RESEARCH

Open Access

Probabilistic polynomial dynamical systems for reverse engineering of gene regulatory networks

Elena S Dimitrova[1*], Indranil Mitra[2] and Abdul Salam Jarrah[3,4]

Abstract

Elucidating the structure and/or dynamics of gene regulatory networks from experimental data is a major goal of systems biology. Stochastic models have the potential to absorb noise, account for un-certainty, and help avoid data overfitting. Within the frame work of probabilistic polynomial dynamical systems, we present an algorithm for the reverse engineering of any gene regulatory network as a discrete, probabilistic polynomial dynamical system. The resulting stochastic model is assembled from all minimal models in the model space and the probability assignment is based on partitioning the model space according to the likeliness with which a minimal model explains the observed data. We used this method to identify stochastic models for two published synthetic network models. In both cases, the generated model retains the key features of the original model and compares favorably to the resulting models from other algorithms.

Keywords: Stochastic modeling, polynomial dynamical systems, reverse engineering, discrete modeling

Introduction

The enormous accumulation of experimental data on the activities of the living cell has triggered an increasing interest in uncovering the biological networks behind the observed data. This interest could be in identifying either the *static network*, which is usually a labeled directed graph describing how the different components of the network are wired together, or the *dynamic network*, which describes how the different components of the network influence each other. Identifying dynamic models for gene regulatory networks from transcriptome data is the topic of numerous published articles, and methods have been proposed within different computational frameworks, such as continuous models using differential equations [1,2], discrete models using Boolean networks [3], Petri nets [4-6], or Logical models [7,8], and statistical models using dynamic Baysein networks [9,10], among many other methods. For an up-to-date review of the state-of-the-art of the field, see, for example [11,12]. Most of these methods identify a particular model of the network which could be deterministic or stochastic. Due to the fact that the experimental data are typically noisy

and of limited amount and that gene regulatory networks are believed to be stochastic, regardless of the used framework, stochastic models seem a natural choice [9,13,14]. Furthermore, discrete models where a gene could be in one of a finite number of states are more intuitive, phenomenological descriptions of gene regulatory networks and, at the same time, do not require much data to build. These models could actually be more suitable, especially for large networks [15].

The discrete modeling framework for gene regulatory networks that has received the most attention is *Boolean networks*, which was introduced by Kauffman [3]. They have been used successfully in modeling gene regulatory and signaling networks; see, for example [16-18]. Many reverse engineering methods have been developed to infer such networks, see, for example [19,20].

For the purpose of better handling noisy data and the uncertainty in model selection, Boolean networks were extended to *probabilistic Boolean networks* (PBN) in [13,21,22]. A PBN is a Boolean network where each node i may possibly have more than one Boolean transition function, say $f_{i1}, \ldots, f_{it_i}$, where $t_i \geq 1$, and, to decide the future state of i, a function $f_j^{(i)}$ is chosen with probability p_{ij}, where $p_{i1} + \cdots + p_{it_i} = 1$. To be precise, to each node i in a PBN, the set $F_i = \{(f_{ij}, p_{ij})\}_{j=1,\ldots,t_i}$ of possible transition functions and their probabilities is

* Correspondence: edimit@clemson.edu
[1]Department of Mathematical Sciences, Clemson University, Clemson, SC 29634-0975, USA
Full list of author information is available at the end of the article

assigned. Notice that if $t_i = 1$ for all nodes in the network, then the PBN is just a Boolean network. As it is the case with Boolean networks, a PBN could be updated synchronously or asynchronously. However, throughout this article, we focus on synchronous PBN. Aspects of PBNs, and also asynchronous PBNs, have been studied in, for instance [23,24] and they have been applied to the modeling of gene regulatory networks in, for example, [25,26]. Furthermore, methods for inferring PBN have been developed in [27].

One disadvantage of Boolean models for gene regulatory networks is the limited number of states in which a gene can be. Indeed, although for a molecular biologist the state of a gene is usually discrete, it could be not only "expressed" and "not expressed" but also "over expressed," for example. There has thus been some consideration of more-than-binary discrete models in the Boolean network community. In the context of PBNs, generalizations of Boolean networks for ternary gene expression have been proposed in [28-31]. In addition, in [32] a ternary model has been considered as a preliminary stage for a Boolean one.

Other discrete multistate modeling frameworks have been developed too. Logical models [8] and K-bounded Petri nets [6,33] are two multistate modeling frameworks that have been used for modeling gene regulatory networks. A natural generalization of Boolean networks to multistate networks are the so-called polynomial dynamical systems (also known as algebraic models), which were introduced in [34]. In an algebraic model, the set of possible states of each node is a finite set, and once the mathematical structure of finite fields is imposed on that set, the transition function of each node is necessarily a polynomial. As this framework is rooted in computational algebra and algebraic geometry, results from these fields are used for the reverse engineering of dynamic and static biological networks [34-37], as well as for analyzing model dynamics [34,38], which usually is a challenge. Furthermore, in [39], it was shown that logical models and K-bounded Petri nets can be viewed as polynomial dynamical systems and algorithms for their translation into algebraic models were provided which facilitates the analysis of their dynamics.

In this article, we first introduce a stochastic generalization of polynomial dynamical systems, namely, *probabilistic polynomial dynamical systems*, which is also a generalization of the above-mentioned probabilistic Boolean networks to multistate models. Then, using this framework, we present a novel method for the reverse engineering of multistate gene regulatory networks from limited and noisy data. The novelty of our approach is two-fold. First, the stochastic model we construct is based on all minimal models in the model space and second, the probabilities assigned to the minimal models

are based on an algebraic partition, called Gröbner fan, of the models space, which provides an algorithmic and algebraic method for the construction of such stochastic models.

In the next section, we present our method for the reverse engineering of gene regulatory networks as probabilistic polynomial dynamical systems. Then we demonstrate this method using the yeast cell cycle model in [17], as well as the synthetic network of the yeast cell cycle in [40].

