Just: Data requirements
Data requirements of reverse-engineering algorithms

Winfried Just
Department of Mathematics
Ohio University
Athens, OH 45701

Phone: (740)-593-1260
Fax: (740)-593-9805
just@math.ohiou.edu

April 30, 2007

Keywords: reverse-engineering, biochemical networks, data requirements, prior, No Free Lunch Theorem, Gröbner basis, term order

Abstract: Data sets used in reverse-engineering of biochemical networks contain usually relatively few high-dimensional data points, which makes the problem in general vastly underdetermined. It is therefore important to estimate the probability that a given algorithm will return a model of acceptable quality when run on a data set of small size but high dimension. We propose a mathematical framework for investigating such questions. We then demonstrate that without assuming any prior biological knowledge, in general no theoretical distinction between the performance of different algorithms can be made. We also give an example of how expected algorithm performance can in principle be altered by utilizing certain features of the data collection protocol. We conclude with some examples of theorems that were proven within the proposed framework.

High-throughput methods make it currently feasible to simultaneously collect data on all chemicals in large biochemical networks, such as gene regulatory networks, metabolic networks, or signal-transduction networks. Still, data collection remains expensive, and reverse-engineering algorithms typically rely on small data sets. Thus reverse-engineering problems are typically vastly underdetermined in the sense that a huge number of network models are consistent with the available data. Reverse-engineering algorithms usually deal with this problem by selecting and returning one model that is consistent with the data.

In view of the above, it will be extremely useful to develop a theory of data requirements for reverse-engineering algorithms. Ideally, such a theory would be able to predict the probability that a given algorithm returns the correct, or an approximately correct, model of the network from a given data set. Moreover, if an algorithm uses input parameters, the theory should provide some guidelines for the most promising choice of input parameters.

Here we outline some general issues that arise in developing such a theory for dynamical systems models of biochemical networks. The paper is organized as follows: In Section 1, we set up a framework for turning questions of data requirements for a given algorithm into precise mathematical problems. In Section 2, we discuss the role of prior biological knowledge about networks and of the data collection procedure in assessing the expected performance of reverse-engineering algorithms. In particular, we prove a simple theorem that shows that in the absence of any such prior knowledge, it will in general be impossible to prove superior performance of one algorithm relative to another algorithm. We also show by means of a simple example that knowledge of the data collection protocol alone can in principle be used to design algorithms with higher expected performance. In Section 3, we review some previously published theorems of the author on data requirements. Section 4 contains a summary and discussion.

1 Expected data requirements as a mathematical problem

The purpose of reverse-engineering is to build a model of a biochemical network based on experimental data and prior biological knowledge. A reverse-engineering algorithm takes as input a set of data and outputs a model of the network. We are interested in estimating the probability that a given reverse-engineering algorithm returns a network model of acceptable quality when run on a data set of a given size, or, alternatively, in estimating the expected size of the data set that would be needed for the algorithm to return a network model of acceptable quality.

In this section we will discuss how the loosely formulated preceding paragraph can be converted into rigorous mathematical problems. First we will need to give precise meaning to the terminology.

Models

Biochemical networks can be studied in a variety of mathematical frameworks [1], [2]. At the most fine-grained level, chemical reactions are stochastic events that happen between individual molecules, and therefore one can treat the whole network as a stochastic process whose state is a vector of the numbers of molecules of each species in the network. We will refer to this approach as the *single-molecule paradigm.*

At a level of intermediate resolution, one can treat the state of the network as a vector of concentrations of individual chemical species and model the network dynamics as a continuous flow governed by a system of ODE's (if spatial homogeneity is assumed) or PDE's (if concentration is allowed to vary according to location). We will refer to these approaches as the *continuous paradigm.*

On the coarsest level of resolution, one can discretize concentrations of individual species into a small number of discrete classes (*e.g., high, medium, low*) and consider the state of the network as

a vector of these discretized concentrations. This approach conceptualizes biochemical networks as discrete dynamical systems, and we will call it the *DDS paradigm.*

Each of the paradigms considered so far treats the biochemical network as a (stochastic or deterministic) *dynamical system.* Systems biology [3], [4], [5] aims at understanding how the individual components of networks (molecules, chemical species, individual reactions) generate emergent properties at higher levels of biological organization. Since these properties are likely to result from the network dynamics, it appears that in order to fulfil the promise of systems biology, we will ultimately need reliable models that fall within one of the above paradigms. For this reason, I will focus in most of this paper on dynamical systems models.

