Available online at www.sciencedirect.com

ScienceDirect

Advances in Applied Mathematics 39 (2007) 477–489

ADVANCES IN Applied Mathematics

www.elsevier.com/locate/yaama

Reverse-engineering of polynomial dynamical systems

Abdul Salam Jarrah [a], Reinhard Laubenbacher [a,*], Brandilyn Stigler [b], Michael Stillman [c]

[a] *Virginia Bioinformatics Institute, Polytechnic Institute and State University, Blacksburg, VA 24061-0477, USA*
[b] *Mathematical Biosciences Institute, The Ohio State University, Columbus, OH 43210, USA*
[c] *Mathematics Department, Cornell University, Ithaca, NY 14853, USA*

Received 19 May 2006; accepted 1 August 2006

Available online 24 October 2006

Abstract

Multivariate polynomial dynamical systems over finite fields have been studied in several contexts, including engineering and mathematical biology. An important problem is to construct models of such systems from a partial specification of dynamic properties, e.g., from a collection of state transition measurements. Here, we consider static models, which are directed graphs that represent the causal relationships between system variables, so-called wiring diagrams. This paper contains an algorithm which computes all possible minimal wiring diagrams for a given set of state transition measurements. The paper also contains several statistical measures for model selection. The algorithm uses primary decomposition of monomial ideals as the principal tool. An application to the reverse-engineering of a gene regulatory network is included. The algorithm and the statistical measures are implemented in Macaulay 2, and are available from the authors.

MSC: 93A10; 13P10

Keywords: Finite dynamical system; Finite field; Monomial ideal; Primary decomposition; Reverse-engineering; Gene regulatory network

* Corresponding author.
E-mail addresses: ajarrah@vbi.vt.edu (A.S. Jarrah), reinhard@vbi.vt.edu (R. Laubenbacher), bstigler@mbi.osu.edu (B. Stigler), mike@math.cornell.edu (M. Stillman).

0196-8858/$ – see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.aam.2006.08.004

1. Introduction

A *polynomial dynamical system* (PDS) over a finite field k is a function

$$f = (f_1, \dots, f_n) : k^n \to k^n,$$

with the coordinate functions $f_i \in k[x_1, \dots, x_n]$. Iteration of f results in a time-discrete dynamical system. PDSs are special cases of *finite dynamical systems*, which are maps $X^n \to X^n$ over arbitrary finite sets X. Such systems arise in applications in engineering (see, e.g., [3,6,15,17–19,25] as well as in biology [2,13,14]). They include in particular the class of Boolean networks ($k = \mathbb{F}_2$) and cellular automata, which are studied and applied extensively in computer science, engineering, and life science disciplines [5,8,11,12,22,23]. The problem of finding systems with specified dynamic properties has arisen in several contexts, e.g., [14,19].

The main result in this paper is an algorithm that identifies minimal sets of variables for which a model exists that explains the data. That is, the output consists of all possible minimal "wiring diagrams" of dynamic networks that fit the data. This paper also relates to the algorithm in [13]. There, the question is considered what can be inferred about a biological network from a set of experiments. The network is represented by a mapping $f : X^n \to X^n$, where n is the number of variables and X is a finite set of state values for the variables. It is shown in [13] that the problem of finding a minimal wiring diagram for the network from a set of observations is NP-hard. The author then describes a greedy algorithm that finds a wiring diagram which is "close" to minimal and as sparse as possible in polynomial time. The algorithm in this paper can be interpreted as a deterministic method that finds ALL possible minimal wiring diagrams. This is accomplished by imposing an algebraic structure on the set X and using tools from symbolic computation. The main idea is to construct a square-free monomial ideal from the given observations, whose minimal primes are generated by the desired minimal variable sets. This method is quite effective for networks arising from biological systems, even though the general problem is NP-hard.

2. Reverse-engineering of polynomial dynamical systems

We first describe briefly the algorithm in [14] and how it can be applied to the reverse-engineering of biochemical networks, in order to create a context for the algorithm in the present paper. Recent technological advances, such as DNA microarray chips, have made it possible to make large system-level measurements of chemical species in cell extracts, such as gene transcripts from large numbers of genes. It would be desirable to have efficient computational methods to extract information about regulatory interactions between genes from repeated measurements of gene transcript concentrations. The relatively high cost of gene chip technology makes it infeasible to collect sufficient time-course measurements to uniquely determine the network. A further complication is that the variance in gene chip data is still significant, adding additional complexity to the problem.