Methods
Probabilistic polynomial dynamical systems
Laubenbacher and Stigler [34] proposed a modeling approach that describes a regulatory network on n genes as a deterministic polynomial dynamical system (PDS), i.e., a polynomial function $(f_1, ..., f_n): \mathbf{K}^n \to \mathbf{K}^n$, where $\mathbf{K}$ is a finite field. (F is just a Boolean network when $\mathbf{K} = \{0, 1\}$.) Indeed, when $\mathbf{K}$ is a finite field, any function $F : \mathbf{K}^n \to \mathbf{K}^n$ is a polynomial function, i.e., F can be described as $(f_1, ..., f_n)$ where, for all i, $f_i : k^n \to k$ is a polynomial (see Appendix 1). This shows that PDSs are a suitable modeling framework naturally generalizing Boolean networks. We expand this framework to include stochastic models as follows.
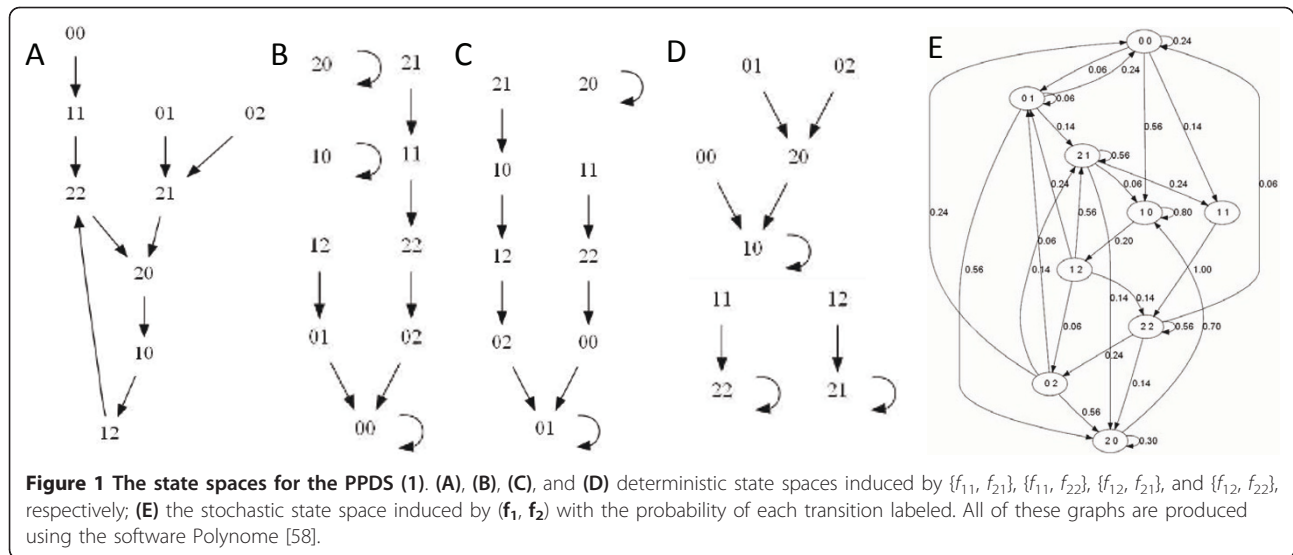
A *probabilistic polynomial dynamical system* (PPDS) on n nodes is a polynomial function $(\mathbf{f_1}, ..., \mathbf{f_n}) : \mathbf{K}^n \to \mathbf{K}^n$ where $\mathbf{K}$ is the set of possible sates of each node, and, for each node i, $\mathbf{f_i} = \{(f_{i1}, p_{i1}), (f_{i2}, p_{i2}), . . . , (f_{it_i}, p_{it_i})\}$ is the set of functions that could be used to determine the future state of node i with probabilities $p_{ij}, \sum_{j=1}^{t_i} p_{jt_j} = 1$. Given any state $\mathbf{x} = (x_1, ..., x_n)$ in state space $\mathbf{K}^n$ of the system, the next state is determined as follows. For each node i, a local function f_{ij} is selected from $\mathbf{f_i}$ with probability p_{ij}, and is used to compute the next state of node i, say y_i. The set of all such transitions $\mathbf{x} \to \mathbf{y}$ forms a directed graph, called the *state space* or *phase space*, on the vertex set $\mathbf{K}^n$. For example, the PPDS $(\mathbf{f_1}, \mathbf{f_2}) : \mathbf{F}_3^2 \to \mathbf{F}_3^2$, where

$$\begin{aligned}
\mathbf{f_1} &= \{(x_2^2 + 1, 0.7), (x_1 x_2 + x_1, 0.3)\}, \\
\mathbf{f_2} &= \{(x_1 + 1, 0.2), (2x_1 x_2, 0.8)\},
\end{aligned} \tag{1}$$

and $\mathbf{F}_3 = \{0, 1, 2\}$ is the finite field of three elements, is a PPDS whose state space (Figure 1E) has nine states. Notice that the state space of a PPDS is the union of the state spaces of all associated deterministic systems. In this example, as each node has two functions, there are four deterministic systems and their state spaces are in Figure 1A,B,C,D. For example, the state space of

$$\begin{aligned}
f_1 &= x_2^2 + 1 \\
f_2 &= x_1 + 1
\end{aligned}$$

is in Figure 1A.

Figure 1 The state spaces for the PPDS (1). **(A)**, **(B)**, **(C)**, and **(D)** deterministic state spaces induced by {f_{11}, f_{21}}, {f_{11}, f_{22}}, {f_{12}, f_{21}}, and {f_{12}, f_{22}}, respectively; **(E)** the stochastic state space induced by (**f**$_1$, **f**$_2$) with the probability of each transition labeled. All of these graphs are produced using the software Polynome [58].

Reverse engineering PDSs

Laubenbacher and Stigler's reverse-engineering method [34] first constructs the set of all PDSs that fit the given discretized data, which we call here the model space, and then uses a minimality criterion to select one system from the model space. A unique feature of their method is that the model space is presented as an algebraic object. Their algorithm is summarized here as Algorithm 2.1.

Unless all state transitions of the system are specified, there will be more than one network that fits the given data set. Since this much information is hardly ever available in practice, any reverse-engineering method usually identifies one network model according to a pre-specified criterion, and different methods typically identify different models. In [34], first the set of all models is computed and then a particular one $f = (f_1, ... f_n)$ is chosen that satisfies the following property: For each node i, the transition function f_i is *minimal* in the sense that there is no non-zero polynomial $g \in k[x_1 ... x_n]$ such that $f_i = h + g$ and g is identically equal to zero on the given time points. This criterion for model selection is analogous to excluding the terms of f_i that vanish on the data. The advantage of the polynomial modeling framework is that there is a well-developed algorithmic theory that provides mathematical tools for generating the model space as well as identifying the minimal models.

Algorithm 2.1 **Reverse engineering of PDSs**.

Input: A discrete time series of network states

$$\mathbf{s}_1 = (s_{11}, \dots, s_{n1}), \dots, \mathbf{s}_m = (s_{1m}, \dots, s_{nm}) \in \mathbf{K}^n.$$

Output: All minimal PDS's $(f_1, ..., f_n)$ such that the *coordinate polynomials* $f_i \in k[x_1, ..., x_n]$ satisfy $f_i(\mathbf{s}_j) = s_{i,j+1}$ for all $i = 1, ..., n$ and $j = 1, ..., m - 1$, and f_i does not contain any term that vanish on the time series.

Step 1: Compute a PDS $f_0 : \mathbf{K}^n \rightarrow \mathbf{K}^n$ that fits the data. There are several methods to do this, Lagrange interpolation being one of them.

Step 2: Compute the collection I of all polynomials that vanish on the data. Notice that if two polynomials $f_i, g_i \in k[x_1, ..., x_n]$ satisfy $f_i(\mathbf{s}_j) = s_{i,j+1} = g_i(\mathbf{s}_j)$, 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. Those functions form an algebraic object called the *ideal of points* and can be computed algorithmically.

Step 3: Reduce $f_0 = (f_1, ..., f_n)$ found in Step 1 modulo the ideal I. That is, write each f_i as $f_i = g + h$ with $h \in I$ and g being minimal in the sense that it cannot be further decomposed into $g = g' + h'$ with $h' \in I$. In other words, h represents the part of f_i that lies in I and is, therefore, identically equal to 0 on the given time series.

Algorithm 2.1 efficiently generates the set of all minimal PDS models that fit the data. However, identifying a single model may hardly be possible. There is a problem originating from Step 2 of Algorithm 2.1: finding all polynomials that vanish on a set of points. This is equivalent to computing the ideal of these points and computation of an ideal of points boils down to intersection of ideals. There is a well-known consequence of the Buchberger algorithm [41] for their computation. The output of the algorithm is a finite set of polynomials $\{g_1, ..., g_s\} \subset k[x_1, ..., x_n]$, called a *Gröbner basis* (for details see Appendix 2.1) that generates the ideal of vanishing on the data polynomials I:

$$I = \langle g_1, \dots, g_s \rangle = \left\{ \sum_{i=1}^{s} h_i g_i : h_i \in k[x_1, \dots, x_n] \right\}.$$

The Gröbner basis, however, is not unique and its computation depends on the way the polynomial terms are ordered, called *monomial ordering* (Definition 2.1). The reason is that the remainder of polynomial division in polynomial rings in more than one variable is not unique and depends on the way the monomials are ordered. In contrast, this is not an issue in $k[x]$ (a polynomial ring in one variable) where the monomials are ordered by degree: $... \succ x^{m+1} \succ x^m \succ ... x^2 \succ x \succ 1$. However, whenever there is more than one variable, there is more than one choice for ordering the monomials (*e.g.*, $x^2 \succ xy$ and $xy \succ x^2$ are both possible) and thus the possibility of obtaining several different Gröbner bases. Consequently, the PDS model generated in Step 3 also depends on the choice of monomial ordering, as Example 3.1 illustrates.

