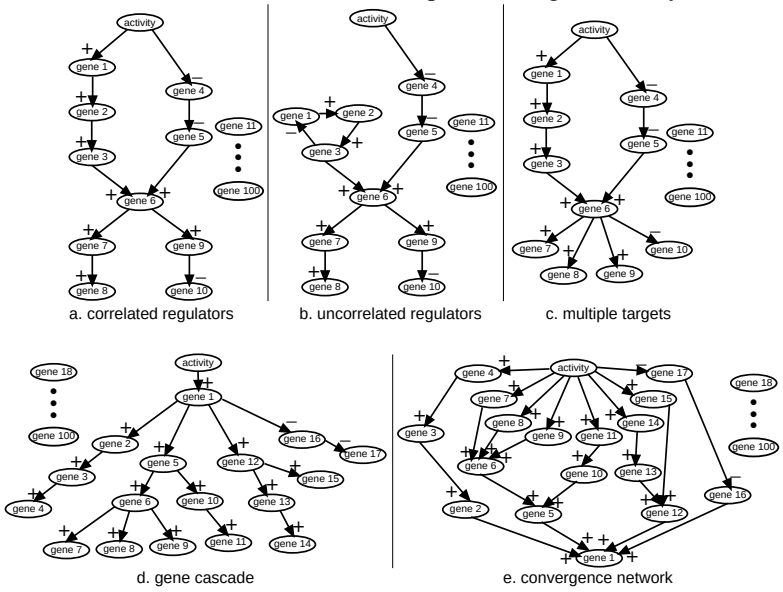

Figure 2. Genetic regulatory network topologies. Different topologies for electrical-genetic regulation pathways. All topologies contain 100 genes, with those not in the pathway wandering randomly. **a.** Genes 3 and 5 serve as correlated regulators of gene 6. **b.** Genes 3 and 5 serve as uncorrelated regulators of gene 6, and genes 1, 2, and 3 from a regulatory loop. **c.** Genes 7-10 serve as multiple targets of gene 6. **d.** A regulatory cascade, with increasing branches of influence from gene 1. **e.** A convergence network, with regulatory paths of varying lengths of regulation converging on a single target.

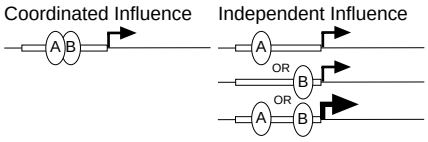

Figure 3. Coordinated versus independent influence. In the case of coordinated influence, both regulators A and B must bind to the promoter region (boxed area) to activate transcription of the gene (right of boxed area, transcription noted by arrow). In the independent case, either A or B alone is capable of activating transcription but transcription is elevated with both present (additive effect, thicker arrow).

pathways. The correlated regulators were produced using the topology we developed previously [2], where gene 6's regulators, 3 and 5, are both downstream of activity (Fig. 2a). Uncorrelated regulators of gene 6 were created by modifying the topology of this network so that genes 1, 2, and 3 formed a self-generating loop uncorrelated with activity (Fig. 2b). With multiple regulators, the manner of combining their influences on the target needs to be considered. We simulated **coordinated influence**, which models a situation where regulation of the target gene requires a molecular complex formed from all regulators, and **independent influence**, which models a situation where regulation of the target gene requires any of the regulators singly, and their influence is additive (Fig. 3) [12]. For independent influences, we tested two situations: one where the two influences on gene 6 were equal to the other influences in the network, i.e. 0.2, such that each link was as "strong" as any other, but also such that gene 6's maximum possible up-regulation was twice as fast as any other gene; and another where these influences were 0.1, making gene 6's dynamics equal to other genes in the network but making the influences from each of its regulators "weaker". All other links in the network had influences of 0.2.

The self-generating loop with genes 1, 2, and 3 in the topology for uncorrelated multiple parents served as the feedback loop for testing (Fig. 2b).

To test multiple targets of a single regulator, we simulated a topology where gene 6 directly regulated four other genes (Fig. 2c). The multiple regulators in this topology were implemented with the coordinated influence rule.