Any approach to network reconstruction must in principle proceed in three steps: (1) choose a model type, e.g., systems of differential equations or Bayesian networks. (2) Describe the space of models that are consistent with the given data set. (3) Choose a "most likely" model that generated the given data set, based on predefined selection criteria. From this model, one can then determine the structure of the network, or its dynamic behavior.

In [14] the authors proposed as choice in Step (1) the class of polynomial dynamical systems over finite fields. This choice was motivated by several considerations. Molecular data sets such as DNA microarray measurements are still sufficiently noisy to justify qualitative models that distinguish only finitely many possible states. Furthermore, biologists typically interpret such data as giving a relatively small number of regulatory states, e.g. fold-change over control measurements. Finally, discrete models for biochemical networks have a long and successful tradition, beginning with the work of S. Kauffman [12], the logical models of Snoussi and Thomas [24], and, more recently, Bayesian network models [7].

Once this model class is chosen, we need to address Step (2), the description of the model space corresponding to a given data set. To be precise, suppose we are considering n variables $x_1, \ldots, x_n$, e.g., gene transcript concentrations for n genes. Suppose that we have measured m real-valued state transition pairs $(\mathbf{s}_1, \mathbf{t}_1), \ldots, (\mathbf{s}_m, \mathbf{t}_m)$ with

$$\mathbf{s}_i = (s_{i1}, \ldots, s_{in}), \qquad \mathbf{t}_i = (t_{i1}, \ldots, t_{in}), \quad i = 1, \ldots, m,$$

where $s_{ij}, t_{ij} \in \mathbb{R}$. That is, if the system is in state $\mathbf{s}_i$, then it transitions to state $\mathbf{t}_i$ at the next time step. The first task is to choose a suitable finite field k and associated discrete state transition pairs $\mathbf{s}_i, \mathbf{t}_i$ in k^n. This is a difficult and very important step. Most existing clustering algorithms are not suitable for this purpose, and a new algorithm is proposed in [4]. An admissible model then is a function $f : k^n \to k^n$ such that

$$f(\mathbf{s}_i) = \mathbf{t}_i. \tag{1}$$

Since any function $k^n \to k$ can be represented as a polynomial function [16, p. 369], the model space under consideration is the collection of all PDSs $f = (f_1, \ldots, f_n)$ such that Eq. (1) holds.

Note that, if f and g are two such models, then $f - g$ is a polynomial function that vanishes identically on the given data set. Hence we can compute the model space by computing a particular model $f^0 = (f_1^0, \ldots, f_n^0)$, where the coordinate function $f_i^0 \in k[x_1, \ldots, x_n]$, and the ideal I of the variety given by the points $\mathbf{s}_i$. This ideal of points can be computed efficiently by the Buchberger–Möller algorithm (see [21] for a description of the algorithm). The model space is given by the tuple of cosets $(f_1^0 + I, \ldots, f_n^0 + I)$. It is clearly sufficient to solve the network reconstruction problem for each network node separately.

Step (3) then consists of the selection of a "most likely" model from the space $h + I$, where $h \in k[x_1, \ldots, x_n]$. In [14], it was proposed to choose the normal form (reduction) of any particular model h with respect to a Gröbner basis for the ideal I. The motivation for this choice was that the model should be minimal, in the sense that it should not contain any terms that vanish identically on the data. Inclusion of such terms may affect the inference of network structure by introducing interactions in the network that are not supported by the data. However, this choice is unsatisfactory for several reasons. In particular, the normal form depends on the choice of a particular term order to compute a Gröbner basis, and there is no canonical choice for such a term order. The algorithms in this paper are partially motivated by an effort by the fourth author to make the model selection criterion less dependent on the choice of term order.

In the next section we describe an algorithm that finds all sets of variables $x_{i_1}, \ldots, x_{i_r}$ for which

$$k[x_{i_1}, \ldots, x_{i_r}] \cap (h + I) \neq \emptyset,$$

and such that the intersection is empty whenever one of the variables is removed. That is, this algorithm finds all minimal sets of variables for which there exists a model consistent with the given data. One advantage is that this algorithm does not depend on the choice of a term order. In a later section we will describe how it can be used for the purpose of model selection in Step (3).

3. Minimal variable sets via primary decomposition

We restrict our attention to the data for a single coordinate. That is, we have the given data set

$$(\mathbf{s}_1, t_1), \ldots, (\mathbf{s}_m, t_m),$$

where $\mathbf{s}_i \in k^n$ and $t_i \in k$ (notice that this is a value in k and not an n-tuple). We are interested in functions $f \in k[x_1, \ldots, x_n]$ such that $f(\mathbf{s}_i) = t_i$. For $a \in k$, let

$$X_a = \{\mathbf{s}_i \mid t_i = a\},$$

and $X = \{X_a \mid a \in k\}$. Write the model space $h + I$ for the given data set as

$$Y = \bigl\{f \in k[x_1, \ldots, x_n] \mid f(\mathbf{p}) = a, \text{ for all } \mathbf{p} \in X_a,\ a \in k\bigr\}.$$

We are interested in the elements f of Y which involve a minimal number of the variables, in the sense that there exists no $g \in Y$ such that the support of g is properly contained in the support of f. To compute these minimal variable sets, we encode them in a simplicial complex.