Since different monomial orderings may give rise to different polynomial models, considering only one arbitrarily chosen monomial ordering is not sufficient. 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 orderings, however, grows rapidly with the number of variables n and can be as large as $n^2n!$ [42] and hence considering all of them is computationally challenging. An alternative approach presented in [43] generates a collection of polynomial models from a fixed number of orderings (all graded reverse lexicographic) with random variable orderings and computes a consensus model using a game-theoretic method. While it is reasonable to try to avoid considering all monomial orderings, restricting oneself to variable orderings within a fixed monomial ordering will very likely miss a large number of PDS models that fit the data. Fortunately, the correspondence between Gröbner bases and monomial orderings is one-to-many. In [35], we presented a method which guarantees that no PDS model fitting the data is overlooked. Like [43], we avoided checking all possible monomial orderings but instead identified only those that produce *distinct* PDS models. The method is based on the combinatorial structure known as the *Gröbner fan* of a polynomial ideal which we discuss in more detail in Appendix 2.4. The Gröbner fan of an ideal I [44] is a polyhedral complex of cones with the property that every point encodes a monomial ordering. The cones are in bijective correspondence with the distinct Gröbner bases of I. (To be precise, the correspondence is to the marked reduced Gröbner bases of I). Therefore, it is sufficient to select exactly one monomial ordering per cone and, ignoring the rest of the orderings, still guarantee that all distinct models are generated. In addition, the relative number

of monomial orderings under which a particular PDS model is generated provides an insight into the likelihood that the model is a good representation of the system; for details on this idea see Appendix 3. An excellent implementation of an algorithm for computing the Gröbner fan of an ideal is the software package *Gfan* [45].

Algorithm for PPDS computation

We propose the following algorithm for the reverse engineering of gene regulatory networks as PPDS models from time series of discrete data. The resulting PPDS consists of all possible reduced PDS models that fit the data. The probability that we assign to each model is proportional to the relative volume of the Gröbner cone that produced that model. See Appendix 3 for assumptions and example.

Algorithm 3.1 **Reverse engineering of PPDSs**.

Input: A discrete time series of a gene regulatory network on n nodes $x_1, ..., x_n$: $S = \{(s_{11}, ..., s_{n1}), ..., (s_{1m}, ..., s_{nm})\} \subseteq \mathbf{K}^n$, where $\mathbf{K}$ is a finite field.

Output: A probabilistic PDS model F, which is a list of all possible reduced local polynomials for each $x_1, ..., x_n$, together with their corresponding probabilities.

Step 1: Compute a particular PDS $F_0 : \mathbf{K}^n \rightarrow \mathbf{K}^n$ that fits S.

Step 2: Compute the ideal I of polynomials that vanish on S.

Step 3: Compute the Gröbner fan $\mathcal{G}$ of the ideal I and the relative sizes of its cones, $c_1, ..., c_s$ (with $c_1 + ... + c_s = 1$).

Step 4: Select one (any) monomial ordering from each cone, $\prec_1, ..., \prec_s$. For each $i = 1, ..., s$, reduce F_0 modulo I using a Gröbner basis computed with respect to $\prec_i$. Let the reduced PDS's be $F_1 = \{f_{11}, f_{12}, ..., f_{1t}\}, ..., F_s = \{f_{s1}, f_{s2}, ..., f_{st}\}$ and adding the cone sizes redefine them as $F_i = \{(f_{i1}, c_1), (f_{i2}, c_2), ..., (f_{it}, c_t)\}$.

Step 5: Construct the list $F = \{\{(f_{11}, c_1), (f_{21}, c_2), ..., (f_{s1}, c_s)\}, ..., \{(f_{1t}, c_1), (f_{2t}, c_2), ..., (f_{st}, c_s)\}\}$. For a fixed i, if $f_{ji} = f_{ki}$ for some j and k, then "merge" the two local polynomials by adding their corresponding probabilities: $(f_{ji}, c_j + c_k)$.

Algorithm 3.1 guarantees that all distinct minimal PDS models will be generated. However, this comes at the expense of having to compute the entire Gröbner fan of the ideal of points. For small networks the computation of the fan is feasible but as the number of network nodes increases, the complexity of the Gröbner fan computation becomes prohibitive [46]. As mentioned earlier, the correspondence between PDS models and Gröbner bases is one-to-many. Therefore, computing the entire Gröbner fan of the ideal of vanishing polynomials is excessive and instead a finite subset of points from the fan should be sufficient. This finite

subset needs to be carefully selected if we want it to reflect the structure of the entire Gröbner fan. Since we want to rank the dependencies according to their strength, the number of points (weight vectors) we select from a Gröbner cone should correspond to the relative size of this cone with respect to the other cones. That is, we want to sample from the Gröbner fan uniformly, so that the relative frequency with which we select term orders from the fan is approximately equal to the relative sizes of its cones. We do this through random sampling of the Gröbner fan of the ideal of points as in [47]. If the number of points is sufficiently large, their distribution approximately reflects the relative size of the Gröbner cones. The number of points is determined using a t test for proportion. Consequently, steps 3 and 4 of Algorithm 3.1 have to be modified in such a way that direct computation of the Gröbner fan is avoided.

Step **3'**: Select vectors $\mathbf{w}_1$, ..., $\mathbf{w}_s$ of length n, with s large, in such a way that every (nonnegative integer) vector in the Gröbner fan of I has equal probability of being chosen.

Step **4'**: For each i = 1, ..., s, use $\mathbf{w}_i$ to define a monomial ordering $\prec_i$ and reduce F_0 modulo I using a Gröbner basis computed with respect to $\prec_i$.

Examples and results
Reverse engineering of the yeast cell cycle
We applied the PPDS method to the reverse engineering of the gene regulatory network of the cell cycle in *Saccharomyces cerevisiae* starting from a data set generated from the well-known discrete model suggested by Li et al. [17]. The cell cycle is the process of cell growth and division and consists of four phases. The cell cycle in *S. cerevisiae* has been extensively studied and about 800 genes are known to participate in the process. It is believed, however, that the number of key regulators is much smaller and, based on an extensive literature review [17] constructed a Boolean network on 11 distinct nodes: Cln3, MBF, SBF, Cln1, 2, Cdh1, Swi5, Cdc20 and Cdc14, Clb5, 6, Sic1, Clb1, 2, Mcm1/SFF. For the network dynamics, a threshold function is assigned to each node in the network according to (2), where a_{ij} represents the weight of effect of node j on node i.

$$S_i(t+1) = \begin{cases} 1, & \sum_j a_{ij}S_j(t) > 0 \\ 0 & \sum_j a_{ij}S_j(t) < 0 \\ S_i(t) \sum_j a_{ij}S_j(t) = 0 \end{cases} \quad (2)$$

This model captures the known features of the cell cycle dynamics. Furthermore, the trajectory of the known cell cycle sequence is stable and attracting, as its size is 1764 out of the total of 2048 states. The remaining states are distributed into 6 very small trajectories. Each of these trajectories converges to a steady state as well.

We used as input to our Algorithm 3.1 54 input-output transitions, four of which are steady states (see Table 1). Our reverse engineering algorithm generated the PPDS (6). The state space of this system consists of 14 connected components, where each component ends in a steady state. The built-in four steady states belong to components of sizes very close to those of the original system. In addition, the other three steady states in the original system were also recovered. These results are summarized in Table 2. The seven steady states of our model, which are not in the original system, with one exception belong to very small components (less than 30 points).

Further, we assessed the quality of the dependency graph of the inferred model using three standard network measures: *positive predictive value*, PPV = TP/(TP + FP) = 0.83, *specificity* , Sp = TN/(TN + FP) = 0.94, and *sensitivity*, Se = TP/(TP + FN) = 0.69, where TP and TN are the numbers of true positive and negative interactions, respectively, and FP and FN are the numbers of false positive and false negative interactions, respectively, weighted by the corresponding probabilities given after every polynomial in (6). The high values of the three measures indicate that the proposed method is not only capable of capturing the dynamic behavior of the system but also its static wiring network.

