Reverse-Engineering of Dynamic Networks

B. Stigler[1], A. Jarrah[2], M. Stillman[3], and R. Laubenbacher[2]

[1] Mathematical Biosciences Institute, The Ohio State University, Columbus, OH USA
[2] Virginia Bioinformatics Institute, Virginia Tech, Blacksburg, VA USA
[3] Mathematics Department, Cornell University, Ithaca, NY USA

bstigler@mbi.osu.edu, {ajarrah, reinhard}@vbi.vt.edu, mike@math.cornell.edu

Abstract

We consider the problem of reverse-engineering dynamic models of biochemical networks from experimental data using polynomial dynamical systems. In earlier work, we developed an algorithm to identify minimal *wiring diagrams*, that is, directed graphs that represent the causal relationships between network variables. Here we extend this algorithm to identify a most likely dynamic model from the set of all possible dynamic models that fit the data over a fixed wiring diagram. To illustrate its performance, the method is applied to simulated time-course data from a published gene regulatory network in the fruitfly *Drosophila melanogaster*.

Key words: Gene regulatory networks, reverse-engineering, wiring diagrams, dynamic models.

1. Introduction

Available reverse-engineering methods for biochemical networks span a range of techniques and modeling frameworks, from statistical methods to algorithms that have systems of ordinary differential equations as output. They require different types and amounts of experimental data to perform well. The goal of the DREAM effort is to provide a rigorous basis on which to assess and compare the performance of individual methods. This paper proposes a new reverse-engineering method within the framework of time-discrete dynamical systems over finite state sets. Such systems can be considered to form a subclass of the class of dynamic Bayesian networks. In order to bring to bear sophisticated mathematical tools we make the additional assumption that the set of variable states can be equipped with the structure of a finite number system. This assumption is familiar in the case of Boolean networks, where Boolean algebra exploits the fact that the set {0,1} is endowed with an algebraic structure. Namely, we have an addition (1+1=0) and an obvious multiplication, both of which satisfy the usual rules of arithmetic. It is natural to consider more general finite number systems, such as the integers $\mathbf{Z}$ modulo a prime integer, allowing for data discretization that is finer than the binary ON/OFF discretization.

In this paper we assume that the set *k* of possible variable states forms a finite number system called a *finite field*. It is well-known that every function $f: k^n \longrightarrow k$ on a finite field *k* can be represented by a polynomial function. Therefore we call the vector-valued function $f = (f_1,...,f_n): k^n \longrightarrow k^n$ a *polynomial dynamical system* (PDS), where each $f_i: k^n \longrightarrow k$ is called a *transition function*. Note that each $f_i \in k[x_1,...,x_n]$ is a polynomial in *n* variables. This construction allows us to apply a rich collection of algorithms from computational algebra and algebraic geometry.

Suppose we are given a data set *D* of state transition pairs $(\mathbf{s}_i, \mathbf{t}_i)$ for a network on *n* nodes. Here, the inputs $\mathbf{s}_i$ and the outputs $\mathbf{t}_i$ are *n*-tuples of elements in the finite field *k*, obtained by discretizing the real-valued measurements into finitely many states. In applications, *k* typically has less than 20 elements, depending on the dynamic range of the data. We consider the following problem: find a PDS $f: k^n \longrightarrow k^n$ such that $f(\mathbf{s}_i) = \mathbf{t}_i$. Note that as a consequence one also obtains a wiring diagram for the network. In fact, the proposed algorithm consists of two parts, the first of which infers one or a small family of most likely wiring diagrams for the network. The second part infers a most likely dynamic model for each of these wiring diagrams. While the second step in the algorithm could be used as a stand-alone inference algorithm for dynamic modeling, coupling it with the first part to provide constraints significantly improves its performance.