Definition 1. For $F \subset \{1, \ldots, n\}$, let $R_F = k[x_i \mid i \notin F]$. Let

$$\Delta_X := \bigl\{F \subset \{1, \ldots, n\} \mid Y \cap R_F \neq \emptyset\bigr\}.$$

Notice that Δ_X is a simplicial complex, since if $G \subset F$, then $R_F \subset R_G$. We now associate a square-free monomial ideal to Δ_X.

Definition 2. Given X as above, let $M_X \subset k[x_1, \ldots, x_n]$ be the square-free monomial ideal generated by

$$W = \bigl\{m(\mathbf{p}, \mathbf{q}) \mid \mathbf{p} \in X_a,\ \mathbf{q} \in X_b, \text{ and } a \neq b \in k\bigr\},$$

where

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

Note that the monomial $m(\mathbf{p}, \mathbf{q})$ encode the coordinates in which $\mathbf{p}$ and $\mathbf{q}$ differ. Note that M_X is the same as the face ideal for the Alexander dual of Δ_X. The following fact is the key technical result underlying the algorithm.

Proposition 3. *For a given subset $F \subset \{1, \ldots, n\}$, $F \in \Delta_X$ if and only if the ideal $\langle x_i \mid i \notin F\rangle$ contains the monomial ideal M_X.*

Proof. First, assume that $F \in \Delta_X$. Then $Y \cap R_F \neq \emptyset$. Let $\mathbf{p} \in X_a$, $\mathbf{q} \in X_b$, with $a \neq b$. Then there exists $f \in k[x_i \mid i \notin F]$ such that $f(\mathbf{p}) = a$ and $f(\mathbf{q}) = b$. This implies that $\mathbf{p}$ and $\mathbf{q}$ must differ in some coordinate $j \notin F$. Hence, the monomial $m(\mathbf{p}, \mathbf{q})$ must contain x_j as a factor, and is therefore contained in $\langle x_i \mid i \notin F\rangle$. This proves one direction.

For the other direction, assume that $M_X \subset \langle x_i \mid i \notin F\rangle$. Then all $m(\mathbf{p}, \mathbf{q})$ are contained in this ideal, which implies that $\mathbf{p} \in X_a$ and $\mathbf{q} \in X_b$, $a \neq b$, differ in a coordinate $i \notin F$. Let f be a function which sends all $\mathbf{p} \in X_a$ to a for all a. This function can be represented as a polynomial f, and, since $\mathbf{p} \in X_a$ and $\mathbf{q} \in X_b$ differ in coordinates $i \notin F$, this polynomial can be expressed using only variables x_i, $i \notin F$. Hence $f \in Y \cap R_F$. This completes the proof of the proposition. □

The following corollary is immediate.

Corollary 4. *The minimal subsets F such that $Y \cap R_F \neq \emptyset$ are precisely the generating sets for the minimal primes in the primary decomposition of the ideal M_X.*

This corollary forms the basis for Algorithm 5.

Algorithm 5 *(Minimal Sets).*

Input: $\{(\mathbf{s}_1, t_1), \ldots, (\mathbf{s}_m, t_m)\}$, with $\mathbf{s}_i \in k^n$, $t_i \in k$.
Output: All minimal subsets $F \subset \{1, \ldots, n\}$ such that there exists a polynomial function $f \in k[\{x_i \mid i \notin F\}]$ with $f(\mathbf{s}_i) = t_i$.

Step 1. Compute the ideal M_X.
Step 2. Compute the primary decomposition of M_X.
Step 3. Compute the generating sets of all minimal primes of M_X.