However, a lot of insight about a biochemical network can already be obtained from knowing its connectivity, which can be modeled as a (directed or indirected) graph that shows which species in the network interact, a signed directed graph (that also shows whether an influence is positive or negative), or a Bayesian network. We will refer to these approaches collectively as the *connectivity paradigm.*

Data

Typically, the data will tell us about concentration levels at various times and under various conditions. For modeling within the single-molecule paradigm we would need data on numbers of molecules of a given chemical species, for modeling within the continuous paradigm we would need reasonably precise measurements of the concentration levels of all species in the network; for modeling within the DDS or connectivity paradigms, we can use data that tell us about rough concentration levels of all species at various times or their change (*e.g.,* "upregulated," "downregulated") between different times or between different experimental conditions. The latter type of data (*e.g.,* microarray data [6]) are currently being collected on a large scale for a variety of networks. However, large-scale data sets of sufficiently precise concentration measurements for the continuous paradigm may eventually become available, and the single-molecule paradigm has been successfully used for some small networks [7]. Much of what will be said below does in principle apply to all three paradigms for building dynamical systems models of biochemical networks.

The data in a given data set need not necessarily be all of the same kind; for example, once could use both data from microarray and Chip-on-Chip experiments [8] simultaneously to reverse-engineer a given network. If all data are of the same kind, then the *size* of the data set is the number of experiments performed; if the data come from different kinds of experiments, the notion of size would be multidimensional. We want to emphasize that our notion of size (*e.g.,* the number microarray experiments) is different from that of the *dimension* (*e.g.,* the number of probes on a single microarray) of the data set.

For obvious reasons, not much is known about the distribution of data sets of a given size that are likely to be obtained from performing experiments on a given network. We will need to assume that the data are randomly drawn from a distribution Λ that reflects this lack of knowledge.

Prior biological knowledge

The reverse-engineering process may (and should) also take into account prior biological knowledge about the network under consideration. When studying gene regulatory networks, for example, an algorithm may take into account that the transcription of each is likely to be regulated by the products of relatively few other genes [9], that the gene regulatory functions tend to be predominantly *nested canalyzing functions* [10], [11], that the set of such potential regulators comprises only a fraction of all network nodes, and that some genes are already known to code for transcription factors and are thus more likely candidates for coding regulators of a given gene than other genes

that for example are known to code for certain enzymes. In practice, this can be done via human intervention, for example, by tuning certain input parameters or rejecting biologically implausible solutions.

For the purpose of formulating rigorous mathematical problems, we will assume throughout this paper that the process of incorporating prior biological knowledge is part of the algorithm itself. Thus, in the case of human intervention, part of the algorithm would be run on a computer and part by the modeler's brain, and if a formally defined prior is part of the input, then we would consider two runs of the same algorithms on different priors as running two different algorithms.

Quality of the solution

It will in general depend on the modeling paradigm how we measure the quality of a solution. If only the network connectivity is to be modeled, then measures based on the percentage of of true/false positive/negatives in predicted arcs of the network are appropriate. In the DDS paradigm, we may be interested in the percentage of correctly identified individual regulatory functions. In this case our criterion for individual regulatory functions would be *perfectionist,* in the sense that we don't accept partially correct individual regulatory functions. In the continuous paradigm, one would typically want differential equations for individual biochemicals in the network that have the correct general form, but would allow the coefficients to differ from the correct ones to a certain extent. Similar considerations apply to the single-molecule paradigm. Finally, for dynamical systems models, we might base quality measures on the dynamics of the derived models. For example, we might be satisfied with a solution that correctly predicts the steady states and limit cycles of the model.

In general, quality measures can take the form of an acceptability threshold or may express the quality on a real-valued scale.

The data requirement problem

Now let us assume that we are given a reverse-enginnering algorithm A that returns models M^* of the network from a certain set of possible models $\mathcal{M}$. Depending on the modeling paradigm, this set would either consist of possible dynamical systems or possible network connectivities. Let us further assume a probability distrubution Θ on the set of all these possible models that reflects our prior biological knowledge. Let us moreover assume that a model M is randomly drawn from Θ, and that a data set D is randomly generated from a network modeled by M, according to a probability distribution Λ that reflects, to the extent possible, our knowledge about how data are being collected in the lab.