We present an algorithm, previously described in [8], which computes all possible minimal wiring diagrams for a given data set of measurements from a biochemical network and scores the diagrams. The algorithm uses computational algebra, namely primary decomposition of monomial ideals, as the principal tool. By assigning a probability distribution on the set of all minimal diagrams we identify a single wiring diagram or a small family of most likely diagrams. We have extended the algorithm to construct all possible dynamic models on that wiring diagram and a most likely one is identified by assigning a probability distribution on the set of all dynamic models. The algorithm to compute a most likely dynamic model involves a parameter choice, namely a total ordering on all terms of all appearing polynomials. There are infinitely many such orderings. A geometric construct, called the *Gröbner fan*, divides these infinitely many choices into finitely many segments each of which gives rise to a dynamic model. The probability distribution is constructed on these finitely many models. We validated the method on a Boolean data set from a published segment polarity network in *D. melanogaster*.

2. Network Inference

In this discussion, we will focus on the reverse engineering of a gene regulatory network; however, the proposed methods can be applied to more general settings.

The method we describe below computes the solution of the problem one gene at a time, so we can consider the outputs to be measurements for a single gene, which we will denote by t_i.

2.1 Inference of Network Topology

Let $D_l = \{(\mathbf{s}_1, t_1),...,(\mathbf{s}_m, t_m)\}$ be the data set for gene ℓ, where $\mathbf{s}_i \in k^n$ and $t_i \in k$. We first consider the problem of identifying all sets of variables $V = \{x_{e_1},\ldots,x_{e_h}\}$ such that there exists a polynomial function $f \in k[x_{e_1},\ldots,x_{e_h}]$ where $f(\mathbf{s}_i) = t_i$ for all *i*=1,…,*r* and *V* is *minimal* in the sense that if we remove any

variable x from V, there is no such function on V-$\{x\}$. Any set which satisfies these criteria we call a *minimal set*.

We briefly describe the minimal-sets algorithm that finds and scores all such minimal sets, and identifies one as the most likely by assigning a probability distribution on the set of minimal sets. While a detailed description is available in JLSS, we include it here for completeness.

We first partition the inputs $\{\mathbf{s}_1,\ldots,\mathbf{s}_m\}$ as follows. For $a\in k$, let

$$X_a = \{\mathbf{s}_i \in k^n \ : \ t_i = a\}.$$

Next we encode this information in a *monomial ideal*, that is, a set of monomials closed under addition and multiplication.

Definition 1. *Given the partition* $\{X_a : a\in k\}$ *of the inputs* $\{\mathbf{s}_1,\ldots,\mathbf{s}_m\}$, *define* $M \subset k[x_1,...,x_n]$ *to be the square-free monomial ideal generated by*

$$\{m(\mathbf{p},\mathbf{q}) \ : \ \mathbf{p}\in X_a,\ \mathbf{q}\in X_b,\ \text{and } a\neq b\in k\},$$

where $m(\mathbf{p},\mathbf{q}) \coloneqq \prod_{p_i\neq q_i} x_i$.

In other words, the monomial $m(\mathbf{p},\mathbf{q})$ encodes the coordinates in which $\mathbf{p}$ and $\mathbf{q}$ differ.

The following proposition is the key technical result underlying the algorithm.

Proposition 2. *Let* $V=\{x_{e_1},\ldots,x_{e_h}\}$ *be a subset of* $\{x_1,\ldots,x_n\}$. *Then, there is a function* $f\in k[x_{e_1},\ldots,x_{e_h}]$ *such that* $f(\mathbf{s}_i)=t_i$ *for all* i *if and only if the ideal* $\langle x_{e_1},\ldots,x_{e_h}\rangle$ *contains* M. *Furthermore, the ideals* $\langle x_{e_1},\ldots,x_{e_h}\rangle$ *are precisely the associated primes in the primary decomposition of the ideal* M.

In particular, to find all minimal sets, it is enough to find the so-called *minimal primary decomposition* of M, which is a decomposition of an ideal into an intersection of "irreducible" components analogous to decomposing a number into a product of factors. Such decompositions can easily be found using methods from computational algebra [7].

Each minimal set V is a candidate set of inputs for gene ℓ and there may potentially be a very large number of possible minimal sets. If additional information about the network is available, e.g., about the existence or absence of certain interactions, then it can be used for model selection (see [8] for more details). Next we describe a statistical measure for model selection in the case where no additional information about the network is available.