Example 6. Consider the following set of state transition pairs, with entries in the field $\mathbb{F}_5$ with five elements:

$$\begin{aligned}
\mathbf{s}_1 &= \big((3,0,0,0,0),3\big),\\
\mathbf{s}_2 &= \big((0,1,2,1,4),1\big),\\
\mathbf{s}_3 &= \big((0,1,2,1,0),0\big),\\
\mathbf{s}_4 &= \big((0,1,2,1,1),0\big),\\
\mathbf{s}_5 &= \big((1,1,1,1,3),4\big).
\end{aligned}$$

Then $X_0 = \{\mathbf{s}_3, \mathbf{s}_4\}$, $X_1 = \{\mathbf{s}_2\}$, $X_2 = \emptyset$, $X_3 = \{\mathbf{s}_1\}$, and $X_4 = \{\mathbf{s}_5\}$. Recall that the ideal M_X is generated by all monomials $m(\mathbf{p}, \mathbf{q})$ with $\mathbf{p} \in X_a$ and $\mathbf{q} \in X_b$ for every $a \neq b \in k$. For $a = 1$ and $b = 3$, the monomial $m((0,1,2,1,4),(0,1,2,1,0)) = x_5$ since the two points differ in the fifth coordinate, whereas $m((0,1,2,1,4),(1,1,1,1,3)) = x_1x_3x_5$. The ideal M_X is generated by the monomials $\{x_1x_2x_3x_4, x_5\}$ and has associated primes $\langle x_1, x_5\rangle$, $\langle x_2, x_5\rangle$, $\langle x_3, x_5\rangle$, $\langle x_4, x_5\rangle$. The minimal sets of variables required to define a function for the data given above are:

$$\{x_1, x_5\}, \quad \{x_2, x_5\}, \quad \{x_3, x_5\}, \quad \{x_4, x_5\}.$$

We now discuss the complexity of Algorithm 5.

Lemma 7. *Let $|k| = q$ and let W be the generating set of the ideal M_X as in Definition* 2. *Then*

$$|W| \leqslant \frac{1}{2}\left(1 - \frac{1}{q}\right)m^2.$$

Proof. Suppose that $X = \bigcup_{l=1}^{j} X_{i_l}$, where $i_l \in k$, for all l. Suppose $|X_{i_l}| = r_{i_l}$ for all l. Then

$$|W| \leqslant \sum_{g,h=1;\, g \neq h}^{j} r_{i_g} r_{i_h} \leqslant \binom{q}{2}\left(\frac{m}{q}\right)^2 = \frac{1}{2}\left(1 - \frac{1}{q}\right)m^2. \qquad \square$$

There are two well-known methods for finding the primary decomposition of a monomial ideal like M_X, and the most efficient one uses the Alexander dual of M_X [10]. This is the method used in our implementation in Macaulay 2 [9] of the algorithms presented in this paper. Using the notion of a monomial tree in the Alexander dual approach, it is conjectured [20] that the irreducible primes of M_X can be found with order $O(|W|^2 \log(n))$.

4. Model selection

The algorithm described in the previous section produces a potentially very large number of possible models for a given data set. If additional information about the network is available, e.g., the existence or absence of certain interactions, then this can be used for model selection. This is the case for the application discussed in the next section. In this section we describe a small collection of statistical measures for model selection in the case where no additional information about the network is available. Each of the measures has a different set of underlying assumptions, ranging from a bias toward small sets to a bias toward large sets containing variables that appear in many of the minimal sets. Which measure is appropriate to use depends on the type of network under consideration.

Let $\{x_1, \ldots, x_n\}$ be the set of variables and let

$$\mathcal{F} = \{F_1, \ldots, F_t\} \subset \mathcal{P}\big(\{x_1, \ldots, x_n\}\big),$$

be the output of Algorithm 5 for a given data set. We will construct several statistical measures on this output that allow the choice of one or more subsets/models with highest probability.

For $1 \leqslant s \leqslant n$, let Z_s be the number of sets F_j in $\mathcal{F}$ of length s, and for $x_i \in \{x_1, \ldots, x_n\}$, let $W_i(s)$ be the number of sets F_j in $\mathcal{F}$ of length s such that $x_i \in F_j$. That is,

$$Z_s = \big|\{j\colon\ F_j \in \mathcal{F} \text{ and } |F_j| = s\}\big| \quad \text{and}$$
$$W_i(s) = \big|\{j\colon\ F_j \in \mathcal{F},\ x_i \in F_j \text{ and } |F_j| = s\}\big|.$$

We propose three different methods to score each variable $x_i \in \{x_1, \ldots, x_n\}$. Let

$$S_1(x_i) = \sum_{s=1}^{n} \frac{W_i(s)}{s \cdot Z_s},$$

$$S_2(x_i) = \sum_{s=1}^{n} \frac{W_i(s)}{s},$$

$$S_3(x_i) = \sum_{s=1}^{n} W_i(s).$$

Next we propose two different methods to compute a score for each set $F_j \in \mathcal{F}$. Define

$$T_1(F_j) = \prod_{x_i \in F_j} S(x_i),$$

$$T_2(F_j) = \frac{\sum_{x_i \in F_j} S(x_i)}{|F_j|},$$

where $S(x_i)$ is the score of x_i using any of the three variable-scoring methods. If we now normalize the set scores by dividing by $D = \sum_j T_i(F_j)$, $i = 1, 2$, then we obtain a probability distribution on the set $\mathcal{F}$.