Comparison to other methods
We also performed a comparison of our algorithm to several other reverse engineering methods. In [40], Cantone et al. built in *S. cerevisiae* a synthetic network for in vivo "benchmarking" of reverse-engineering and modeling approaches. The network in Figure 2 is composed of five genes (CBF1, GAL4, SWI5, GAL80, and ASH1) that regulate each other through a variety of regulatory interactions. The mathematical model of the network is based on nonlinear differential equations obtained from standard mass-balance kinetic laws. Time series and steady-state expression data were measured after multiple perturbations. In particular, they performed perturbation experiments by shifting cells from glucose to galactose ("switch-on" experiments) and from galactose to glucose ("switch-off" experiments). The synthetic network was then used to assess the ability of experimental and computational approaches to infer regulatory interactions from gene expression data. Four published algorithms were selected as representatives of reverse-engineering approaches: BANJO (Bayesian networks) [48], NIR and TSNI (ordinary differential equations)

Table 1 The 54 state transitions used for generating the PPDS model (6), each represented by a pair of input-output states

Cln3	MBF	SBF	Cln1, 2	Cdh1	Swi5	Cdc20 and Cdc 14	Clb5, 6	Sic1	Clb1, 2	Mcm1/ SFF	Cln3	MBF	SBF	Cln1, 2	Cdh1	Swi5	Cdc20 and Cdc14	Clb5, 6	Sic1	Clb1, 2	Mcm1/ SFF

Table 1 The 54 state transitions used for generating the PPDS model (6), each represented by a pair of input-output states *(Continued)*

The four fixed points are listed first in bold. The transitions are generated from the 11 variable model of Li et al. [17].

Table 2 Comparison of the steady states of model (2) and those of the probabilistic PDS (6) built via our reverse engineering method using the data set in Table 1 generated from model (2)

Fixed point	Is it input?	Original system component size	Reverse engineered component size
1	No	1, 764	1, 015
2	Yes	151	70
3	No	109	9
4	No	9	9
5	Yes	7	7
6	Yes	9	5
7	Yes	1	1

Table 3 PPV, positive predictive value and Se, sensitivity of the reverse-engineering approaches NIR, TSNI, BANJO, ARACNE, and PPDS when applied to data generated from the synthetic network in [40]

	Switch-on		Swith-off	
	PPV	Se	PPV	Se
NIR and TSNI	0.80	0.50	0.6	0.38
BANJO	*	*	0.6	0.38
ARACNE	*	*	*	*
PPDS	0.57	0.50	0.75	0.38

The symbol * stands for "worse than random."

[49,50], and ARACNE (information theory) [51]. These methods were assessed based on their positive predictive value (PPV) and sensitivity (Se). In order to test the significance of the algorithms, the "random" performance was computed, which refers to the expected performance of an algorithm that randomly assigns edges between a pair of genes. For example, for a fully connected network, the random algorithm would have a 100% accuracy (PPV = 1) for all the levels of sensitivity (as any pair of genes is connected in the real network). For the net-work in Figure 2, the expected PPV for a random guess of directed interactions among genes is PPV = 0.40, so any value higher than 0.4 will be significant. (In the case of undirected interactions, the random guess has PPV = 0.70.)

Using the same data sets, which we discretized into three states applying the algorithm in [52], our method (PPDS) performed well when compared to the best method (the ordinary differential equations approach TSNI) according to [40]. A summary is given in Table 3. Notice that although the PPV value of PPDS on the switch-on data is lower than that of TSNI, it is still well above 0.40 and thus it is better than random.

Conclusion

Gene regulatory networks are structured as inter-connected entities and their complex nature is inherently stochastic. The framework of stochastic 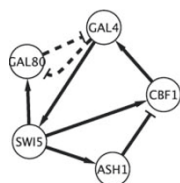dynamical systems is natural for modeling and analyzing such networks. We focused on PPDSs due to their applicability to limited and possibly noisy data. Within this modeling framework, we developed a systematic method based on combinatorial topology, algebraic geometry, and statistics for the reverse engineering of the dynamics, as well as the gene dependencies, in biochemical regulatory networks from experimental data. The algorithm can handle large regulatory networks and hence is applicable to many networks of interest. The constructed models are comprised of minimal polynomials according to the definition in [34]. We plan to explore the use of other types of biologically relevant functions, such as nested canalyzing functions [53]. An algorithm for the inference of deterministic nested Boolean canalyzing networks has recently been presented (F Hinkelmann, A Jarrah: Inferring biologically relevant models: nested canalyzing functions, submitted). Combining this with our algorithm here will provide a systematic method for the reverse engineering of gene regulatory networks as probabilistic Boolean nested canalyzing networks.

Appendices

1 Polynomial dynamical systems

Definition 1.1 Let X be a finite set. A **finite dynamical system** of dimension n is a function $F = (f_1, ..., f_n) : X^n \rightarrow X^n$ with $f_i : X^n \rightarrow X$.

By requiring that the cardinality of the set X be a power of a prime number, one can impose on X the structure of a *finite field*. This structure determines the only type of functions f_i that need to be considered. The following theorem from [54] characterizes functions over finite fields.

Theorem 1.1 *Let k be a finite field. Then every function $f : k^n \rightarrow k$ is a polynomial of degree at most n.*

Therefore, over a finite field, polynomials are the appropriate modeling framework rather than a constraining assumption.

Definition 1.2 If the set X for a finite field, then any function $F : X \rightarrow X$ is called a **polynomial dynamical system** (PDS).

Figure 2 The five gene synthetic networks in *S. cerevisiae* built by Cantone et al. [40].

Definition 1.3 A **probabilistic polynomial dynamical system** (PPDS) on n nodes $(\mathbf{f_1}, ..., \mathbf{f_n}) : \mathbf{K}^n \rightarrow \mathbf{K}^n$ with parallel update order consists of n sets of local functions and their associated probabilities such that $\mathbf{f_i} = \{(f_{i1}, p_{i1}), (f_{i2}, p_{i2}), \ldots, (f_{it_i}, p_{it_i})\}$ is the set of local functions that determine the dynamics of node i and $\sum_{j=1}^{t_i} p_{jt_j} = 1$. In order to determine each transition in the state space of the system, $(x_1, ..., x_n) \rightarrow (y_1, ..., y_n)$, for each node i a local function f_{ij} is selected from $\mathbf{f_i}$ with probability p_{ij}.

As an example, see (1).

2 Concepts from commutative algebra and algebraic geometry [55]

2.1 Gröbner bases

A polynomial in $k[x_1, ..., x_n]$ is a linear combination of monomials of the form $x^{\alpha} = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ over k, where α is the n-tuple exponent $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{Z}_{\geq 0}^n$. For many purposes, such as polynomial division, it is necessary to arrange the terms in a polynomial unambiguously in some order. Unlike polynomials in one variable, there are more than one way of ordering the terms (monomials) of multivariate polynomials. Any ordering of the monomials must be a *total* ordering, i.e., for every pair of monomials x^{α} and x^{β}, exactly one of the following must be true: $x^{\alpha} \prec x^{\beta}$, $x^{\alpha} = x^{\beta}$, $x^{\alpha} \succ x^{\beta}$. Taking into account the properties of the polynomial sum and product operations, the following definition emerges.

Definition 2.1 A **monomial ordering** on $k[x_1, ..., x_n]$ is any relation $\succ$ on $\mathbb{Z}_{\geq 0}^n$ satisfying:

1. $\succ$ is a total ordering on $\mathbb{Z}_{\geq 0}^n$.
2. If $\alpha \succ \beta$ and $\gamma \in \mathbb{Z}_{\geq 0}^n$, then $\alpha + \gamma \succ \beta + \gamma$.
3. $\succ$ is a well-ordering on $\mathbb{Z}_{\geq 0}^n$, i.e., every nonempty subset of $\mathbb{Z}_{\geq 0}^n$ has a smallest element under $\succ$.

