BIOINFORMATICS ORIGINAL PAPER Vol. 25 no. 13 2009, pages 1686–1693
doi:10.1093/bioinformatics/btp281

Systems biology

List-decoding methods for inferring polynomials in finite dynamical gene network models

Janis Dingel[1,*] and Olgica Milenkovic[2,*]

[1]Institute for Communications Engineering, Technische Universität München, Munich, Germany and
[2]Coordinated Science Lab, University of Illinois at Urbana Champaign, Urbana, IL, USA

Received on August 7, 2008; revised on March 26, 2009; accepted on April 21, 2009

Advance Access publication April 28, 2009

Associate Editor: Olga Troyanskaya

ABSTRACT

Motivation: The problem of reverse engineering the dynamics of gene expression profiles is of focal importance in systems biology. Due to noise and the inherent lack of sufficiently large datasets generated via high-throughput measurements, known reconstruction frameworks based on dynamical systems models fail to provide adequate settings for network analysis. This motivates the study of new approaches that produce stochastic lists of explanations for the observed network dynamics that can be efficiently inferred from small sample sets and in the presence of errors.

Results: We introduce a novel algebraic modeling framework, termed stochastic polynomial dynamical systems (SPDSs) that can capture the dynamics of regulatory networks based on microarray expression data. Here, we refer to *dynamics of the network* as the trajectories of gene expression profiles over time. The model assumes that the expression data is quantized in a manner that allows for imposing a finite field structure on the observations, and the existence of polynomial update functions for each gene in the network. The underlying reverse engineering algorithm is based on ideas borrowed from coding theory, and in particular, list-decoding methods for so called Reed-Muller codes. The list-decoding method was tested on synthetic data and on microarray expression measurements from the M^{3D} database, corresponding to a subnetwork of the *Escherichia coli* SOS repair system, as well as on the complete transcription factor network, available at RegulonDB. The results show that SPDSs constructed via list-decoders significantly outperform other algebraic reverse engineering methods, and that they also provide good guidelines for estimating the influence of genes on the dynamics of the network.

Availability: Software codes for list-decoding algorithms suitable for direct application to quantized expression data will be publicly available at the authors' web-pages.

Contact: janis.dingel@tum.de; milenkov@uiuc.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 INTRODUCTION

Modeling the coupled dynamics of gene expression patterns is an important task in systems biology that is most accurately performed via systems of coupled differential equations, derived by analyzing involved biochemical reactions of the cell cycle (de Jong, 2002). Besides their accuracy, these modeling approaches also have the advantage that they lend themselves to stochastic extensions that capture uncertainties in biological systems (Wilkinson, 2006). However, systems of differential equations have very high parametric complexity and can often only be built bottom-up, i.e. by looking at a detailed list of reactions involved in gene transcription and by determining parameters of the model through extensive experimental studies. An alternative approach to modeling gene networks relies on very coarse approximations of the system's dynamics; this coarse approximation allows only for capturing qualitative features of gene networks, rather than their exact dynamics. This approach was pioneered by Kauffman, who proposed using Boolean networks (BNs) as models of gene regulatory networks (GRN) (Kauffman, 1969). Kauffman's work has inspired extensive research in two directions: theoretical analysis of GRNs and practical application of BNs and generalizations thereof to gene network modeling.

Probabilistic BNs (PBNs) represent stochastic extensions of Boolean models (Shmulevich *et al.*, 2002). In a PBN, a *list* of Boolean functions is associated with each node in the network, and each time the state of a gene is updated, only one of these functions is randomly chosen to compute the new state of the gene (Shmulevich *et al.*, 2002). Many generalizations of these models exist, including hybrid systems (Thieffry and Thomas, 1998), finite state linear models (Brazma and Schlitt, 2003) and deterministic and probabilistic finite dynamical systems (Aviñó *et al.*, 2004; Jarrah *et al.*, 2007). Significant research effort has been devoted to fitting discrete models of GRNs to high-throughput measurements in a manner that allows for good predictive descriptions of experimental data. Such an approach, which is often referred to as reverse engineering, is fundamentally different from the bottom-up methods as it tries to select a model based purely on the observed data, without making use of detailed biochemical information. Early approaches to reverse engineering of GRNs under both Boolean and PBN models have assumed that time-series data from a network can be perfectly observed (Akutsu *et al.*, 2000; Liang *et al.*, 1998). More recent methods account for uncertainty in the data by invoking information theoretic techniques and computational algebraic frameworks. In the former case, using the minimum description length principle allows for developing reverse engineering methods that outperform purely deterministic approaches (Dougherty *et al.*, 2008). In addition, algebraic analytical