Example. Let $n = 6$, and let

$$\mathcal{F} = \{F_1 = \{x_1\};\ F_2 = \{x_2, x_3\};\ F_3 = \{x_2, x_4\};\ F_4 = \{x_3, x_5, x_6\}\}.$$

Then

	S_1	S_2	S_3
x_1	1	1	1
x_2	$\frac{1}{2}$	$\frac{1}{2}$	2
x_3	$\frac{7}{12}$	$\frac{5}{6}$	2
x_4	$\frac{1}{4}$	$\frac{1}{2}$	1
x_5	$\frac{1}{3}$	$\frac{1}{3}$	1
x_6	$\frac{1}{3}$	$\frac{1}{3}$	1

The following table lists set scores using different combinations of the scoring methods S_i and T_j.

	$S_1,\ T_1$	$S_1,\ T_2$	$S_2,\ T_1$	$S_2,\ T_2$	$S_3,\ T_1$	$S_3,\ T_2$
F_1	1	1	1	1	1	1
F_2	$\frac{7}{24}$	$\frac{13}{24}$	$\frac{10}{24}$	$\frac{16}{24}$	4	2
F_3	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{2}{8}$	$\frac{4}{8}$	2	$\frac{6}{8}$
F_4	$\frac{7}{108}$	$\frac{45}{108}$	$\frac{10}{108}$	$\frac{54}{108}$	2	$\frac{4}{3}$

Using either T_1 or T_2 to score sets, we would choose F_1 if we use S_1 or S_2, and F_2 if we use S_3.

Note that using S_1 or S_2 to score variables does not always pick the singleton sets as one might suspect from the example above. Suppose we have the collection of sets

$$\{x_1\},\quad \{x_{13}\},\quad \{x_2,x_3\},\quad \{x_2,x_4,x_5\},\quad \{x_2,x_6,x_7\},\quad \{x_2,x_8,x_9\},\quad \{x_2,x_{10},x_{11},x_{12}\}.$$

Then $T(\{x_2,x_3\}) > T(\{x_1\})$, where T is either T_1 or T_2, using any of the variable-scoring methods above.

Algorithm 8 *(Model Selection).*

```
Input:   A collection of subsets $F_1, \dots, F_t$ of $\{x_1, \dots, x_n\}$.
Output:  Return the set(s) with highest probability score.

  Step 1. For each $x_i$, compute $S(x_i)$.
  Step 2. For each $F_j$, compute $T(F_j)/D$.
  Step 3. Return the set(s) $F_j$ with the highest score $T(F_j)/D$.
```

The user may now make a further selection to obtain a single model, based either on additional information about the network to be modeled or other criteria, such as choosing the simplest model. This is typically a heuristic process. Alternatively one may use this information to obtain appropriate additional data points for further model selection. For applications to biological systems, in particular biochemical networks, certain molecules may have well-understood properties, such as their role in a signaling pathway or in transcription regulation. This type of information can be used in the model selection process, and is generated by the following modification of Algorithm 8.

Algorithm 9 *(Model Selection–Additional Information).*

```
Input:   A collection of subsets $F_1, \dots, F_t$ of $\{x_1, \dots, x_n\}$.
Output:  Return all singleton sets and the variable(s) and set(s) of highest score.

  Step 1. For each $x_i$, compute $S(x_i)$.
  Step 2. For each $F_j$, compute $T(F_j)/D$.
  Step 3. Return the set(s) $F_j$ with the highest probability $T(F_j)/D$ and all variables that have
          scores equal to or higher than the lowest score of variables that appear in the $F_j$ cho-
          sen by Algorithm 8.
```

5. Network reconstruction

As described in Section 2, the algorithm in [14] chooses a reduced polynomial dynamical system that fits the given data set. The main advantage of the algorithm described in Section 3 is that it allows the restriction of the model space from which this choice is made to polynomials that include only essential collections of variables. This improves model selection substantially. We demonstrate this improvement with a simulated biochemical network in the fruit fly *D. melanogaster*.