A monomial ordering can also be defined by a weight vector $\omega = (\omega_1, ..., \omega_n)$ in $\mathbb{Z}_{\geq 0}^n$. We require that ω have nonnegative coordinates in order for 1 to always be the smallest monomial. Fix a monomial ordering $\succ_{\sigma}$, such as $\succ_{lex}$. Then, for $\alpha, \beta \in \mathbb{Z}_{\geq 0}^n$, define $\alpha \succ_{\omega, \sigma} \beta$ if and only if $\omega \cdot \alpha \succ \omega \cdot \beta$, or $\omega \cdot \alpha = \omega \cdot \beta$ and $\alpha \succ_{\sigma} \beta$.

Ideal membership problem Another problem with multivariate polynomial division is that when dividing a given polynomial into more than one polynomials, the outcome may depend on the order in which the division is carried out. Let $f, g_1, ..., g_m \in k[x_1, ..., x_n]$ be polynomials in the variables $x_1, ..., x_n$. The so-called *ideal membership problem* is to determine whether there are polynomials $h_1, ..., h_m \in k[x_1, ..., x_n]$ such that $f = \sum_{i=1}^{m} h_i g_i$. To state this in the language of abstract algebra, we define $I = \langle g_1, ..., g_m \rangle := \{\sum h_i g_i \mid h_1, ..., h_m \in k[x_1, ..., x_n]\}$. The polynomials in I form a so-called *ideal* in $k[x_1, ..., x_n]$, since I is closed under addition and multiplication by any polynomial in $k[x_1, ..., x_n]$, and I is generated by the set $\{g_1, ..., g_m\}$. The ideal membership problem asks if f is an element of I. In general, even under a fixed monomial ordering, the order in which f is divided by the generating polynomials f_i affects the remainder $r_{\{f_i\}}(f)$. Therefore, $r_{\{f_i\}}(f) \neq 0$ does not imply $f \notin I$. Moreover, the generating set $\{f_1, ..., f_m\}$ of the ideal I is not unique but a special generating set $\mathcal{G} = \{g_1, \ldots, g_t\}$ can be selected so that the remainder of polynomial division of f by the polynomials in $\mathcal{G}$ performed in any order is zero if and only if f lies in I: $r_{\mathcal{G}}(f) = 0 \Leftrightarrow f \in I$. A generating set with this property is called a *Gröbner basis* and its precise definition will be given in Definition 2.3. Here we point out that Gröbner bases provide an algorithmic solution to the ideal membership problem and the Buchberger algorithm [41] is designed to compute a Gröbner basis for any ideal other than $\{0\}$ and a fixed monomial ordering.

2.2 Monomial Ideals

Gröbner bases are a key concept in computational algebra. Their theory reduces questions about systems of polynomial equations to the combinatorial study of *monomial ideals*.

Definition 2.2 An ideal $I \subset k[x_1, ..., x_n]$ is a **monomial ideal** if I is generated by monomials, i.e., there is a subset $A \subset \mathbb{Z}_{\geq 0}^n$ such that $I = \langle x^{\alpha} \mid \alpha \in A \rangle$, i.e., consists of all polynomials which are finite sums of the form $\sum_{\alpha \in A} h_{\alpha} x^{\alpha}$, where $h_{\alpha} \in k[x_1, ..., x_n]$.

A special kind of monomial ideal is the *initial ideal* of an ideal $I \neq \{0\}$ for a fixed monomial ordering. It is the ideal generated by the set of initial monomials (under the specified ordering) of the polynomials of I: $in(I) = \langle in(f) \mid f \in I \rangle$. The monomials which do not lie in $in(I)$ are called *standard monomials*.

Definition 2.3 Fix a monomial ordering. A finite subset $\mathcal{G}$ of an ideal I is a **Gröbner basis** if $in(I) = \langle in(g) \mid g \in \mathcal{G} \rangle$.

A Gröbner basis for an ideal may not be unique. If we also require that for any two distinct elements $g, g' \in \mathcal{G}$, no term of g' is divisible by $in(g)$, such a Gröbner basis is called *reduced* and is unique for an ideal and a monomial ordering, provided the coefficient of $in(g)$ in g is 1 for each $g \in \mathcal{G}$

2.3 Ideals of points

Given a set of points, it is often necessary to find all the polynomials that vanish on it. Such a set of polynomials forms an ideal called the *ideal of points* defined as follows.

Definition 2.4 Let $V = \{p_1, ..., p_m\}$, where $p_i = (a_{i1}, ..., a_{in}) \in k^n$. Then we set

$$\mathcal{I}(V) = \{f \in k[x_1, \ldots, x_n] \mid f(a_1, \ldots, a_n) = 0 \text{ for all } (a_1, \ldots, a_n) \in V\}.$$

It can be shown that $\mathcal{I}(V)$ is an ideal of $k[x_1, ..., x_n]$. It is called the *ideal of points* in V.

2.4 The Gröbner fan of an ideal

A combinatorial structure that contains information about the initial ideals of an ideal is the *Gröbner fan* of an ideal. It is a polyhedral complex of cones, each corresponding to an initial ideal, which, as follows from Definition 2.3, is in a one-to-one correspondence with the *marked* reduced Gröbner bases (the initial term of each generating polynomial being distinguished) of the ideal. A brief introduction to the the Gröbner fan folllows. For details see, for example [44].

A polynomial ideal has only a finite number of different reduced Gröbner bases. Informally, the reason is that most of the monomial orderings only differ in high degree and the Buchberger algorithm for Gröbner basis computation does not "see" the difference among them. However, they may vary greatly in number of polynomials and "shape". In order to classify them, we first present a convenient way to define monomial orderings using matrices [56]. Again, we think of a polynomial in $k[x_1, ..., x_n]$ as a linear combination of monomials of the form $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ over k, where α is the n-tuple exponent $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{Z}_{\geq 0}^n$.

Definition 2.5 Let $\omega = (\omega_1, ..., \omega_n)$ be a vector with real coefficients. We can define an ordering $\succ_\omega$ the elements of $\mathbb{Z}_{\geq 0}^n$ by $\alpha \succ_\omega \beta$ if and only if $\alpha \cdot \omega > \beta \cdot \omega$, componentwise.

Definition 2.6 Let $\mathcal{G} = \{g_1, \ldots, g_r\}$ be a marked reduced Gröbner basis for an ideal I. Write each polynomial of the basis as $gi = x^{\alpha_i} + \sum_\beta c_{i,\beta} x^\beta$ where x^{α_i} is the initial term in g_i. The **cone** of $\mathcal{G}$ is $C_\mathcal{G} = \{\omega \in \mathbb{R}_{\geq 0}^n : \alpha_i \cdot \omega \geq \beta \cdot \omega$ for all i, β with $c_{i,\beta} \neq 0\}$

The collection of all the cones for a given ideal is the **Gröbner fan** of that ideal. The cones are in bijection with the marked reduced Gröbner bases of the ideal. Since reducing a polynomial modulo an ideal I, as the reverse engineering algorithm requires in Step 3, can have at most as many outputs as the number of marked reduced Gröbner bases, it follows that the Gröbner fan contains information about *all* Gröbner bases (and thus all monomial orderings) that need to be considered in the process of model selection. There are algorithms based on the Gröbner fan that enumerate all marked reduced Gröbner bases of a polynomial ideal [45].

3 Reverse engineering of PPDSs

Suppose we have time series data from a gene regulatory network on n genes represented by variables $x_1, ..., x_n$. Let $f = (f_1, \ldots f_n)$ be any polynomial system that fits the data, generated using, for instance, Lagrange interpolation, and suppose that variable x_i appears in at least one monomial (with a nonzero coefficient) of polynomial f_j.

Then it follows that variable x_i has effect on variable x_j whose behavior is determined by f_j. The directed graph on $\{x_1, ..., x_n\}$ representing these dependencies is called the *dependency graph* of f. For example, let $f = (f_1, f_2) \in \mathbb{F}_2^2[x_1, x_2]$ where

$$
\begin{aligned}
f_1 &= x_1 x_2 \\
f_2 &= x_1 + 1
\end{aligned}
\tag{3}
$$

Then x_1 depends on both x_1 and x_2, while x_2 depends only on x_1.