If we are given a quality measure for the model M that takes the form of an acceptability threshold, we can then consider the random event Q that will occur whenever M^* is an acceptable approximation of M. In this case, we can ask the following mathematical question:

Question 1 *Given $\mathcal{M}, \Theta, \Lambda, D, A$, estimate $Pr(Q)$.*

If the quality measure is a scalar, then the model quality becomes a random variable on ξ on Λ, and we can ask:

Question 2 *Given $\mathcal{M}, \Theta, \Lambda, D, A$, estimate $E(\xi)$.*

If the quality measure takes the form of a threshold T, and the size of a data set can be expressed on a linear scale, then we can consider sampling an infinite sequence of data points from Λ, running the algorithm A on each initial segment of this sequence and define a a random variable λ as the minimum size of the data set for which A returns a model M of aceptable quality. This leads to the following:

Question 3 *Given $\mathcal{M}, \Theta, \Lambda, A, T$, estimate $E(\lambda)$.*

2 The roles of Θ and Λ

In this section we will illustrate that no meaningful answers to questions 1–3 can be expected without meaningful priors Θ. We will also illustrate how Λ can influence the answers.

We will work within the following version of the DDS-paradigm. The network consists of n chemical species. Species concentrations x_i can be suitably discretized to elements of some finite set F. Concentration levels change simultaneously in well-defined time steps t so that for all $i \in [n]$ and all t we have $x_i(t+1) = h_i(\bar{x}(t))$. In our notation, $[n] = \{1, 2, \ldots, n\}$. The (unknown) function h_i will be called the *i-th regulatory function.* We will also assume that there is no noise in the data in the sense that every measurement of the response $\bar{y}$ of a system to input vector $\bar{x}(t)$ at time step $t+1$ will satisfy $y_i = h_i(\bar{x}(t))$ for all $i \in [n]$. An investigation of when these assumptions are sufficiently close to the truth is beyond the scope of this paper. Our assumptions imply that we can treat the whole network as a discrete dynamical system $< F^n, H >$, where $H = [h_1, \ldots, h_n]$ and $\bar{x}(t+1) = H(\bar{x}(t)) = [h_1(\bar{x}(t)), \ldots, h_n(\bar{x}(t))]$ for all time points t.

A reverse engineering algorithm A is assumed to take as input a data set $D = \{< \bar{x}(t), \bar{y}(t) >: t \in [m]\}$, where m is the number of data points, and to return a function $H^* = [h_1^*, \ldots, h_n^*]$ such that $\bar{y}(t) = H^*(\bar{x}(t)) = [h_1^*(\bar{x}(t)), \ldots, h_n^*(\bar{x}(t))]$ for all $t \in [m]$. In time series data, it will be the case that $\bar{y}(t) = \bar{x}(t+1)$, but we do not make this assumption in general. A component h_i^* of H^* will be called a *model for the i-th regulatory function.*

Our quality criterion will be perfectionist in the sense that we accept a solution only if $H = H^*$ and reject it otherwise.

Note that the space of all models $\mathcal{H}$ in this framework has $|F|^{n|F|^n}$ elements, and for a data set D of size m there are $|F|^{|F|^n(|F|^n - m)}$ different potential solutions H^* for the problem of reverse-engineering H^* from D.

Now let us assume that we have no prior biological knowledge about the network. This would translate into a uniform prior distribution Θ_u on $\mathcal{H}$. Moreover let us assume that set of data inputs $C = \{\bar{x}(t) : t \in [m]\}$ is randomly drawn from an arbitrary distribution Λ, and the data set $D = \{< \bar{x}(t), \bar{y}(t) >: t \in [m]\}$ itself is obtained by measuring the system response to these data inputs. It is important here that all data inputs are drawn before any measurements were taken; this would most definitely not apply if the data come from a time series.

Theorem 4 *Suppose H is randomly drawn from $< \mathcal{H}, \Theta_u >$, the set $C = \{\bar{x}(t) : t \in [m]\}$ of data inputs from an arbitrary distribution Λ, and a reverse algorithm A is run on the resulting data set $D = \{< \bar{x}(t), \bar{y}(t) >: t \in [m]\}$ of size m that was obtained by measuring the system response $\bar{y}(t) = H(\bar{x}(t))$ at all inputs. Let H^* be the output of algorithm A. Then $Pr((H^* = H)|D) = |F|^{n(m-|F|^n)}$.*