Finally, we tested two complex pathways that were designed to be more biologically realistic: a cascade, which contains multiple targets of single regulators (Fig. 2d), and a convergence network, which contains multiple regulators of single targets (Fig. 2e). The cascade represents an expected topology, such as in a signal transduction pathway that might be triggered by activity in a neuron [12,13]. The convergence network represents another expected topology, where a single gene is regulated by multiple branches with different path lengths from the original triggering activity [12,13]. All multiple regulators in the convergence network were implemented with the coordinated influence rule.

2.4 Sampling regime

These network topologies were sampled using the regime described in [2], sampling

at an interval that intuitively matched the system, every 5 time steps, following the reasoning that a 0.2 influence each time step leads to a full influence in 5 time steps. Sampling began just before the animal started its behavior (value switched to 1), and covered the period of the behavior and an equal amount of time after its cessation (value 0), for a total of 21 samples. We also investigated the effect of varying the sampling interval, using the original regulatory topology (Fig. 2a). We sampled at intervals of 1-4, 6-9, 15, 20, and 30 time steps, and examined these along with previous results for intervals of 5 and 10. In order to maintain the same number of data points across the behavior and its cessation, behavior length was adjusted to the sampling interval. For example, when sampling at an interval of 3, instead of behavior lasting for 50 time steps, behavior lasted 30 time steps; when sampling at an interval of 15, behavior lasted 150 time steps. The simulation was run long enough to collect 21 data points in each situation. This is experimentally reasonable, as it possible to adjust the duration of stimulus presentation to an animal or allow an animal to continue to perform its behavior.

2.5 Data generation and discretization

Each data set consisted of sampled data from a simulation containing six subjects. This represents a biologically reasonable maximum sample size for gene expression studies (21 time points x 6 animals per time point = 126 animals). Although not expected to be carried out experimentatally, for statistical purposes we generated multiple data sets to evaluate the robustness of the algorithm to random variation in data with the same structure. Ten data sets were collected for each genetic network topology and sampling interval, and presented to the network inference algorithm independently.

Since NETWORKINFERENCE searches for discrete Bayesian networks, we must present it with data that has been discretized. A smaller number of discrete states reduces computational time and smooths out random noise; a larger number retains more of the information present in the original data. We chose to use four discretization levels as a balance. As in Smith et al. [2], we discretized activity and gene expression levels into quartiles, with the lowest 25% of values being state 1, the next 25% state 2, and so on.

2.6 Search settings and parameters

Because in this study we were interested in evaluating the ability of the NETWORKINFERENCE algorithm to infer different types of genetic network structures, we only presented the inference algorithm the task of determining relationships between activity and gene expression within brain regions and not between activity and behavior as in [2]. To infer a dynamic Bayesian network, we

presented to the algorithm pairs of sampled data points at time t and time $t+\Delta t$ for the activity and values of all 100 genes. The 21 sampled points thus corresponded to 20 sets of pairs. For the five brain regions of the six birds in each data set, this resulted in 600=20x5x6 observations for 202 variables (activity plus 100 genes each at two time points). NETWORKINFERENCE had no prior information about which genes were regulated and which wandered randomly. NETWORKINFERENCE was run approximately 160 minutes on each dataset on a 1GHz Pentium III CPU or 100 minutes on a 1.6GHz Athlon CPU, both running Linux. The algorithm searched an average of 23.6 million structures (range 20.5-24.7 million) and found the best scoring structure within the first 3.8 million on average (range 0.6-23.1 million).

3 Results

3.1 Influence of genetic regulatory network topology

We tested the ability of NETWORKINFERENCE to recover a network with coordinated influence of correlated regulators in our previous report [2] and these results are shown for comparison (Fig. 4, top left). As noted in [2], NETWORKINFERENCE recovered nearly 100% of the topology, only missing one of two parents of gene 6. When we simulated a network with independent influence of correlated regulators, NETWORKINFERENCE performed well, but still did not recover the two parents and instead linked a gene upstream of one of gene 6's parents (gene 2) as the sole parent for over half of the data sets, and its other parent (gene 5) as the sole parent in the rest, for both influence amounts tested (Fig. 4, top right two graphs).