Let D_ℓ be as above and let $\mathcal{V}$ be the collection of all minimal sets from Proposition 2. For $1\leq i, j \leq n$, let Z_j be the number of sets V in $\mathcal{V}$ of length j, and let $W_i(j)$ be the number of sets V in $\mathcal{V}$ of length j such that $x_i\in V$. That is,

$$Z_j = |\{V\in V \ : \ |V| \ = j\}| \ \text{ and } \ W_i(j) \ = \ |\{V\in V \ : \ x_i\in V \ \textit{and} \ |V| \ = j\}|.$$

We now score each variable x_i as follows:

$$S(x_i) = \sum_{j=1}^{n} \frac{W_i(j)}{j\cdot Z_j}$$

Using the scores of the variables, we score each set $V\in\mathcal{V}$ as follows:

$$T(V) = \prod_{x_i \in V} S(x_i).$$

We then obtain a probability distribution on the set $\mathcal{V}$ by normalizing the scores of all sets in $\mathcal{V}$. That is, for $V \in \mathcal{V}$, we assign the normalized score

$$\tilde{T}(V) = \frac{T(V)}{\sum_{G \in V} T(G)}.$$

The minimal sets with the highest score are returned by the algorithm. By repeating this process for each gene, we obtain a family of minimal wiring diagrams for the biochemical network.

Heuristically speaking, the minimal sets algorithm captures minimal sets of variable dependencies that allow for an explanation of the data as a functional relationship, without actually computing that functional relationship. This is left to the second part of the algorithm. If more than one possible minimal set is returned, the family of them can be viewed as a set of experimental hypotheses to be tested. A strength of the algorithm is that it does in fact find ALL possible minimal wiring diagrams by completely surveying the space of possible diagrams rather than searching it heuristically. In practice, the collection of highest-scoring minimal sets is quite small.

2.2 Inference of Network Dynamics

For each of the minimal wiring diagrams returned by the first part of the algorithm we now infer a unique most likely dynamic model which has this wiring diagram as the collection of variable dependencies. As explained earlier, this can be done variable at a time, by inferring the most likely transition function for this variable. Suppose, for a fixed gene ℓ in the network, the minimal set for x_ℓ is $V_l = \{x_{e_1}, \ldots, x_{e_h}\} \subset \{x_1, \ldots, x_n\}$. Then we are interested in the set of all transition functions on V_l given D_l, that is, the set of polynomial functions such that

$$F_l = \{f \in k[x_{e_1}, \ldots, x_{e_h}] \,:\, f(p_{e_1}, \ldots, p_{e_h}) = a, \text{ for all } \mathbf{p} \in X_a,\ a \in k\}.$$

The set F_ℓ can be computed similar to how one solves a linear system of equations [9]. First, a particular solution f is computed, typically using the Lagrange interpolation formula. Then, using computational algebra, the set of homogenous solutions is characterized as the ideal $I = \{g \in k[x_{e_1}, \ldots, x_{e_h}] : g(p_{e_1}, \ldots, p_{e_h}) = 0\}$. The set $f + I$ represents the model space, from which we can select a minimal model as follows. Compute the *normal form* of f, denoted by $f \,\%\, I$, by taking the remainder of f upon division by the elements of I. This can be accomplished by first computing a *Gröbner basis* of I, which is required for multivariate polynomial division. However, the construction of a Gröbner basis requires fixing a *term order*, that is, a total ordering on all possible monomials. Selecting a different term order possibly results in a different normal form, thereby changing the model.

There is a geometric construction, called the *Gröbner fan*, which captures the number of different forms by partitioning the set of all term orders into cones where two term orders in the same cone give rise to the same normal form (For more information about the Gröbner fan, see [10]). Thus it is enough to compute the normal form of f with respect to only one term order from each cone. Then we pick the normal form that shows up most often as the transition function for x_ℓ. To be precise, suppose the Gröbner fan of I has q cones and the distinct normal forms are $\{f^1, \ldots, f^s\}$ where f^i is the normal form of f with

respect to only *h* of the *q* cones. Then the score of the normal form f^i is $\frac{h}{q}$. This gives a probability distribution on the set of normal forms. The normal form with the maximum score is chosen as the transition function for x_l. Repeating this for each gene in the network we obtain a dynamic model of the network.