While inferring the dependency graph from a PDS model is straightforward, identifying that single model may hardly be possible. There is a problem originating from the algorithm proposed in [34]: finding all polynomials that vanish on a set of points. This is equivalent to computing the ideal of these points and computation of an ideal of points boils down to intersection of ideals of polynomials vanishing on one point. There is a well-known consequence of the Buchberger algorithm, originally presented in [57] MISSING, for their computation. The output of the algorithm is a Gröbner basis $\{g_1, ..., g_s\} \subset k[x_1, ..., x_n]$ that generates the ideal of vanishing polynomials: $I = \langle g_1, \ldots, g_s \rangle = \sum_{i=1}^s h_i g_i$, where $h_i \in k[x_1, ..., x_n]$. The Gröbner basis, however, is not unique, as it was discussed in 2.1, and its computation depends on the choice of monomial ordering.

Example **3.1** Consider a network of 3 genes x_1, x_2, and x_3. Suppose we have the following time series of network states in $\mathbb{F}_3^3$: $\mathbf{s}_1 = (2, 1, 0)$, $\mathbf{s}_2 = (1, 2, 0)$, $\mathbf{s}_3 = (2, 1, 1)$, $\mathbf{s}_4 = (0, 0, 1)$.

Depending on the selection of monomial ordering, the algorithm of [34] will generate one of the two polynomial models:

$$
\begin{array}{lll}
f_1 = x_2 - x_3 & & f_1 = -x_1 - x_3 \\
f_2 = -x_2 + x_3 & \text{or} & f_2 = x_1 + x_3 \\
f_3 = x_2 + x_3 - 1 & & f_3 = -x_1 + x_3 - 1
\end{array}
\tag{4}
$$

Notice that all three coordinate polynomials involve x_3 but depending on the monomial ordering, they also contain either x_1 or x_2. In fact, for the given time series $\mathbf{s}_1, ..., \mathbf{s}_4$, these are the only two distinct minimal (in the sense defined in [34]) PDS models that the algorithm generates. While it is not clear whether there is a dependence on x_1 or on x_2, one can be confident that, provided the data are representative of the network, x_3 has a definite impact on all three genes. We expand on this idea in the next section.

Clearly the monomial ordering selection affects not only the dependency graph of the model but also its dynamics which is represented by the model's state space. Let $\mathbf{p} = (1, 0, 0) \ \mathbb{F}_3^3$. In Example 3.1, starting at state $\mathbf{p}$, the first model will transition to state $(0, 0, 2)$, while the second's next state is $(2, 1, 1)$. All coordinates

of the output are different although they are generated by minimal PDS models that fit the experimental time series data. It is not clear how to select one out of, as it is often the case with real data, hundreds of possibilities. The main reason is that it is hard to relate monomial ordering to biology in any meaningful way. The solution we propose is instead of trying to single out one model, to use all minimal models (possibly with different probabilities) to generate the system dynamics. This means allowing stochastic behavior which is consistent with the experimental observations.

For Example 3.1, Algorithm 3.1 generates output $F = \{f_1, f_2, f_3\}$ given in (5). Its state space consists of one connected component (of size 27), and five fixed points, (0, 1, 1), (1, 1, 0), (2, 0, 1), (2, 1, 2), and (2, 2, 0), with stabilities (probability that for a given run of the simulation the point will be fixed) 0.12, 0.12, 0.13, 0.51, and 0.13, respectively. Figure 3 shows the state space graph of (5).

$$f_1 = \{(x_2 - x_3, 0.491), (-x_1 - x_3, 0.509)\}$$
$$f_2 = \{(-x_2 + x_3, 0.491), (x_1 + x_3, 0.509)\} \qquad (5)$$
$$f_3 = \{(x_2 + x_3 - 1, 0.491), (-x_1 + x_3 - 1, 0.509)\}$$

Relative size of the Gröbner cones and interaction strength. Using Algorithm 3.1 for reverse engineering of PPDSs is based upon the following assumptions that relate the network dependencies to the Gröbner fan of the ideal of polynomials that vanish on the network data.

1. *Minimal polynomials are an appropriate framework for the modeling of gene regulatory networks.* The minimal polynomials are only a subset of all polynomials that fit a given data set. In Example 3.1, both $f_1 = x_2 - x_3$ and $f_1' = x_2 - x_3 - x_1^2 - x_1 x_2$ fit the data and can be coordinate polynomials for x_1 but the latter is not minimal since $g = -x_1^2 - x_1 x_2$ is identically zero on the data and $f_1' = f_1 + g$. In that sense, g is not supported by the data and is meaningless for the purpose of reverse engineering.
2. *The strength of dependency of network node x_i on node x_j on the scale from 0 to 1 is proportional to the relative frequency with which x_j appears in x_i's minimal coordinate polynomials that fit the data.* In Example 3.1, both possible polynomials for f_1 involve x_3 which means that the behavior of x_1 cannot be described without involving x_3. It is reasonable then to conclude that the strength of the dependency of x_1 on x_3 is 1.
3. *The strength of dependency of x_i on x_j is proportional of the size of the portion of the Gröbner fan that corresponds to those coordinate polynomials for f_i that involve x_j.*

Each point of the Gröbner fan corresponds to a monomial ordering and all points within the same Gröbner cone produce the same Gröbner basis and thus the same minimal model. Typically, we do not know
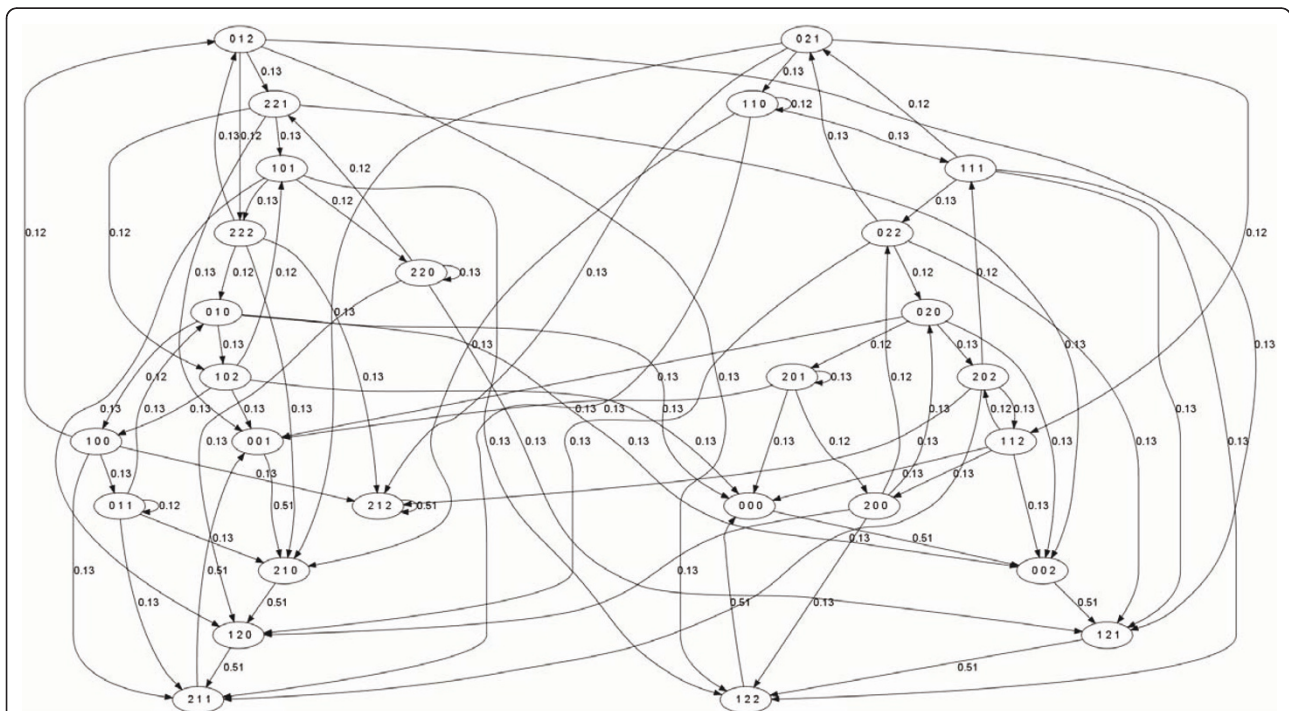

Figure 3 The state spaces of PPDS (5).

which monomial ordering(s) are more appropriate for a particular network and the best we can do is consider all of them and identify which ones are the most likely to represent the network dependencies. If there are two models, one corresponding to a larger portion of the Gröbner fan than the other, this means that there are more monomial orderings which produce the first model.