Proof: Let $Pr(C)$ denote the probability that the given set of data inputs was chosen. Since C was picked first (before any measurements were taken), we can treat C as a random variable that is independent of H. Moreover, since the algorithm A is presumed to return only solutions that are consisistent with the data, we have $Pr(D|H^*) = Pr(C)$. Now we get from Bayes' Formula:

$$Pr((H^* = H)|D) = \frac{Pr(D|H^*) \cdot Pr(H^* = H)}{Pr(D)} = \frac{Pr(C) \cdot |F|^{-n|F|^n}}{Pr(C) \cdot |F|^{-nm}} = |F|^{n(m-|F|^n)}. \quad (1)$$

□

We call this simple observation a "Theorem" only because it is similar to the "No Free Lunch Theorem" of [12]. It says that with a uniform prior and any dependency of data inputs on the dynamics of the system, no reverse-engeneering algorithms performs better than any other, in particular, none performs better than random guessing. Thus meaningful estimates of data requirements must use nonuniform priors; regardless of whether or not the algorithm A itself explicitly uses priors.

It is however possible to improve algorithm performance if the data inputs depend to some extent on the dynamics of the system, even if we don't incorporate knowledge into our prior Θ that could be called "biological knowledge." Let us illustrate this with a toy example of a data set that has just one data point. Suppose the experimentalist tells us: "At time $t = 0$, I set up my experiment. Then I let the network run undisturbed until time t, at which time I collect the first measurement $\bar{x}(t)$, and I measure the system response $\bar{y}(t)$. I am asking you to reverse engineer H from the data set $D = \{< \bar{x}(t), \bar{y}(t) >\}$."

How could this information help? Suppose $H \in \mathcal{H}$. For each transient state $\bar{x}$ we define recursively a depth $d_H(\bar{x})$ as follows: If there is no $\bar{z}$ with $H(\bar{z}) = \bar{x}$, then $d_H(\bar{x}) = 0$. Otherwise, $d_H(\bar{x}) = \max\{d_H(\bar{z}) + 1 : H(\bar{z}) = \bar{x}\}$. For persistent states $\bar{x}$ we define $d_H(\bar{x}) = \infty$.

Now some potential solutions H^* of the reverse engineering problem from D are inconsistent with the data collection procedure, since this procedure implies that $d_H(\bar{x}(1)) \geq t$. Thus only solutions H^* with

$$d_{H^*}(\bar{x}(1)) \geq t \tag{2}$$

are consistent both with the data and with the data collection procedure. Our goal is to find a solution H^* that satisfies inequality (2). We can use any given reverse-engineering algorithm A to find a solution H^* that satisfies inequality (2); all it takes is not to fabricate some data. Here is how this works:

1. Randomly pick pairwise different $\bar{w}(0), \ldots, \bar{w}(t-1) \in F^n \backslash \{\bar{x}(t), \bar{y}(t)\}$. Let $\bar{w}(t) = \bar{x}(t)$.
2. Let $D^+ = D \cup \{< \bar{w}(r-1), \bar{w}(r) >: r \in [t]\}$.
3. Run A on D^+.

Clearly, the solution H^* returned by this algorithm is also a solution of the reverse engineering problem for D, and it satisfies (2).

Note that in this algorithm, selection of the data inputs is no longer independent of H. By how much does this algorithm improve the success probability relative to randomly guessing a model?

Let d be a nonnegative integer, and let $\bar{x} \in F^n$. We define $b_d(\bar{x})$ as the probability that $d_H(\bar{x}) \leq d$, where H is randomly drawn from $< \mathcal{H}, \Theta_u >$. Since the distribution of the H's is uniform, it does not discriminate between different $\bar{x}$'s, and hence $b_d(\bar{x})$ does not depend on $\bar{x}$. We will write b_d instead of $b_d(\bar{x})$.

For $\bar{x} \in F^n$, let $\rho_{\bar{x}}(H) = |\{\bar{z} \in F^n : H(\bar{z}) = \bar{x}\}|$. Then $\rho_{\bar{x}}$ is a random variable on $< \mathcal{H}, \Theta_u >$ whose distribution can be approximated by a Poisson distribution with parameter $\lambda = 1$.

Now we are ready to estimate the probabilities b_d.

For $d = 0$, we note that $d_H(\bar{x}) \leq 0$ iff $\rho_{\bar{x}} = 0$. It follows that $b_0 = \frac{1}{e}$.