In essence, the Gröbner fan approach reduces the infinitely many possible choices of term order to finitely many choices of a normal form and makes the method independent of the term order parameter by algorithmically exploring the result for all possible parameter choices. Again, this is done by systematically taking into account all possibilities rather than heuristically exploring the parameter space.

This algorithm has been implemented in the computer algebra system *Macaulay2* [6] and the Gröbner fan computations are done using the program Gfan [5].

Algorithm-1

Input: Data set $D = \{(\mathbf{s}_i, \mathbf{t}_i) \ : \ i = 1, \ldots, m\}$
Output: Minimal PDS(s) that fit *D*

For each $\ell \in \{1,...,n\}$ do

1. Compute the minimal sets for D_ℓ and select the highest-scoring set(s) *V*.
2. For each *V* do
   (a) Compute the ideal *I* of inputs in D_V, the restriction of D_ℓ to the coordinates defined by *V*.
   (b) Compute the Gröbner fan $\mathcal{G}$ of *I*.
   (c) Compute a transition function f'_l.
   (d) Compute $NF = \{f'_l \,\%G \ : \ G \in G\}$ and select the highest-scoring normal form(s) f_ℓ.
3. Return the normal form(s) f_ℓ for D_ℓ.

The output of the above algorithm is a collection of PDSs, whose transitions functions are most representative among all possible transition functions.

3. An Application to a Gene Regulatory Network

In this section, we consider a segment polarity gene network in the *D. melanogaster* embryo, for which a bottom-up Boolean model was proposed in [1]. The authors constructed a dynamic model consisting of 15 Boolean functions based on known connectivity structure of the network. In [9], the authors aimed to reconstruct this Boolean model of the network from data generated by this model. While the authors demonstrated favorable performance of their reverse-engineering method (84% identification with 20%

false positives and 15% false negatives), it relies heavily on the choice of a total ordering of the variables. Jarrah *et al.* [8] reconstructed the wiring diagram, based on the minimal-sets algorithm. We have extended these results to include a dynamic model of the network.

3.1 Methods and Results

We generated 168 state transitions from the Boolean model in [1] and applied Algorithm-1 to these data. We note that the data make up less than 0.01% of all possible state transitions.

Each node had a unique highest-scoring minimal set, with the exception of nodes 8 and 10: the data for $x8$ and $x10$ admitted two highest-scoring minimal sets each. Figure1 shows the reconstructed wiring diagram. We identified 38 edges, with all but 1 being correct, namely, the self-loop on $x11$, and missed 9 edges. While we failed to discover the remaining edges, all have been identified as nonidentifiable interactions. For more details, see [8].

For each node, the highest-scoring normal forms associated to a choice of minimal set, were returned. In this example, each minimal set admitted a unique normal form, producing 4 distinct PDSs. The transition functions are given below. Notice that each of the nodes 8 and 10 could take one of two local transitions, since each of $x8$ and $x10$ has two highest-scoring minimal sets. Underlined terms represent false positives and terms in brackets are true negatives.

For any choice of model, we correctly identified 6 of the 15 Boolean functions. In the PDS with $f8$ and $f10$, on average 90% of the terms in the transition functions are correct; however, 39% of the terms in the Boolean functions were not identified. These terms are not recoverable since they vanish on the given data and are divided out in the computation of the normal forms.

$f1 = x1$
$f2 = x1x14 + x2x14 + \underline{x2x15} + \underline{x2} +$ [12 terms]
$f3 = x2$
$f4 = x16 +$ [5 terms]
$f5 = x4$
$f6 = x5 +$ [1 terms]
$f7 = x6$
$f8 = x11x13x20 + x11x13x21 + \underline{x4x13} + x11x13 + x13x20 + x13x21 +$ [37 terms]
$f8' = x11x13x20 + x11x13x21 + x11x13 + \underline{x13x16} + x13x20 + x13x21 +$ [37 terms]
$f9 = x8x9 + \underline{x8x18} + \underline{x8x19} + x8 + x9 + \underline{x18} + \underline{x19} +$ [6 terms]
$f10 = x8x9x20 + x8x9x21 + x8x20 + x9x20 + x8x21 + x9x21 +$ [21 terms]
$f10' = \underline{x8x10x20} + \underline{x8x10x21} + x8x20 + \underline{x10x20} + x8x21 + \underline{x10x21} +$ [25 terms]
$f11 = \underline{x8x9x11} + x8x9 + \underline{x9x11} + x8x20 + x8x21 + x8 + x9 + 1 +$ [31 terms]
$f12 = x5 + 1$
$f13 = x12$
$f14 = x11x13x20 + x11x13x21 + x11x13 + x13x20 + x13x21 +$ [2 terms]
$f15 = x11x13x20 + x11x13x21 + x11x13 + x13x20 + x13x21 + x13 +$ [2 terms]