5.1. Network reconstruction using multiple term orders

Let $f = (f_1, \dots, f_{21})$ be the PDS with coordinate functions in $\mathbb{F}_2[x_1, \dots, x_{21}]$ defined in the appendix. This dynamical system was first introduced in [1] as a Boolean network model for the segment polarity genes expressed in a developmental cycle of the fruit fly embryo. There the authors assembled the Boolean functions from the known connectivity structure, depicted as a graph in [1, Fig. 1]. The authors in [14] aimed to reconstruct the Boolean model, as well as the connectivity graph, from data generated by f (see Section 5.2). They applied the reverse-engineering method described above to the generated data and constructed a minimal PDS. (For reasons outside the scope of this discussion, we focus on the reconstruction of the first 15 functions; the remaining ones are associated to “dummy” variables introduced by the authors of [14] and are not considered here.) To test the accuracy of their polynomial model, they associated a directed graph to the PDS, which they used to compare with the wiring diagram of the Boolean model.

Definition 10. Let $g \in k[x_1, \dots, x_n]$. The *support* of g, denoted $\operatorname{supp}(g)$, is the set of variables that appear in g.

Definition 11. Let f be an n-dimensional PDS; that is, $f = (f_1, \dots, f_n)$ and $f_i \in k[x_1, \dots, x_n]$. The *dependency graph* of f, denoted $D(f)$, is a directed graph (V, E) with vertex set $V := \{x_1, \dots, x_n\}$ and edge set $E := \{(t, x_i) \mid t \in \operatorname{supp}(f_i),\ i = 1, \dots, n\}$.

The terms “wiring diagram,” “static model” and “dependency graph” are different names for the same concept and will be used interchangeably.

In the reconstruction process, the authors of [14] used 4 graded reverse lexicographical orders (grevlex) and produced a consensus dependency graph. They reported 46 edges in the graph of their polynomial model, of which 37 are correct. As the network graph has 44 edges, their method has a false-positive rate (FPR) of $\frac{46-37}{46} \approx 0.20$ and a false-negative rate (FNR) of $\frac{44-37}{46} \approx 0.15$.

While the authors demonstrated favorable performance of the reverse-engineering method, it relies heavily on the choice(s) of term order. In fact, if we repeat the exercise outlined above for only one term order, say grevlex with $x_1 > \cdots > x_n$, then we get a dependency graph with 58 edges, 37 being correct, an FPR of $\frac{58-37}{58} \approx 0.36$ and a FNR of $\frac{44-37}{58} \approx 0.12$. In [2], this approach was improved by using a large number of term orders, and applied to the reverse-engineering of a protein network.

We show next that we can improve the performance of the reverse-engineering method proposed in [14] by using the minimal sets algorithm described in Section 3.

5.2. Network reconstruction using minimal variable sets

The data set consists of 24 sets of 7 state transition pairs, each generated by applying f to 24 initializations $\mathbf{s}_0$, taken from [14]. So, each data set is comprised of pairs

$$\begin{aligned} &\big(\mathbf{s}_i, f(\mathbf{s}_{i+1})\big), \quad && \text{for } i = 0, \\ &\big(f(\mathbf{s}_i), f(\mathbf{s}_{i+1})\big), \quad && \text{for } 1 \leqslant i \leqslant 7. \end{aligned}$$

There are 6 different experimental conditions represented: $\mathrm{WT} = f$ and $\mathrm{KO}i = f^{(i)} := (f_1, \dots, f_{i-1}, 0, f_{i+1}, \dots, f_n)$ for $i = 2, 4, 6, 8, 12$. The condition WT represents data from the *wildtype*, that is, observations of a biological process in its natural state. We call $f^{(i)}$ the *knock-out for node i*, as it simulates the "knocking out" or silencing of one biochemical, namely a gene product.

For each experimental condition, there are 4 initializations, in which a small number of entries are set to 1 and the rest are set to 0. For example, the third initialization in the WT experiments has a 1 in the first, 8th, and 12th coordinates and 0s everywhere else. The table below summarizes this information.

s_0	WT	KO2	KO4	KO6	KO8	KO12
1	4, 6	4, 6	6	4	4, 6	4, 6
2	8, 12, 20	8, 12, 20	8, 12, 20	8, 12	12, 20	8, 20
3	1, 8, 12	1, 8, 12	1, 8, 12	1, 8, 12	1, 12	1, 8
4	1, 2, 8, 12, 21	1, 8, 12, 21	1, 2, 8, 12, 21	1, 2, 8, 12	1, 2, 12, 21	1, 2, 8, 21

