

A Gröbner Fan Method for Biochemical Network Modeling

Elena S. Dimitrova^{*}
Mathematical Sciences Dept.
Clemson University
O-303 Martin Hall
Clemson, SC 29634
edimit@clemson.edu

Abdul Salam Jarrah and
Reinhard Laubenbacher
Virginia Bioinformatics Inst.
Virginia Tech
Washington St. (0477)
Blacksburg, VA 24061
{ajarrah,reinhard}@vbi.vt.edu

Brandilyn Stigler
Mathematical Biosci. Inst.
The Ohio State University
250 Mathematics Building
Columbus, OH 43210
bstigler@mbi.osu.edu

ABSTRACT

Polynomial dynamical systems (PDSs) have been used successfully as a framework for the reconstruction, or *reverse engineering*, of biochemical networks from experimental data. Within this modeling space, a particular PDS is chosen by way of a Gröbner basis, and using different monomial orders may result in different polynomial models. In this paper, we present a systematic method for selecting most likely polynomial models for a given data set, using the Gröbner fan of the ideal of the input data. We apply the method to reverse engineer two biochemical networks, a Boolean model of lactose metabolism in *E. coli* and a protein signal transduction network in *S. cerevisiae*, and compare our results to those from two published network-reconstruction methods.

Categories and Subject Descriptors

G.2 [Discrete Mathematics]: Applications

General Terms

Algorithms

Keywords

Reverse engineering, network inference, computational algebra, polynomial dynamical systems, monomial orderings, model selection, Gröbner bases, Gröbner fan.

1. INTRODUCTION

Systems of polynomial equations have been used extensively in optimization, coding, robotics, control theory, statistics, and other fields. Recently, Laubenbacher and Stigler [6] introduced the use of multivariate polynomial functions to reconstruct, or *reverse engineer*, biochemical networks from discretized time course experimental data, such as gene expression measurements. Subsequently the method has been

^{*}Corresponding author

applied to other types of time course biological data, such as protein modification data [2]. It has been demonstrated that the Laubenbacher-Stigler method [6] can successfully capture the dependencies among the network nodes, represented by variables in a polynomial ring over a finite field, and provide extensive information about network dynamics.

In this setting the dependency relations, as well as network dynamics, are encapsulated by a space of polynomial functions. Models are chosen using Gröbner bases of the ideal of the input data points. However, different monomial orders may give rise to different polynomial models, thereby affecting the identification of network dependencies and dynamics. Considering only one arbitrarily chosen monomial ordering is not sufficient to explore all the solutions of the reverse-engineering problem and to choose “the best” model from this space. Therefore, a systematic method for studying the monomial orderings that affect the model selection is crucial for modeling approaches utilizing Gröbner bases.

A naïve approach is to compute all possible Gröbner bases with respect to all monomial orderings. The number of monomial orders, however, grows rapidly with the number of variables n and can be as large as $n^2n!$ [8], and hence considering all of them is computationally challenging. An alternative approach [2] generates a collection of polynomial models from a fixed number of graded reverse lexicographic orders with random variable orders and computes a consensus model using a game-theoretic method, which provides a lower bound on the number of variable orderings. While it is reasonable to try to avoid considering all monomial orderings, restricting oneself to variable orderings within a fixed monomial ordering can possibly omit a large number of polynomial models that fit the data and thus prevent the selection of the most appropriate model.

In this paper we propose a systematic method to sample the modeling space so that all distinct models are obtained. From this sampling, we generate a consensus graph representing the most likely dependency relations among the nodes in the network. The method uses the *Gröbner fan* and its strength is due to the fact that any two monomial orders in the same *cone* of the fan give rise to the same polynomial model, and hence using one representative from each cone is enough to produce all polynomial models.

In Section 2 we outline the algebraic modeling framework and in Section 3 we present an algorithm which finds all monomial orders of interest and constructs a consensus graph of the dependencies among the variables. In Section 4 we use the proposed method to reverse engineer the variable dependencies in a Boolean model of lactose metabolism in

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ISSAC'07, July 29–August 1, 2007, Waterloo, Ontario, Canada.
Copyright 2007 ACM 978-1-59593-743-8/07/0007 ...\$5.00.

E. coli. We demonstrate that the choice of data points to be used is more important than the amount of data utilized and that this choice is specific to the system under consideration. Then in Section 5 we apply the method to the protein modification data from [9] and compare the dependencies we inferred to those found in [2].