For $d > 1$, note that

$$b_d(\bar{x}) = Pr(\forall \bar{z}\ (H(\bar{z}) = \bar{x} \rightarrow d_H(\bar{z}) \leq d-1)). \tag{3}$$

Unfortunately, for pairwise different $\bar{z}_1, \ldots, \bar{z}_k$, the events $(d_H(\bar{z}_1) \leq d-1), \ldots, (d_H(\bar{z}_k) \leq d-1)$ are not independent; neither can any of these events be presumed independent of the value of $\rho_{\bar{x}}(H)$.

However, if $\rho_{\bar{x}}(H)$ and k are small, then the assumption of independence will be violated only to an insignificant degree. Thus we can approximate the right-hand side of equation (3) as follows:

$$\begin{aligned} b_d(\bar{x}) &= Pr(\forall \bar{z}\ (H(\bar{z}) = \bar{x} \rightarrow d_H(\bar{z}) \leq d-1)) \\ &= \sum_{r=0}^{\infty} Pr(\rho_{\bar{x}} = r) \cdot Pr(\forall \bar{z}\ (H(\bar{z}) = \bar{x} \rightarrow d_H(\bar{z}) \leq d-1) | \rho_{\bar{x}} = r) \\ &\approx \sum_{r=0}^{\infty} e^{-1} \frac{(b_{d-1})^r}{r!} = e^{b_{d-1}-1}. \end{aligned} \tag{4}$$

We have proved the following approximate recursion:

$$\begin{aligned} b_0 &= \frac{1}{e} \\ b_{d+1} &\approx e^{b_d - 1} \end{aligned} \tag{5}$$

For example, if the improvement the system had been running for two time steps before the first measurement was taken, then the the success probability of our data fabrication algorithm would improve by a factor of about $\frac{1}{b_1} \approx \frac{1}{0.5315}$ relative to (1). Admittedly, we are talking about a two-fold improvement of a ridiculously low probability here, but this calculation is only meant as an illustration that taking into account the interplay between data collection procedures and the dynamics of the system can improve prediction accuracy. If measurements are taken from systems that have been running unperturbed for a long time, then data can be presumed to be collected from an attractor, which may lead to more substantial improvements.

3 Examples of theorems on data requirements

In this section we review some results of the author along the lines of our questions 1–3 that have been published in [13].

In [14], Laubenbacher and Stigler developed a reverse-engineering algorithm for the case when the set F of discretized concentration levels is a finite field, for example, the field $\{0, 1\}$ of Boolean values. This algorithm will henceforth be called the *LS-algorithm.* If F is a finite field, then each regulatory function h_i is a polynomial in $F[x_1, \ldots, x_n]$, and if h_i^* is any model for a given data set $D = \{< \bar{x}(t), \bar{y}(t) >:\ t \in [m]\}$, then the set of all models of the i-th regulatory function for D can be concisely described as $h_i^* + I_D$, where $I_D = \{h \in F[x_1, \ldots, x_n] : \forall t \in [m]\ h(\bar{x}(t)) = 0\}$ is the ideal of all polynomials that vanish on all data inputs [14].

A *term order* is a well-order $\prec$ of the monomials in $F[x_1, \ldots, x_n]$ such that $x^\alpha \preceq x^\beta$ implies $x^\alpha x^\gamma \preceq x^\beta x^\gamma$, where α, β, γ are multiexponents. Prominent examples are *lex(icographical)* orders generated by a variable order $x_{\pi(1)} \prec x_{\pi(2)} \prec \cdots \prec x_{\pi(n)}$ and *graded* term orders in which $\sum \alpha < \sum \beta$ always implies $x^\alpha \prec x^\beta$. Note that for the variable order given by the identity permutation, $x_1 x_2 \prec_{lex} x_3$, whereas $x_3 \prec x_1 x_2$ for any graded term order $\prec$.

For any ideal $I \in F[x_1, \ldots, x_n]$ and term order $\prec$, there exists a basis $G_\prec$ for I called a *(reduced) Gröbner basis with respect to* $\prec$. This basis has the property that for every $f \in F[x_1, \ldots, x_n]$ there exists a unique polynomial $f\%G_\prec$, called the *normal form of* f *with respect to* $G_\prec$, such that $f - f\%G_\prec \in I$ and no term of $f\%G_\prec$ is divisible by the leading term of any polynomial in $G_\prec$.