We applied Algorithm 5 to the generated data and computed the minimal sets for each node. We note that minimal sets are not unique (see Example 6). In fact, for only 9 of the 15 functions is there a unique minimal set of variables. Let us restrict our attention to the following coordinate functions of the true network for which there is more than one choice:

$$\begin{aligned}
f_8 &= (x_4+1)(x_{13})\big[(x_{11}+1)(x_{20}x_{21}+x_{20}+x_{21})+x_{11}\big],\\
f_9 &= (x_{19}+1)(x_8x_9x_{18}+x_8x_9+x_9x_{18}+x_9)+x_8,\\
f_{10} &= f_9(x_{20}x_{21}+x_{20}+x_{21}),\\
f_{11} &= f_9+f_{10}+1.
\end{aligned}$$

There are 30, 19, 2, and 5 choices of minimal sets, respectively. In each case, we chose the set that coincided with basic biological properties of the network. To see this, consider the subgraph G of the dependency graph of f generated by the support of $f_8, \ldots, f_{11}$. This graph has 21 edges and is given in Fig. 1.

The functions f_8, f_{10} and f_{11} are associated to biochemicals known to not directly regulate their own synthesis, so we selected those sets that do not contain x_8, x_{10}, and x_{11}, respectively. We made similar selections for the reconstruction of f_9. This resulted in identification of 19 of the 21 expected edges, all of which are correct. The 2 edges not discovered correspond to

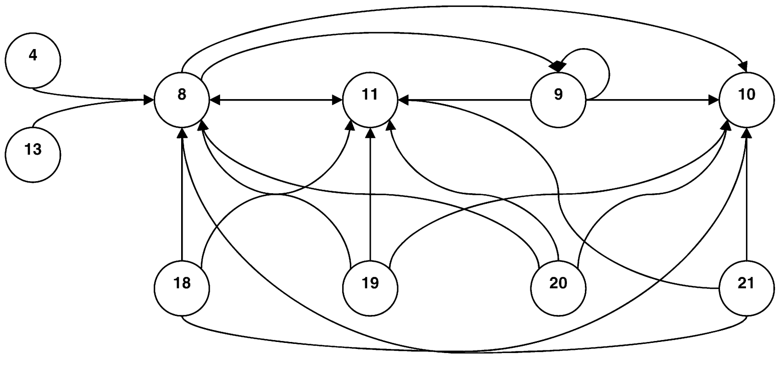

Fig. 1.

variables that are in the support of f_{10}, namely x_{18} and x_{19}. Upon inspection, we find that the part of f_{10} involving these two variables is identically 0 on the given data set. Consequently, the corresponding interactions in the network are not identifiable using this data set.

For the entire network, we identified 39 edges, all of which are correct, using Algorithms 5 and 9. While we failed to discover the remaining 5 edges, all have been identified as corresponding to elements of the ideal of the inputs.

6. Discussion

We have presented an algorithm that identifies all possible minimal dependency graphs of polynomial dynamical systems that fit a given data set of state transition pairs. The algorithm does not share the shortcoming of dependence on the choice of a term order, present in the algorithm in [14], which, on the other hand, generates an actual dynamical system model that reproduces the data. And we have compared the two algorithms by applying them to the same data set, generated from a Boolean model of fruit fly embryonic development. Furthermore, Algorithms 5 and 8 improve on the greedy algorithm described in [13].

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.

Acknowledgments

The authors thank Miguel Colón-Velez for the implementation of the statistical measures. The first and second authors were supported partially by NSF Grant DMS-0511441. The second and third authors were supported partially by NIH Grant RO1 GM068947-01, a joint computational biology initiative between NIH and NSF. The fourth author was supported partially by NSF Grant DMS-0311806.

Appendix A

Following is the PDS consisting of 21 functions in $\mathbb{F}_2[x_1,\dots,x_{21}]$ used as an example in Section 5.

$$\begin{aligned}
f_1 &= x_1,\\
f_2 &= x_1x_2x_{15} + x_1x_{14}x_{15} + x_2x_{14}x_{15} + x_1x_2 + x_1x_{14} + x_2x_{14},\\
f_3 &= x_2,\\
f_4 &= x_1x_{16}x_{17} + x_1x_{16} + x_1x_{17} + x_{16}x_{17} + x_{16} + x_{17},\\
f_5 &= x_4,\\
f_6 &= x_5x_{15} + x_5,\\
f_7 &= x_6,\\
f_8 &= (x_4+1)(x_{13})\big[(x_{11}+1)(x_{20}x_{21} + x_{20} + x_{21}) + x_{11}\big],\\
f_9 &= (x_{19}+1)(x_8x_9x_{18} + x_8x_9 + x_9x_{18} + x_9) + x_8,
\end{aligned}$$

$$\begin{aligned}
f_{10} &= f_9(x_{20}x_{21} + x_{20} + x_{21}),\\
f_{11} &= f_9 + f_{10} + 1,\\
f_{12} &= x_5 + 1,\\
f_{13} &= x_{12},\\
f_{14} &= (x_{11} + 1)(x_{13}x_{20}x_{21} + x_{13}x_{20} + x_{13}x_{21} + x_{13}) + x_{13},\\
f_{15} &= f_{14} + x_{13},\\
f_i &= x_i \quad \text{for } 16 \leqslant i \leqslant 21.
\end{aligned}$$

References

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

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

[3] P. Cull, Linear analysis of switching nets, Kybernetik 8 (1971) 31–39.