2. AN ALGEBRAIC APPROACH TO REVERSE ENGINEERING

Let k be a field. A *polynomial dynamical system* (PDS) of dimension n is a function $f = (f_1, \dots, f_n) : k^n \rightarrow k^n$ with coordinate functions $f_i : k^n \rightarrow k$. That is, for $\mathbf{x} = (x_1, \dots, x_n) \in k^n$, we have $f(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_n(\mathbf{x}))$. Since every function over a finite field is a polynomial, then any function $f : k^n \rightarrow k^n$ is a PDS. The reverse engineering method of [6] is interested mostly in the case when k is a finite field of cardinality p , *i.e.* $k = \mathbb{F}_p$, and thus the polynomials f_i are in the quotient ring $R = k[x_1, \dots, x_n] / \langle x_1^p - x_1, \dots, x_n^p - x_n \rangle$. As a result, they are polynomials in n variables with coefficients in k and the degree of each variable is at most equal to $p - 1$.

In [6] biochemical networks, such as gene regulatory networks, are modeled as polynomial dynamical systems. The objective of the Laubenbacher-Stigler method is to identify dependency relations among the nodes in a network from *discrete* data. Their algorithm constructs the set of all PDSs that fit the data and then uses a minimality criterion to select one system from the set. The construction of all PDSs is not done via enumeration, but rather it is accomplished by way of Gröbner bases. We include a brief description of their algorithm.

Laubenbacher-Stigler Algorithm

Input: A time course of network states $\mathbf{s}_1, \dots, \mathbf{s}_m \in k^n$ where $\mathbf{s}_i = (s_{i1}, \dots, s_{in})$ and a monomial order \succ .

Output: A *minimal* PDS $f = (f_1, \dots, f_n)$ with coordinate polynomials $f_i \in k[x_1, \dots, x_n]$ such that

$$f_i(\mathbf{s}_j) = s_{j+1,i}$$

for all $i = 1, \dots, n$ and $j = 1, \dots, m - 1$, such that f_i does not contain terms that vanish on the input points.

Step 1: Compute a particular PDS $F = (F_1, \dots, F_n) : k^n \rightarrow k^n$ that fits the data.

Step 2: Let $S = \{\mathbf{s}_1, \dots, \mathbf{s}_{m-1}\}$. Compute $I = \mathcal{I}(S)$, the ideal of the input points in S .

Step 3: Compute the reduced Gröbner basis \mathcal{G} of I with respect to \succ . Compute $f = (f_1, \dots, f_n)$ where f_i is the normal form of F_i with respect to \mathcal{G} .

There are several methods for computing the initial PDS F in Step 1, Lagrange interpolation being one of them. In Step 2, notice that if two polynomials $f_i, g_i \in k[x_1, \dots, x_n]$ satisfy $f_i(\mathbf{s}_j) = s_{j+1,i} = g_i(\mathbf{s}_j)$, then $(f_i - g_i)(\mathbf{s}_j) = 0$ for all j . Therefore, in order to find all functions that fit the data, we need to find all functions that vanish on the given time points, which are the polynomials in $I = \mathcal{I}(S)$. All possible PDSs that fit the time course data are obtained in the form $F = f + h$, as we do in Step 3, where the coordinate polynomials of h run through all elements of I . The function

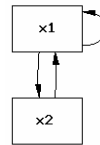


Figure 1: Dependency graph of (1).

f is called the *normal form* of F with respect to the reduced Gröbner basis for I under a fixed monomial ordering.

One drawback of the previous algorithm is its dependence on the choice of monomial order. Two monomial orders may produce different models. To have a good understanding of the network, one may want to explore all possible models, which requires identifying all monomial orders that produce different models. As we will see below, it is in fact sufficient to consider a subset of all possible orders for the purpose of reverse engineering.

3. RECONSTRUCTING DEPENDENCIES

For a PDS $f = (f_1, \dots, f_n)$, we say that variable x_i *depends on* x_j if x_j appears in f_i with a nonzero coefficient. The directed graph on $\{x_1, \dots, x_n\}$ representing these dependencies is called the *dependency graph* of f . For example, let $f = (f_1, f_2) \in \mathbb{F}_2^2[x_1, x_2]$ have coordinate polynomials

$$\begin{aligned} f_1 &= x_1 x_2 \\ f_2 &= x_1 + 1 \end{aligned} \quad (1)$$

Then x_1 depends on both x_1 and x_2 , while x_2 depends only on x_1 . The *dependency graph* of (1) is given in Figure 1.

Therefore any PDS has an associated dependency graph which encodes the dependency relations among the variables. For a given data set S , we present an algorithm that computes the most likely dependency graph.