The LS-algorithm takes as input a data set D and a term order $\prec$, and outputs a model h_i^* of the i-th regulatory function for D. It finds h_i^* by first constructing one model h^* of h_i and then

computing and returning the normal form $h^*\%G_\prec$, where $G_\prec$ is the reduced Gröbner basis for I_D with respect to $\prec$. The model returned by the algorithm is the *most parsimonious* model of h_i for D in a sense that depends on the choice of $\prec$.

When running the LS-algorithm on a given data set, one faces the problem of choosing a suitable term order as input parameter. One goal of analyzing the data requirements for the algorithm was to develop some guidelines for a suitable choice of term order. In a version of the LS-algorithm subsequently developed in [15], the problem is partially solved by using preprocessing to find a suitable variable order $x_{\pi(1)} \prec x_{\pi(2)} \prec \cdots \prec x_{\pi(n)}$, and then running the LS-algorithm with a term order $\prec$ that is consistent with this variable order. The latter version will be referred to as the *LS-algorithm with preprocessing.*

An analysis of data requirements for both versions of the LS-algorithm under the assumption of random data input vectors was performed in [13]. It uses the assumptions that the number of inputs to a given regulatory function is bounded (which corresponds to a prior Θ that assigns zero probability to models where this assumption is violated, but is uniform otherwise), a perfectionist quality criterion for individual regulatory functions, and that a potentially infinite number of data inputs is independently drawn with replacement from a uniform distribution. The following theorems summarize the main results.

Theorem 5 *Let $h_i = a_1 x^{\alpha_1} + \cdots + a_\ell x^{\alpha_\ell}$ be such that $\max\{\sum \alpha_w : w \in [\ell]\} = k$ and the maximum number of variables that occur in any monomial of h_i is j. Then the expected number of data points needed before the LS-algorithm that is run with a* randomly chosen graded *term order returns $h_i^* = h_i$ is on the order of at least $\Omega(n^j)$ and at most $O(|F|^k n^k)$. The expected number of data points needed before the LS-algorithm that is run with an* optimally chosen *graded term order returns h is on the order of at least $\Omega(n^{k-1})$.*

Theorem 6 *Suppose h_i depends on at most k variables. The expected number of data points needed before the LS-algorithm with an* optimally chosen lex *order returns $h_i^* = h_i$ is on the order of $O(|F|^k k \ln |F|)$.*

Theorem 7 *Suppose h_i depends on at least $k > 0$ variables. The expected number of data points needed before the LS-algorithm with a* randomly chosen lex *order returns $h_i^* = h_i$ is on the order of $\Omega(c^n)$ for some constant $c > 1$. Moreover, for any fixed positive q, the expected number of data points needed for the LS-algorithm to return $h_i^* = h_i$ with probability $> q$ grows exponentially in n.*

Theorem 8 *Suppose h_i depends on at most k variables. Then the expected number of data points needed before the LS-algorithm with preprocessing, run with the corresponding lex order, returns $h_i^* = h_i$ is on the order of $O(|F|^{2k}\, 2k \ln n)$. Moreover, if h_i is nested canalyzing, then this expected number is on the order of $O(|F|^{k+1}\,(k+1)\ln n)$.*

Together, Theorems 5–8 show the importance of appropriate choice of the input parameter (the term order) when running the LS-algorithm, and they allow us to identify situations when the data set is simply too small to expect the algorithm to output correct models. Moreover, they give some indications that for very small data sets working with lex orders after preprocessing would be best, but if no preprocessing is done and the data set is relatively large, using graded term orders is more promising. This type of conclusions may be of practical importance for users of the algorithm.

The bound in Theorem 8 is similar to bounds obtained in [16], [17], and [18] for other reverse-engineering algorithms that operate within the DDS paradigm or connectionist paradigm. Moreover, similar scaling as in Theorem 8 was reported in [19] and [20] for reverse-engineering algorithms that operate under the continuous paradigm, when the system is modeled by piecewise linear differential

equations. The authors of [19] and [20] also infer a specific recommendation for experimentalists from their results, namely that contrary to the common practice of making observations at evenly spaced time intervals, at least some expression samples should be packaged closely in time so that the signs of derivatives can be directly observed.

4 Summary and discussion

We outlined a mathematical framework for estimating data requirements of reverse-enginnering algorithms. Theorem 4 shows that meaningful estimates can only be obtained in view of appropriate priors. We also illustrated that algorithm performance can be enhanced by taking into account the process of data collection. Therefore, this protocol also should be taken into account when expected performance of the algorithms is estimated.