For any choice of model, we correctly identified 6 of the 15 Boolean functions. In the PDS with $f8$ and $f10$, on average 90% of the terms in the transition functions are correct; however, 39% of the terms in the Boolean functions were not identified. These terms are not recoverable since they vanish on the given data and are divided out in the computation of the normal forms. For similar analysis for the other three models, see Table 1.

This result is very interesting for two reasons. Firstly, it is extremely surprising that the method performs as well as it does on the given data set. On a random Boolean network of comparable size and with a comparable data set the method would very likely have performed extremely poorly, since the input is a vanishingly small part of the model's state space. It would be interesting to explore rigorously what is special about biological models that lead to comparatively easy identifiability. Secondly, it is due to the mathematical foundation of the method that we are able to assess the specific reason for the algorithm's failure to infer certain features of the network. In this case the data set does not contain enough information to infer some of the network connections and, therefore, some of the transition functions. This fact points to the difficulty of assessing the absolute performance of reverse-engineering methods without having a way to measure the information content of the data set used. It is possible to include more data that allow the algorithm to correctly identify all features of the network.

4. Discussion

We have presented an algorithm that identifies most likely polynomial dynamical systems for the most likely wiring diagrams. The method is an extension of the minimal-sets algorithm described in [8] in which we incorporate the Gröbner fan, a geometric object to encode the dynamic model space. The use of Gröbner fans for modeling was first introduced by Dimitrova *et al.* [3]. In their paper, the authors used the Gröbner fan to compute the model space globally, that is, in the full ring $k[x_1,...,x_n]$. We have adapted it to perform a local computation of the model space, that is, over a smaller ring $k[x_1,...,x_r]$, for $r \leq n$. The advantage to this approach is that we reduce the model space dimension by restricting the ring to the variables in a minimal set. This allows us to rigorously analyze the space of PDSs as opposed to performing random sampling as was done in [9,2].

As with all other system-identification methods of this type, a rigorous validation requires techniques to measure the quality of the given input data. No such methods have been proposed at this time for this modeling framework, an important open problem, so validation rests on individual case studies. Also, it is important to use simulated data sets rather than actual experimental data, since the underlying regulatory networks are not completely known in almost all cases.

5. Acknowledgements

BS was supported by NSF Grant 0112050 and NIH Grant RO1 GM068947-01. AJ was partially supported by NSF Grant DMS-0511441. RL was partially supported by NIH Grant RO1 GM068947-01 and NSF Grant DMS-0511441. MS was partially supported by NSF Grant DMS-0311806.

References

[1] R. Albert and H. Othmer, The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in *Drosophila melanogaster*, *J. Theor. Biol.* **223** (2003), 1–18.

[2] E. Allen, J. Fetrow, L. Daniel, S. Thomas, and D. John, *Algebraic dependency models of protein signal transduction networks from time-series data*, Journal of Theoretical Biology **238** (2006), 317–330.

[3] E. Dimitrova, A. Jarrah, R. Laubenbacher, and B. Stigler, A Gröbner fan-based method for biochemical network modeling, *2007 ISSAC Proceedings*, accepted.

[4] E. Dimitrova, P. Vera-Licona, J. McGee, and R. Laubenbacher, Discretization of time series data, submitted, 2007.

[5] A. Jensen, Gfan, a software system for Gröbner fans, Available at `http://home.imf.au.dk/ajensen/software/gfan/gfan.html`

[6] D. Grayson and M. Stillman, Macaulay 2, a software system for research in algebraic geometry, Available at `http://www.math.uiuc.edu/Macaulay2/`.