3.1 Gröbner Fan

A combinatorial structure that contains information about all reduced Gröbner bases of a polynomial ideal I is the *Gröbner fan* of I . It is a polyhedral complex of cones, each corresponding to an initial ideal of I . The cones are in a one-to-one correspondence with the marked reduced Gröbner bases of I . (*Marked* means that the initial terms of each generating polynomial are distinguished). For details, see [7, 10].

There are algorithms that compute the Gröbner fan of a polynomial ideal. An excellent implementation of such an algorithm is the software package **Gfan** [5] that we used for the current work.

3.2 Reconstructing Dependencies

Finding all minimal polynomial models that fit the data provides all possible sets of variable dependencies. These models are the PDSs from Step 3 of the Laubenbacher-Stigler algorithm: the normal form of the interpolating PDS F with respect to one Gröbner basis corresponding to one cone of the Gröbner fan. If one is interested in finding a set of variable dependencies that most accurately reflects the actual network of interactions, some scoring method should

be applied that distinguishes the variable dependencies that appear most often in the F 's. One such technique is the algorithm developed in [2], which uses a Monte Carlo method to approximate the Deegan-Packel Index of Power (see [3] for details). It identifies dependencies for each variable x_i given time course data in the form of a matrix M with entries from a finite field k . We modified their algorithm in [2] (Steps 4* and 6*) to incorporate the Gröbner fan. A brief outline follows.

Algorithm DEP-GR

Input. A $k_1 \times n_1$ matrix M of time course data for a network, where k_1 is the length of the time course and n_1 is the number of biochemicals measured.

Output. A dependency graph of the network.

Step 1. Construct a $k_1 \times n_1$ matrix M_1 by discretizing M .

Step 2. Remove duplicate rows and columns from M_1 to create an $l \times n$ matrix Q .

Step 3. Initialize D as an $n \times n$ zero matrix.

Step 4*. Compute the reduced Gröbner basis of I with respect to any monomial ordering corresponding to a cone in the Gröbner fan \mathcal{F} of I . Compute the interpolating coordinate functions $f_i(x_1, \dots, x_n)$.

Step 5. Update D using the Deegan-Packel Index of Power.

Step 6*. Repeat Steps 4* and 5 for all cones of \mathcal{F} .

Step 7. Use D to identify dependencies: x_i depends on x_j if the difference between $D_{i,j}$ and the average entry in row i is greater than a given threshold T .

4. RECONSTRUCTING DEPENDENCIES IN *E. COLI*

We construct a very simple but logically accurate Boolean model of lactose metabolism in the bacterium *E. coli* and use it to generate data points. We apply Algorithm DEP-GR to these data to reconstruct the dependency graph of the Boolean model.

4.1 Lactose Metabolism

The primary food source for many bacteria is glucose. When glucose is sparse, bacteria have innate mechanisms for converting other sugars into glucose. One such mechanism in *E. coli* is encapsulated in the *lac* operon, a transcriptionally regulated system consisting of the genes and their regulatory regions that participate in lactose metabolism [1]. Allolactose, an isomeric form of lactose acts as an inducer of the *lac* operon. It enters the cell and binds to the *lac* repressor, inducing a conformational change that allows the repressor to fall off the DNA. In this configuration, the *lac* genes are activated and the associated proteins can then transport more lactose into the cell and metabolize it into various other sugars, including glucose and allolactose. When allolactose is not longer present in the cell, the repressor returns to its original conformation and binds to the DNA. In this configuration, no *lac*-regulated proteins are made. For the purpose of illustration, we shall ignore the role of glucose in the regulation and consider this simplified regulatory network.

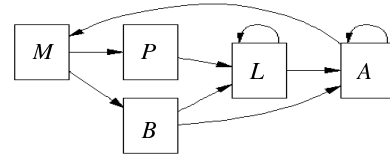


Figure 2: Dependency graph of the Boolean model (2) of lactose metabolism.

4.2 Boolean Model of Lactose Metabolism

Following the rules described in the preceding paragraph, we constructed a simple Boolean model that captures the network's major characteristics. The model involves five variables:

M : genomic region containing the *lac* genes b and p

B : beta-galactosidase, a protein encoded by b which cleaves lactose to produce glucose, and allolactose

A : allolactose

L : intracellular lactose

P : lactose permease, a protein encoded by p which transports extracellular lactose into the cell

We also make the following assumptions: molecular synthesis and din degradation require one time unit and extracellular lactose is always available. The resulting model (2) is given in Boolean and polynomial form whenever the two differ and its dependency graph is presented in Figure 2.

$$\begin{aligned}
 f_M &= A \\
 f_B &= M \\
 f_A &= A \vee (L \wedge B) = A + LB + ALB \\
 f_L &= P \vee (L \wedge \neg B) = P + L(B + 1) + PL(B + 1) \\
 f_P &= M
 \end{aligned} \tag{2}$$

From f_L it follows that intracellular lactose is available (has value 1) if at the previous system state lactose permease (P) was present to transport lactose if there already was lactose in the cell (L) and there was no beta-galactosidase (B) to cleave it. The other regulatory relationship can be read from the remaining functions in a similar fashion.