The results that we reviewed in the previous section show that meaningful and mathematically rigorous answers to Questions 1–3 can sometimes be given, and that such answers can sometimes guide our choice whether and with which input parameters to use a given algorithm. Moreover, development of improved algorithms requires evaluation of their performance. This can be done empirically to some extent, but theorems about expected performance are likely to give us additional insights into when an algorithm works well and when it doesn't, which in turn may suggest ways of developing better algorithms.

Our results in Section 2 underscore the importance of priors Θ and the distributions Λ of data sets. One key to progress in both algorithm development and performance evaluation appears to lie in discovering priors that more closely reflect our ever advancing biological knowledge. A word of caution about priors is needed: If an algorithm relies too heavily on a prior Θ, then it will miss correct network models with properties that go against conventional assumptions. This type of overfitting should be more easily avoidable if we use algorithms that rely on explicitly formulated priors, or if we understand for which kind of priors an algorithm has high expected performance. A second key to progress is likely to come from understanding of how the data sets obtained from actual experiments are distributed. The should be of use both for algorithm development and for performance evaluation; it could also offer guidelines to experimentalists on how to collect data whose distribution maximizes the success probability of reverse-engineering algorithms.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Agreement No. 0112050 while the author was a visitor at the Mathematical Biosciences Institute.

References

[1] D'haseleer, P., Liang, S., and Somogyi, R. (2000) Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics* **16**(8), 707–726.

[2] De Jong, H. (2002) Modeling and Simulation of Genetic Regulatory Systems: A Literature Review. *Journal of Computational Biology* **9**(1), 67–103.

[3] Kitano, H. (2001) *Foundations of Systems Biology.* MIT Press.

[4] Palsson, B. O. (2006) *Systems Biology: Properties of Reconstructed Networks.* Cambridge U Press.

[5] Alon, U. (2006) *An Introduction to Systems Biology.* Chapman and Hall.

[6] Schena M., Shalon D., Davis R. W., and Brown, P. O. (1995) Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science* **270**, 467–470.

[7] McAdams, H. M. and Arkin, A. (1997) Stochastic mechanisms in gene expression. *PNAS* **94**, 814–819.

[8] Horak, C. E. and Snyder, M. (2002) ChIP-chip: a genomic approach for identifying transcription factor binding sites. *Methods in Enzymology* **350**, 469–483.

[9] Arnone, M. I., and Davidson, E. H. (1997) The hardwiring of development: Organization and function of genomic regulatory systems. *Development* **124**, 1851–1864.

[10] Harris, S. E., Sawhill, B. K., Wuensche, A. and Kauffman, S. A. (2002) A model of transcriptional regulatory networks based on biases in the observed regulation rules. *Complexity* **7**(4), 23–40.

[11] Kauffman, S., Peterson, C., Samuelsson, B., and Troein, C. (2003) Random Boolean network models and the yeast transcriptional network. *PNAS* **100**(25), 14796–14799.

[12] Wolpert, D. H. and Macready, W. G. (1997) No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation* **1**, 67–82.

[13] Just, W. (2006) Reverse engineering discrete dynamical systems from data sets with random input vectors. *Journal of Computational Biology* **13**(8) 1435–1456.

[14] Laubenbacher, R. and Stigler, B. (2004) A computational algebra approach to reverse engineering of gene regulatory networks. *Journal of Theoretical Biology* **229**, 523–537.

[15] Jarrah, A., Laubenbacher, R., Stigler, B. and Stillman, M. (2006) Reverse-engineering of polynomial dynamical systems. *In press.*

[16] Akutsu, T., Miyano, S., and Kuhara, S. (1999) Identification of genetic networks from a small number of gene expression patterns under the boolean network model. *Pacific Symposium on Biocomputing* **4**, 17–28.

[17] Krupa, B. (2002) On the Number of Experiments required to Find the Causal Structure of Complex Systems. *Journal of Theoretical Biology* **219**, 257–267.

[18] Perkins, T. J. and Hallett, M. (2005) A trade-off between sample complexity and computational complexity in learning Boolean networks from time series data. *Preprint.*

[19] Perkins, T. J., Hallett, M., and Glass, L. (2004) Inferring models of gene expression dynamics. *Journal of Theoretical Biology* **230**, 289-299.

[20] Perkins, T. J, Hallett, M., and Glass, L. (2006) Dynamical properties of model gene networks and implications for the inverse problem. *BioSystems* **84**, 115-123.