When testing coordinated influence from uncorrelated regulators, no parents were recovered for gene 6 in most data sets (Fig. 4, bottom left). With independent influences of genes 3 and 5 on gene 6 set at a weight of 0.2, gene 5 was found as the parent in all 10 data sets. However, many unregulated genes were incorrectly linked to the downstream genes in over half the data sets, a phenomenon that did not occur in any other situation (Fig. 4, bottom center). With independent influences set at 0.1, gene 5 was linked as a parent in 4 of 10 data sets (Fig. 4, bottom right).

The loop structure was recovered completely in 100% of the data sets tested (Fig. 4, bottom graphs). Recovery of multiple targets was also highly successful (Fig. 5).

NETWORKINFERENCE performed well in recovering the gene cascade, finding most links with reasonable consistency (Fig. 6). It performed less well in recovering the convergence network, only finding one parent of each gene with multiple parents. This was usually the gene with the shortest path length from the original influence at activity. When path length of multiple regulators was the same, it found each in reasonably equal proportions of the data sets (Fig. 6).

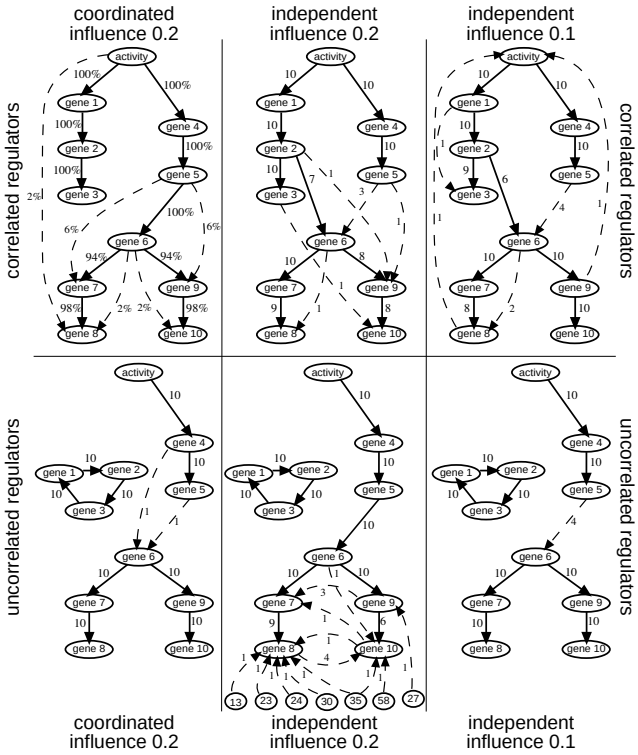

Figure 4. Multiple regulator and feedback loop recovery. The number of data sets in which a particular link was recovered is shown next to the link. Links discovered in at least half of the data sets are shown as solid lines; those in less than half are dashed. The upper left graph shows the result of the original structure from [2], with percentages of data sets instead of absolute numbers reported. In all cases except for the lower center, all 90 unregulated genes were correctly identified as having no part in any pathway. In the lower center, incorporated unregulated genes are shown as numbers in circles. All graphs depict a causal translation of the recovered DBN. Compare with the true structure in Fig. 2a for the top graphs; Fig 2b for the bottom.

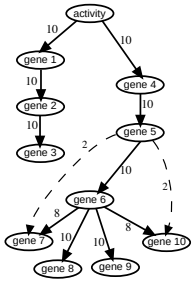

Figure 5. Multiple target recovery. Numbers next to links represent the number of data sets in which that link was found, as described in Fig. 4. Compare with the true structure in Fig. 2c.