4.3 Reconstructing the Boolean Model

With data generated from the Boolean model, we show how to reverse engineer the dependency graph for model (2) using Algorithm DEP-GR. Since we have a Boolean model in five variables, there are $2^5 = 32$ possible system states and the order in which the network transitions from state to state constitutes our data.

We first used seven state transitions (about 22% of the total number of transitions) from the four series of transitions listed below. The variables are listed in the order (M, B, A, L, P) and $S_1 \rightarrow S_2$ indicates that the system changes from state S_1 to state S_2 after one unit of time. $S \circlearrowleft$ means that S is a fixed point.

Series 1: $(0, 1, 0, 0, 0) \rightarrow (0, 0, 0, 0, 0) \circlearrowleft$

Series 2: $(1, 1, 0, 0, 0) \rightarrow (0, 1, 0, 0, 1) \rightarrow (0, 0, 0, 1, 0) \circlearrowleft$

Series 3: $(1, 0, 1, 1, 0) \rightarrow (1, 1, 1, 1, 1) \circlearrowleft$

Series 4: $(0, 1, 0, 1, 0) \rightarrow (0, 0, 1, 0, 0)$

Using Lagrange interpolation and a lexicographic monomial order, we found the following PDS $F: \mathbb{F}_2^5 \rightarrow \mathbb{F}_2^5$:

$$\begin{aligned} F_M &= A \\ F_B &= M \\ F_A &= BL + A \\ F_L &= BL + L + P \\ F_P &= M \end{aligned}$$

The ideal of the input points in Series 1–4 has a Gröbner fan with 13 cones. We selected a monomial ordering and a Gröbner basis corresponding to each cone and computed the 13 normal forms of F (only ten of them were distinct). The resulting matrix D is

$$D = \begin{bmatrix} 0.0 & 0.0 & \mathbf{1.0} & 0.0 & 0.0 \\ \mathbf{1.0} & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.088 & \mathbf{0.262} & \mathbf{0.262} & \mathbf{0.262} & 0.127 \\ 0.0 & \mathbf{0.333} & 0.0 & \mathbf{0.333} & \mathbf{0.333} \\ \mathbf{1.0} & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}.$$

The mean of each row is 0.2 and only the entries given in bold are greater than it. As described in Step 7, we identify the variable dependencies based on these values and conclude that the corresponding dependency graph is identical to the one of the original system (Figure 2).

We applied other variable dependency identification methods [4] to the same data set. None of them completely recovered the dependency graph of the Boolean model: there were missing or additional edges on the graph and/or the methods were not decisive in selecting the correct variable dependency versus a non-existing one.

4.4 Effects of Data Selection on the Algorithm

We were successful in recovering the variable dependencies of the Boolean model (2) from the surprisingly small data set of Series 1–4, demonstrating the usefulness of the proposed Gröbner fan method. However, we may not have been able to reverse engineer all dependencies had we chosen a different data set. Intuitively one may suggest that the more data are used, the better the chances are to correctly recover the variable dependencies. The following example shows that the choice of data affects the algorithm’s ability to identify dependencies.

Consider the data in Series 1, 2, 3’, and 4’ where

Series 3’: $(1, 0, 0, 1, 0) \rightarrow (0, 1, 0, 1, 1) \rightarrow (0, 0, 1, 1, 0) \rightarrow (1, 0, 1, 1, 0) \rightarrow (1, 1, 1, 1, 1) \circlearrowleft$

Series 4’: $(0, 1, 1, 0, 0) \rightarrow (1, 0, 1, 0, 0) \rightarrow (1, 1, 1, 0, 1) \rightarrow (1, 1, 1, 1, 1) \circlearrowleft$

This data set consists of 44% of the total number of state transitions (14 transitions, twice as many as before). The Gröbner fan of the ideal of the input points has eight cones. Using Algorithm DEP-GR, we identified the variable dependencies as displayed in Figure 3. As compared to the “real” dependency graph in Figure 2, we see that we did not recover all dependencies as two edges are missing: an edge from B to L and an edge from B to A . These omissions would incorrectly imply that beta-galactosidase has no effect on allolactose or intracellular lactose.