[7] S. Hoşten and G. Smith, *Monomial ideals*, Computations in algebraic geometry with Macaulay 2, Algorithms Comput. Math., vol. 8, Springer, Berlin, 2002, pp. 73–100.

[8] A. Jarrah, R. Laubenbacher, B. Stigler, and M. Stillman, Reverse-engineering of polynomial dynamical systems, *Adv. Appl. Math.*, in press.

[9] R. Laubenbacher and B. Stigler, A computational algebra approach to the reverse engineering of gene regulatory networks, *J. Theor. Biol.* **229** (2004), 523–537.

[10] B. Sturmfels, 'Gröbner Bases and Convex Polytopes', American Mathematical Society, Univ. Lectures Series, No 8, Providence, Rhode Island, 1996.

6. Appendix

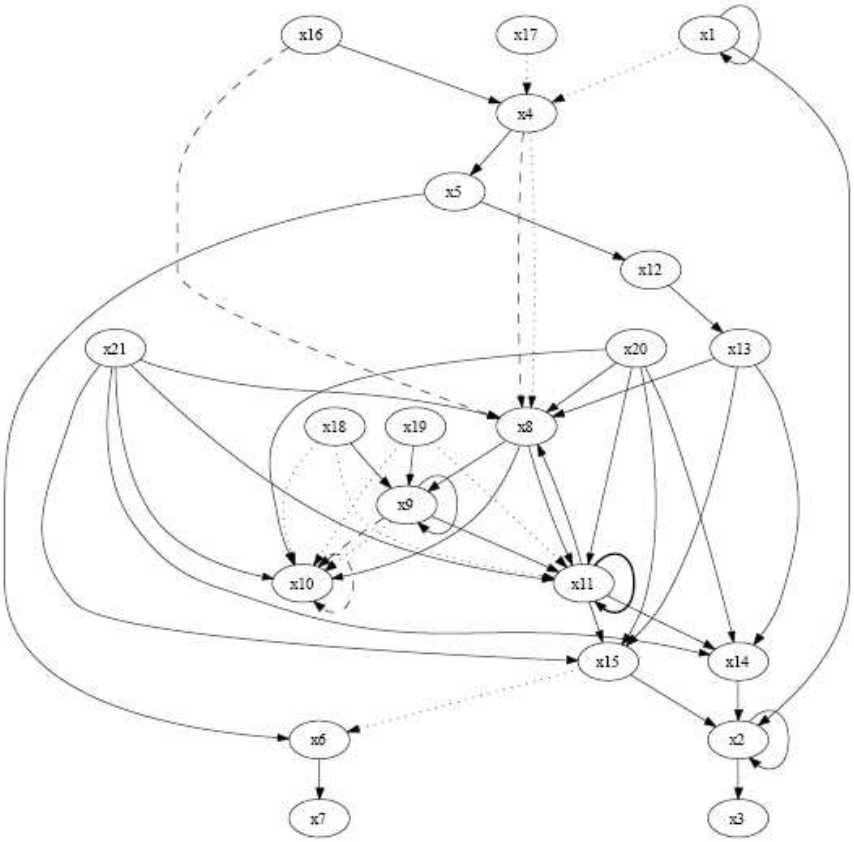

Figure 1. The reverse-engineered wiring diagram for the Boolean network. Solid and dashed edges make up the wiring diagram returned from the minimal-sets algorithm. Solid edges represent true positives; dotted, true negatives; and bold, false positives ($x11$). Dashed edges arise from multiple minimal sets. For example, $x8$ has two highest-scoring minimal sets: one involving $x4$, the other $x16$.

	TP	FP	TN
M1 (*f8,f10*)	0.90	0.10	0.39
M2 (*f8',f10*)	0.90	0.10	0.39
M3 (*f8,f10'*)	0.86	0.14	0.40
M4 (*f8',f10'*)	0.86	0.14	0.40

Table 1. A summary for the 4 PDSs for the Boolean network. Each value represents an average over the terms in the transitions functions. TP = true positives; FP = false positives; TN = true negatives.