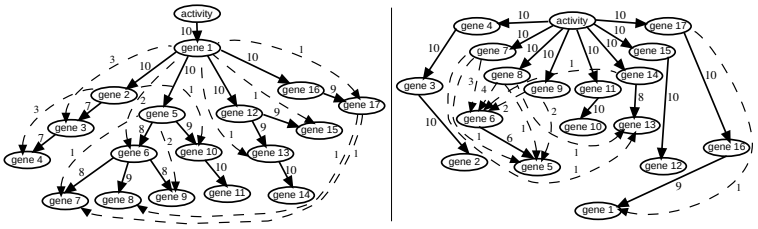

Figure 6. Cascade and convergence recovery. Recovery of the cascade structure is shown on the left; the convergence on the right. Numbers next to links represent number of data sets in which that link was found, as described in Fig. 4. Compare with the true structures in Fig. 2d and 2e.

3.2 Influence of sampling regime

Fig. 7 shows recovery of the genetic network topology of Fig. 2a under different sampling intervals. Recovery success deteriorates with sampling intervals greater than 5, and gets worse as the interval increases beyond 10, the links jumping to genes further and further downstream. Interval 1 shows little correct recovery; however, 2, 3, and 4 perform similarly to 5. Quantitatively, we calculated (1) the percentage of links of the true structure recovered by the algorithm, called the "percent recovered", and (2) the percentage of links in the recovered structure that correspond to correct links of the true structure, called the "percent correct". We then compared these values from each sampling interval to the corresponding values for a sample interval of 5, using a one-tailed student's t-test, assuming that the interval 5 represented the best recovery possible.

Intervals 1 and 2 were not significantly different from interval 5 on percent correct (t_{58}=1.17, P=0.12 for both), but performed significantly worse on percent recovered (t_{58}≥16.77, P<0.0001). That is, the links they recovered were correct, but they did not recover as many links. Intervals 3 and 4 were not significantly different from 5 on either percent recovered or percent correct (t_{58}≤1.27, P≥0.1). All intervals 6 and greater had significantly lower percent recovered and percent correct than interval 5 (t_{58}≥5.04, P<0.001) (Fig. 7).

With the two shortest intervals, 1 and 2, the behavior did not last long enough for influences to propagate throughout the entire pathway. A failure to infer the structure in this situation could be due to the lack of dynamics in the system, and not lack of ability of the algorithm to find the structure. When tested in a situation where the behavior persisted throughout the sampling period and influences were able to propagate through the entire pathway, interval 2 performed similarly to 5 on both measures (t_{58}≤0.58, P≥0.3); however, interval 1 recovered significantly worse than 5 on both measures (t_{58}≥13.02, P<0.001) (Fig. 7). For interval 1, the sampling period lasted only 21 time steps, and the latter elements in the pathway did not receive their regulation until 30 or 40 time steps after a behavioral switch, thus it is possible that it simply still did not observe enough of the network dynamics. In order to determine whether an interval of 1 could recover the network given a long enough sampling time, we created a single data set sampling 61 data points over two behavioral switches. In this situation, all the links in the original network, except the link from gene 3 to gene 6 not recovered in the other tests, were found (not shown).

4 Discussion

We found that our inference algorithm, NETWORKINFERENCE, is successful at recovering both multiple targets of a regulator and regulatory loops. Our algorithm