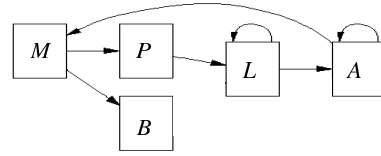


Figure 3: Dependency graph using the data in Series 1, 2, 3’, and 4’ generated from (2). Two edges are missing ($B \rightarrow L$ and $B \rightarrow A$) compared to that of the Boolean model.

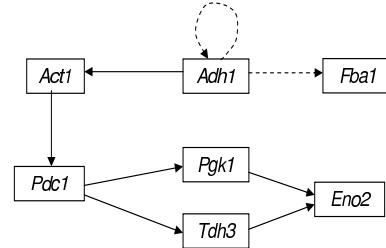


Figure 4: Dependency graph proposed by Allen *et al.* (solid arrows) and using DEP-GR (solid and dashed arrows).

As the above example shows, selecting an appropriate data set is an important issue that affects the identifiability of variable dependencies. For example, we were not able to reconstruct the dependency graph using Series 3 and 4 (or 3’ and 4’) alone. The problem becomes additionally complicated when working with real data since experimentally it may not always be possible to collect information on all desired state transitions. In practice, care must be taken when using experimental data to model network dependencies. However, this issue is outside of the scope of this paper.

5. RECONSTRUCTING DEPENDENCIES IN YEAST

As another example, we apply our method to reverse engineer the dependency graph for a signal transduction network in the yeast *S. cerevisiae* and compare the results to those in [2]. Signal transduction networks are complex pathways that allow the cell to receive, transmit, and act upon molecular signals. The signals in these networks are often transmitted by post-translational modification of proteins.

We use the protein carbonylation data of [9] detected in response to copper-induced stress in *S. cerevisiae* to identify relationships between seven proteins. The data consist of measurements for the proteins *Act1*, *Adh1*, *Eno2*, *Fba1*, *Pdc1*, *Pgk1* and *Tdh3* at times 0, 5, 15, 30, and 60 minutes. We use the data discretization of [2] and a threshold T that equals the average of the entries in each row of matrix D plus one standard deviation. The variable dependencies we identified are presented in Figure 4 (all edges).

The solid edges are the dependencies identified by [2]. Therefore, in addition to their results, we also predict that protein *Adh1* influences not only *Act1* but also *Fba1* and itself.

6. CONCLUSIONS AND FUTURE WORK

The use of the Gröbner fan for the reverse engineering of biochemical networks is a powerful and systematic way of obtaining a complete list of all models that fit the data which allows for the identification of the network dependencies. Future work includes exploration of other scoring methods for identifying variable dependencies and possibly constructing new ones. More rigorous results on the type and amount of data needed for variable-dependency recovery will be sought. We are currently applying our method to other data sets, with a focus on model validation.

7. REFERENCES

- [1] “MIT biology hypertextbook”. World Wide Web, 2006.
<http://web.mit.edu/esgbio/www/pge/lac.html>.
- [2] E. Allen, J. Fetrow, L. Daniel, S. Thomas, and D. John. Algebraic dependency models of protein signal transduction networks from time-series data. *J. Theor. Biol.*, 238:317–330, 2006.
- [3] J. Deegan and E. Packel. A new index for simple n-person games. *Int. J. Game Theory*, 7:113–123, 1978.
- [4] A. S. Jarrah, R. Laubenbacher, B. Stigler, and M. Stillman. Reverse-engineering of polynomial dynamical systems. *Adv. Appl. Math.*, In Press, 2006.
- [5] A. Jensen. Gfan, a software system for Gröbner fans, <http://home.imf.au.dk/ajensen/software/gfan/gfan.html>, 2005.
- [6] R. Laubenbacher and B. Stigler. A computational algebra approach to the reverse engineering of gene regulatory networks. *J. Theor. Biol.*, 229:523–537, 2004.
- [7] T. Mora and L. Robbiano. Gröbner fan of an ideal. *J. Symbolic Computation*, 6(2/3):183–208, 1988.
- [8] L. Robbiano. On the theory of graded structures. *J. Symbolic Computation*, 2:139–170, 1986.
- [9] A. Shanmuganathan, S. Avery, S. Willetts, and J. Houghton. Copper-induced oxidative stress in *Saccharomyces cerevisiae* targets enzymes of the glycolytic pathway. *FEBS Lett.*, 556:253–259, 2004.
- [10] B. Sturmfels. *Groebner Bases and Convex Polytopes (University Lecture Series, No. 8)*. American Mathematical Society, 1996.