In Example 3.1, the Gröbner fan is three-dimensional since there are three variables and consists of two Gröbner cones of equal sizes (volumes). Each one of the two coordinate polynomials for f_1, for instance, corresponds to .5 of the total size of the Gröbner fan. Therefore, under our assumption, the strength of dependency of x_1 on x_2 is .5 and the probability that x_1 depends on itself is also .5.

4 Reverse engineering the yeast cell cycle

Based on the data generated from the model presented in [17] and given in Table 1, the reverse engineering method we proposed generated the PPDS (6). For each variable x_i, the set f_i consists of pairs of the form (F, p), where F is a polynomial in $\mathbb{F}_2[x_1, ..., x_{11}]$ and p is the probability with which F is used to determine x_i's next value at each iteration.

$$
\begin{aligned}
f_1 =& \{(0, 1)\} \\
f_2 =& \{(x_1x_3x_{10} + x_1 + x_2x_7 + x_2x_8 + x_2x_9, 0.064), \\
 & (x_1x_3 + x_1x_{10} + x_2x_{10} + x_2, 0.436), \\
 & (x_1x_{10} + x_1x_2 + x_1 + x_2x_{10} + x_2, 0.5)\} \\
f_3 =& \{(x_1x_3 + x_3x_{10} + x_1x_{10} + x_3 + x_1, 0.5), \\
 & (x_3x_{10} + x_1x_{10} + x_1x_2 + x_3, 0.5)\} \\
f_4 =& \{(x_3, 1)\} \\
f_5 =& \{(x_4x_7x_8 + x_4x_7x_{10} + x_7x_8x_{10} + x_4x_5x_8 \\
 & + x_4x_5x_{10} + x_5x_8x_{10} + x_4x_7 + x_7x_8 + x_7x_{10} \\
 & + x_5x_7 + x_4x_5 + x_5x_8 + x_5x_{10} + x_7 + x_5, 1)\} \\
f_6 =& \{(x_7x_{10} + x_7x_{11} + x_7 + x_{10}x_{11} + x_{11}, 1)\} \\
f_7 =& \{(x_{10}x_{11} + x_{10} + x_{11}, 1)\} \\
f_8 =& \{(x_1x_2x_{10} + x_1x_2 + x_7x_8 + x_8x_9 + x_2x_{10} + x_8, 0.212), \\
 & (x_1x_3x_{10} + x_1x_3 + x_1x_{10} + x_1 + x_7x_8 + x_8x_9 + x_2x_{10} + x_8, 0.256) \\
 & (x_7x_8 + x_8x_9 + x_2x_7 + x_2x_8 + x_2x_9 + x_8 + x_2, 0.532)\} \\
f_9 =& \{(x_4x_7x_8 + x_4x_7x_{10} + x_7x_8x_{10} + x_4x_7 + x_7x_8 \\
 & + x_7x_{10} + x_7x_9 + x_4x_9 + x_8x_9 + x_6x_7 + x_9x_{10} \\
 & + x_4x_6 + x_6x_8 + x_7 + x_6x_{10} + x_6x_9 + x_9 + x_6, 1)\} \\
f_{10} =& \{(x_5x_7x_8 + x_5x_7x_{10} + x_7x_8 + x_7x_{10} + x_8x_{10} \\
 & + x_5x_8 + x_8x_9 + x_5x_{10} + x_7x_{11} + x_9x_{10} + x_8x_{11} \\
 & + x_{10}x_{11} + x_5x_{11} + x_8 + x_9x_{11} + x_{10} + x_{11}, 1)\} \\
f_{11} =& \{(x_8x_{10} + x_8 + x_{10}, 1)\}
\end{aligned}
$$
(6)

Author details
[1]
Department of Mathematical Sciences, Clemson University, Clemson, SC 29634-0975, USA [2]
Sealy Center of Molecular Medicine, University of Texas Medical Branch, Galveston, TX 77550, USA [3]
Virginia Bioinformatics Institute, Virginia Tech, Blacksburg, VA 24061-0477, USA [4]
Department of Mathematics and Statistics, American University of Sharjah, Sharjah, UAE

References
1. T Akutsu, S Miyano, S Kuhara, Inferring qualitative relations in genetic networks and metabolic path-ways. Bioinformatics **16**(8), 727–734 (2000)
2. T Tian, K Burrage, Stochastic models for regulatory networks of the genetic toggle switch, PNAS **103**(22), 8372–8377 (2006)
3. SA Kauffman, Metabolic stability and epigenesis in randomly constructed genetic nets. J Theor Biol. **22**, 437–467 (1969)
4. J-P Comet, H Klaudel, S Liauzu, Modeling multi-valued genetic regulatory networks using high-level Petri nets, in *Proceedings of the International Conference on the Application and Theory of Petri Nets* ed. by Ciardo G, Darondeau P (Springer-Verlag, Berlin), pp. 208–227 (2005)
5. S Grunwald, A Speer, J Ackermann, I Koch, Petri net modelling of gene regulation of the duchenne muscular dystrophy. Biosystems **92**(2), 189–205 (2008)
6. L Steggles, R Banks, O Shaw, A Wipat, Qualitatively modelling and analysing genetic regulatory networks: a Petri net approach. Bioinformatics **23**, 336–343 (2007)
7. J Saez-Rodriguez, L Simeoni, J Lindquist, R Hemenway, U Bommhardt, B Arndt, U Haus, R Weismantel, E Gilles, S Klamt, B Schraven, A logical model provides insights into T cell receptor signaling. PLoS Comput Biol. **3**(8), e163 (2007)
8. R Thomas, R D'Ari, *Biological Feedback* (CRC Press, Boca Raton, 1989)
9. B Perrin, L Ralaivola, A Mazurie, S Bottani, J Mallet, F d'Alche-Buc, Gene networks inference using dynamic Bayesian networks, Bioinformatics **19**, 138–148 (2003)
10. N Radde, L Kaderali, Bayesian Inference of Gene Regulatory Networks Using Gene Expression Time Series Data. Proc BIRD'2007, 1–15 (2007)
11. R Albert, Network inference, analysis, and modeling in systems biology. Plant Cell **19**(11), 3327–3338 (2007)
12. C Sima, J Hua, S Jung, Inference of gene regulatory networks using time-series data: a survey. Curr Genomics, **10**(6) 416–429 (2009)
13. I Shmulevich, ER Dougherty, W Zhang, From Boolean to probabilistic Boolean networks as models of genetic regulatory networks. Proc IEEE **90**(11), 1778–1792 (2002)
14. T Tian, Stochastic models for inferring genetic regulation from microarray gene expression data. Biosystems **99**(3), 192–200 (2010)
15. S Bornholdt, Less is more in modeling large genetic networks. Science **310**(5747), 449–451 (2005)
16. R Albert, H Othmer, Newblock The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in *Drosophila melanogaster*. J Theor Biol. **223**, 1–18 (2003)
17. F Li, T Long, Y Lu, Q Ouyang, C Tang, The yeast cell-cycle network is robustly designed. PNAS **11**(14), 4781–4786 (2004)
18. A Samal, S Jain, The regulatory network of E. coli metabolism as a Boolean dynamical system exhibits both homeostasis and flexibility of response. BMC Syst Biol. **2**, Article 21 (2008)
19. GJ Hickman, TC Hodgman, Inference of gene regulatory networks using boolean-network inference methods, J Bioinformatics Comput Biol. **7**(6), 1013–1029 (2009)