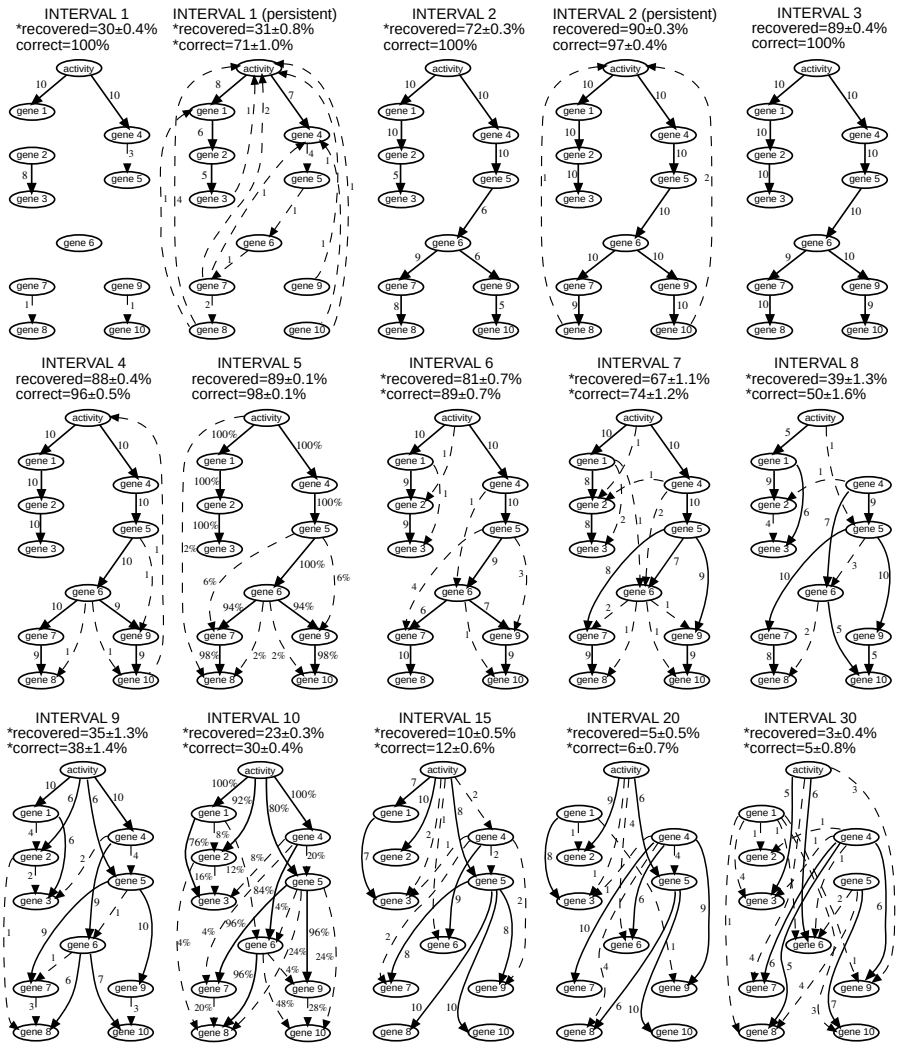

Figure 7. Sampling recovery. Recovery for various sampling intervals from 1 to 30. Numbers next to links represent number of data sets in which that link was found, as described in Fig. 4. Graphs with intervals 5 and 10 are from [2], and show percentages instead of number of trials. Intervals 1 and 2 (persistent) represent the situation when the behavior persisted, in order to allow influence to propagate through the entire pathway. Mean±SE of percent recovered and correct are shown for each sampling type. Asterisks preceding the percent recovered and percent correct indicate that the value is significantly different from those of interval 5. Compare with the true structure in Fig. 2a.

repeatedly proved its ability to find multiple targets of a regulator from different topologies, in our 10 gene network, in the gene cascade, and in the convergence network where it found all eight children of the activity node. Feedback regulatory loops are a prominent feature in many biological systems, thus it is encouraging that our algorithm correctly recovered the loop in 100% of the trials tested. It is often the impression that Bayesian network algorithms should have difficulty discovering loops in a structure, due to the acyclic nature of Bayesian networks. However, use of a dynamic Bayesian network negated this problem.

NETWORKINFERENCE had difficulty, however, recovering multiple regulators of a target gene in both the simple two-parent cases and the more complex convergence network. Because Bayesian networks are designed to guard against over-fitting, they naturally prefer simpler model structures to more complex ones. In the absence of large amounts of data suggesting inclusion of multiple parents, models with only one regulator as a parent often score better than models with both. As the amount of data increases to support the inclusion of additional parents, however, these edges appear. The amount of data needed varies, and is partially dependent on the degree to which two properties are satisfied: Bayesian network recovery of multiple parents works best when first, the value of the child depends on the value of each parent independently, that is, the child's value would be different were one or the other of the parents not present; and second, many possible combinations of parent states are observed [10].