[4] E. Dimitrova, R. Laubenbacher, J. McGee, Discretization of time series data, 2005, under review.

[5] A. Doeschl, M. Davison, H. Rasmussen, G. Reid, Assessing cellular automata based models using partial differential equations, Math. Comput. Modelling 40 (2004) 977–994.

[6] B. Elspas, The theory of autonomous linear sequential networks, IRE Trans. Circuit Theory CT-6 (1959) 45–60.

[7] N. Friedman, M. Linial, I. Nachman, D. Pe'er, Using Bayesian networks to analyze expression data, J. Comput. Biol. 7 (2000) 601–620.

[8] A. García-Olivares, M. Villarroel, P. Marijuán, Enzymes as molecular automata: A stochastic model of self-oscillatory glycolytic cycles in cellular metabolism, BioSystems 56 (2000) 121–129.

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

[10] S. Hoşten, G. Smith, Monomial ideals, in: Computations in Algebraic Geometry with Macaulay 2, in: Algorithms Comput. Math., vol. 8, Springer, Berlin, 2002, pp. 73–100.

[11] M.-Th. Hütt, R. Neff, H. Busch, F. Kaiser, Method for detecting the signature of noise-induced structures in spatiotemporal data sets, Phys. Rev. E 66 (2) (2002) 26117–26127.

[12] S. Kauffman, Metabolic stability and epigenesis in randomly constructed genetic nets, J. Theoret. Biol. 22 (1969) 437–467.

[13] B. Krupa, On the number of experiments required to find the causal structure of complex systems, J. Theoret. Biol. 219 (2002) 257–267.

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

[15] M. LeBorgne, A. Benveniste, P. LeGuernic, Polynomial dynamical systems over finite fields, in: G. Jacob, F. Lamnabhi-Lagarrigue (Eds.), Algebraic Computing in Control (New York), in: Lecture Notes in Control and Inform. Sci., vol. 165, Springer, 1991, pp. 212–222.

[16] R. Lidl, H. Niederreiter, Finite Fields, second ed., Encyclopedia Math. Appl., vol. 20, Cambridge Univ. Press, New York, 1997.

[17] H. Marchand, M. LeBorgne, On the optimal control of polynomial dynamical systems over $\mathbb{Z}/p\mathbb{Z}$, in: Proceedings of the Fourth Workshop on Discrete Event Systems, Cagliari, Italy, IEEE, 1998, pp. 385–390.

[18] H. Marchand, M. LeBorgne, Partial order control of discrete event systems modeled as polynomial dynamical systems, in: Proceeding of the 1998 IEEE International Conference on Control Applications, Trieste, Italy, IEEE, 1998, pp. 817–822.

[19] D. Milligan, M. Wilson, The behavior of affine Boolean sequential networks, Connection Science 5 (2) (1993) 153–167.

[20] R.A. Milowski, Computing Irredundant Irreducible Decompositions of Large Scale Monomial Ideals, ISSAC, 2004, pp. 235–242.

[21] L. Robbiano, Gröbner bases and statistics, in: Gröbner Bases and Applications, Linz, 1998, London Math. Soc. Lecture Note Ser., vol. 251, Cambridge Univ. Press, Cambridge, 1998, pp. 179–204.
[22] I. Shmulevich, E. Dougherty, S. Kim, W. Zhang, Probabilistic Boolean networks: A rule-based uncertainty model for gene regulatory networks, Bioinformatics 18 (2) (2002) 261–274.
[23] G. Sirakoulis, I. Karafyllidis, Ch. Mizasa, V. Mardirisa, A. Thanailakis, P. Tsalides, A cellular automaton model for the study of DNA sequence evolution, Comput. Biol. Medicine 33 (2003) 439–453.
[24] E. Snoussi, R. Thomas, Logical identification of all steady states: The concept of feedback loop characteristic states, Bull. Math. Biol. 55 (1993) 973–991.
[25] M. Wilson, D. Milligan, Cyclic behavior of autonomous synchronous Boolean networks: Some theorems and conjectures, Connection Science 4 (2) (1992) 143–154.