20. W Liu, H Lähdesmäki, ER Dougherty, I Shmulevich, Inference of Boolean networks using sensitivity regularization. EURASIP J Bioinformatics Syst Biol. **2008**, Article ID 78054 (2008)
21. ER Dougherty, I Shmulevich, Mappings Between probabilistic Boolean networks. Signal Process. **83**(4), 799–809 (2003)
22. I Shmulevich, SA Kauffman, Activities and sensitivities in Boolean network models. Phys Rev Lett. **93**(4), 1–4 (2004)
23. C Gershenson, Introduction to random Boolean networks. in *Workshop and Tutorial Proceedings, Ninth International Conference on the Simulation and Synthesis of Living Systems (ALife IX)*, ed. by Bedau M, Husbands P, Hutton T, Kumar S, Suzuki H, pp. 160–173 (2004)
24. MT Matache, J Heidel, Asynchronous random Boolean network model based on elementary cellular automata rule 126. Phys Rev E, **71**, 1–13 (2005)
25. E Balleza, E Alvarez-Buylla, A Chaos, SA Kauffman, I Shmulevich, M Aldana, Critical dynamics in genetic regulatory networks: examples from four kingdoms. PLoS ONE **3**(6), e2456 (2008)
26. I Shmulevich, I Gluhovsky, R Hashimoto, ER Dougherty, W Zhang, Steady-state analysis of genetic regulatory networks modeled by probabilistic Boolean networks, Comp Funct Genomics, **4**(6), 601–608 (2003)
27. S Marshall, L Yu, Y Xiao, E Dougherty, Inference of a probabilistic Boolean network from a single observed temporal sequence. EURASIP J Bioinformatics Syst Biol. (2007). doi: 10.1155/2007/32454
28. A Datta, R Pal, ER Dougherty, Intervention in probabilistic gene regulatory networks. Curr Bioinformatics **1**, 167–184 (2006)
29. B Faryabi, G Vahedi, A Datta, J-F Chamberland, ER Dougherty, Recent advances in intervention in markovian regulatory networks. Curr Genomics **10**(7), 463–477 (2009)
30. S Kim, ER Dougherty, ML Bittner, Y Chen, K Sivakumar, P Meltzer, JM Trent, A general framework for the analysis of multivariate gene interaction via expression arrays. Biomedical Opt. **5**(4), 411–424 (2000)
31. R Layek, A Datta, R Pal, ER Dougherty, Adaptive intervention in probabilistic Boolean networks. Bioinformatics **25**(16), 2042–2048 (2009)
32. Á Chaos, M Aldana, C Espinosa-Soto, B García Ponce de León, A Garay Arroyo, ER Alvarez-Buylla, From genes to flower patterns and evolution: dynamic models of gene regulatory networks. J Plant Growth Regul. **25**(4), 278–289 (2006)
33. M Peleg, D Rubin, R Altman, Using Petri net tools to study properties and dynamics of biological systems. J Am Med Informatics Assoc. **12**(2), 181–199 (2005)
34. R Laubenbacher, B Stigler, A computational algebra approach to the reverse engineering of gene regulatory networks. J Theor Biol. **229**(4), 523–537 (2004)
35. ES Dimitrova, AS Jarrah, R Laubenbacher, B Stigler, A Gröbner, Fan method for biochemical network modeling, in *Proc International Symposium on Symbolic and Algebraic Computation (ISSAC)*, ACM, pp. 122–126 (2007)
36. A Jarrah, R Laubenbacher, B Stigler, M Stillman, Reverse-engineering of polynomial dynamical systems. Adv Appl Math. **39**, 477–489 (2007)
37. B Stigler, A Jarrah, M Stillman, R Laubenbacher, Ann N.Y. Acad Sci. **1115**, 168–177 (2008)
38. A Jarrah, R Laubenbacher, A Veliz-Cuba, The dynamics of conjunctive and disjunctive Boolean network models. Bull Math Bio. (2010). doi: 10.1007/s11538-010-9501-z.
39. A Veliz-Cuba, A Jarrah, R Laubenbacher, Polynomial algebra of discrete models in systems biology. Bioinformatics (2010). doi: 10.1093/bioinformatics/bt9240.
40. I Cantone, L Marucci, F Iorio, MA Ricci, V Belcastro, M Bansal, S Santini, M di Bernardo, D di Bernardo, MP Cosma, A yeast synthetic network for in vivo assessment of reverse-engineering and modeling approaches, Cell **137**, 172–181 (2009)
41. B Buchberger, in *Gröbner Bases: an Algorithmic Method in Polynomial Ideal Theory*, Chapter 6, ed. by Bose NK (D. Reidel Publishing Co., Dordrecht, Boston, Lancaster, 1985), pp. 184–232
42. L Robbiano, On the theory of graded structures. J Symbolic Comput. **2**, 139–170 (1986)
43. E Allen, J Fetrow, L Daniel, S Thomas, D John, Algebraic dependency models of protein signal transduction networks from time-series data. J Theor Biol. **238**(2), 317–330 (2006)
44. T Mora, L Robbiano, Gröbner fan of an ideal. J Symbolic Comput. **6**(2/3), 183–208 (1988)
45. A Jensen, Gfan, a software system for Gröbner fans, 2005. http://www.math.tuberlin.de/~jensen/software/gfan/gfan.html
46. K Fukuda, AN Jensen, RR Thomas, Computing Gröbner fans. Math Comput. **76**(260), 2189–2212 (2007)
47. ES Dimitrova, Estimating the relative volumes of the cones in a Groebner fan. Math Comput Sci. **3**(4), 457 (2009)
48. J Yu, VA Smith, PP Wang, AJ Hartemink, ED Jarvis, Advances to Bayesian network inference for generating causal networks from observational biological data. Bioinformatics **20**, 3594–3603 (2004)
49. G Della Gatta, M Bansal, A Ambesi-Impiombato, D Antonini, C Missero, D di Bernardo, Direct targets of the TRP63 transcription factor revealed by a combination of gene expression profiling and reverse engineering. Genome Res. **18**, 939–948 (2008)
50. TS Gardner, D di Bernardo, D Lorenz, JJ Collins, Inferring genetic networks and identifying compound mode of action via expression profiling. Science **301**, 102–105 (2003)
51. K Basso, AA Margolin, G Stolovitzky, U Klein, R Dalla-Favera, A Califano, Reverse engineering of regulatory networks in human B cells. Nat Genet. **37**, 382–390 (2005)
52. ES Dimitrova, J McGee, R Laubenbacher, P Vera Licona, Comparison of discretization methods for network inference. J Comput Biol. (2010). doi: 10.1016/j.tcs.2010.04.034
53. S Kauffman, C Peterson, B Samuelsson, C Troein, Random Boolean network models and the yeast transcriptional network. PNAS **100**(25), 14796–14799 (2003)
54. R Lidl, H Niederreiter, *Finite Fields, Encyclopedia of Mathematics and Its Applications,* vol. 2nd edn. (Cambridge University Press, New York, 1997)
55. D Cox, J Little, D O'Shea, D. *Ideals, Varieties, and Algorithms* (Springer Verlag, New York, 1997)
56. D Pritchard, Walking Through the Gröbner Fan (2001). www.math.uwaterloo.ca/dagpritc/math.html.
57. B Buchberger, An algorithm for finding a basis for the residue class ring of a zero-dimensional polynomial ideal. Ph.D. Thesis, University of Innsbruck (1965)
58. ES Dimitrova, LD Garcia-Puente, F Hinkelmann, AS Jarrah, R Laubenbacher, B Stigler, P VeraLicona, Parameter estimation for Boolean models of biological networks. J Theor Comput Sci. (2010). doi: 10.1016/j.tcs.2010.04.034