In the case of correlated influence of multiple parents, these properties do not hold. In addition, even in the case of independent influence, the regulated gene was usually pushed in our simulations to its maximum or minimum expression level, where regulation by either parent would be more than sufficient to keep it at its extremum. As a result, only when the regulated gene was increasing or decreasing could the influences of both parents be detected. Given that these properties fail to hold in our simulations, it is to be expected that larger amounts of data would be required to fully recover all links in these networks.

In Yu et al. [14], we examine the influence of the amount of data on network recovery and demonstrate that more data enables a Bayesian network inference algorithm to recover networks with larger numbers of parents. To test whether NETWORKINFERENCE was capable of identifying multiple parents given larger amounts of data in this context, we ran BRAINSIM with uncorrelated regulators (Fig. 2b) of independent influence weight 0.1 through 20 behavioral transitions, resulting in 401 sampled data points per bird region and thus 12,000 observations of the 202 variables, unrealistic experimentally. NETWORKINFERENCE successfully recovered all links in the underlying structure, including gene 3 and gene 5 as joint parents of gene 6. We have not yet determined the minimum amount of data required for successful recovery, and it will be interesting to see if the required amount is close to the realm of experimental feasibility.

Our simulation framework enabled us to test in fine detail another important experimental consideration, sampling regime. It is clear that sampling with an interval too long for the system's dynamics produces poor results. Even the sampling interval of 6, whose recovery graph did not visually look much worse than the intuitive choice of 5 (Fig. 7), performed significantly more poorly than the shorter intervals in terms of both proportion recovered and proportion correct. It is encouraging that the intervals of 2, 3, and 4 all performed as well as interval 5. Interval 1 only observed enough of the network dynamics to recover the entire system when the period of data sampling was lengthened. In Fig. 7, it can be seen how those links most often recovered with the shorter sampling period were the first links in the pathway. Thus an interval of 1 is not too short to recover the dynamics of the system; however, it is too short to observe the entire system with a biologically realistic number of data samples.

Our results suggest a strategy for finding the correct sampling regime for a biological system by slowly increasing the interval. With increasing sampling interval, first more distal elements of the pathway are recovered as the samples expand to cover the dynamics of the entire system. Then there is a plateau when the recovered graph does not change. When the sampling interval exceeds the internal dynamics of the system, recovery begins to skip nodes in the graph with more and more regularity, and links genes to further and further downstream elements. Thus an interval somewhere on the plateau should be chosen.

In summary, our network inference evaluation approach is showing us the promises and limitation of this algorithm, and thus, areas that need improvement. Our approach also is informative on how to sample data from real biological systems for use with network inference algorithms.

Reference

1. P. D'Haeseleer, S. Liang, and R. Somogyi, *Bioinformatics* **16**, 707 (2000).
2. V.A. Smith, E.D. Jarvis, and A.J. Hartemink, *Bioinformatics* **18**, S216 (2002).
3. E.D. Jarvis, et al., *J Comp Physiol A* in press (2002).
4. D.E. Zak, et al., *ICSB* **2**, 231 (2001).
5. E.D. Jarvis, et al., *Nature* **406**, 628 (2000).
6. E.A. Brenowitz, D. Margoliash, and K.W. Nordeen, *J Neurobiol* **33**, 495 (1997).
7. N.A. Hessler and A.J. Doupe, *J Neurosci* **19**, 10461 (1999).
8. H.H. McAdams and A. Arkin, *PNAS* **94**, 814 (1997).
9. A.J. Hartemink, et al., *PSB* **7**, 437 (2002).
10. D. Heckerman, D. Geiger, and D.M. Chickering, *Mach Learn* **20**, 197 (1995).
11. N. Friedman, K. Murphy, and S. Russell, *UAI* **14**, 139 (1998).
12. A. West, et al., *Proc Natl Acad Sci U S A* **98**, 11024 (2001).
13. K. Miyazono, K. Kusanagi, and H. Inoue, *J Cell Physiol* **187**, 265 (2001).
14. J. Yu, et al., *Genome Research* (submitted).