*To whom correspondence should be addressed.

frameworks allow for fitting a finite dynamical system model to time-course microarray expressions, assuming purely deterministic observations (Dimitrova *et al.*, 2007; Laubenbacher and Stigler, 2004). However, noise in the expression measurements and the inherently stochastic nature of biological processes make reverse engineering within any of the above described deterministic frameworks only of limited practical value.

In this article, we present a constructive approach for reverse engineering gene expression dynamics that is casted within the algebraic framework developed in (Laubenbacher and Stigler, 2004). First, we develop a theoretical framework for the study of the reverse engineering problem and show that it is closely related to problems arising in coding theory (Pellikaan and Wu, 2004). We then focus on the statistical predictive inference problem of network dynamics given the topology of the network. We address randomness, measurement errors and small sample size issues jointly by applying powerful list-decoding algorithms that can be shown to optimally deal with missing observations and noise from a coding theoretic perspective. This method overcomes the drawbacks of models that assume noiseless observations and that rely on large sample set sizes. Our method is first validated and compared with existing methods using synthetic simulations. It is subsequently applied to *Escherichia coli* DNA microarray data, where it is successfully used for decoding the responses of genes in the SOS repair network. We use the well-known concept of gene influence (D'haeseleer, 2000; Shmulevich *et al.*, 2002) to evaluate our method, and suggest different methods for computing this quantity within a general finite field modeling framework. Our analysis of the complete transcription factor network of *E.coli*, as available from RegulonDB, reveals that the predictions made by the list-decoding algorithm can significantly improve the discrimination of basic features of network dynamics when compared with standard algebraic methods. To the best of our knowledge, this is the first analysis performed on real expression data and networks under the discrete algebraic model.

2 SYSTEM AND METHODS

2.1 Basic definitions

A GRN is a directed graph $G=\{V,E\}$ with vertices $V=\{v_1,...,v_n\}$ representing genes and edges $E\subseteq\{1,..,n\}\times\{1,..,n\}$ describing relationships among the genes. An edge (j,i) is drawn from gene v_j to gene v_i if gene v_j regulates the expression of gene v_i. In this case, we say that v_j is a *regulator* of v_i, and that v_i is a *regulatee* of v_j. Let $\breve{\boldsymbol{v}}_i$ denote the vector of regulators of gene v_i, i.e. $\breve{\boldsymbol{v}}_i=(v_{i_1},\ldots,v_{i_{m(i)}}),\forall i_k$ s.t. $(i_k,i)\in E$. We refer to the number of regulators $m(i)$ of a gene as its in-degree. We refer to the trajectories of the gene expression levels in time as the *dynamics of the network*. We introduce a temporal dimension t and let $v_i(t)$ denote the expression of gene v_i at time t. In this work, we focus on discrete time models and can therefore assume, without loss of generality, that $t\in\mathbb{N}_0$. The network states at times $t=0,1,2,...$ are described by the vector $\boldsymbol{v}(t)=(v_1(t),\ldots,v_n(t))$.

2.2 BNs and polynomial dynamical systems

BN models (Kauffman, 1969) are discrete deterministic models which allow genes to be only in two different states—'ON' or 'OFF'. In other words, expression levels can be seen as elements from a finite field with two elements, denoted by $\mathbb{F}_2$. A BN model is defined via a map of the binary state vector from time t to $t+1$ that can be decomposed into n Boolean functions $f_i:\mathbb{F}_2^{m(i)}\to\mathbb{F}_2$, each associated with one gene v_i in the network:

$$v_i(t+1)=f_i(\breve{\boldsymbol{v}}_i(t)). \tag{1}$$

The dynamics of the network are simulated by starting from an initial state vector $\boldsymbol{v}(0)$, and by iteratively and synchronously updating the Boolean functions f_i. This gives rise to a sequence of states.

The algebraic framework for gene expression profiles considered in the remainder of this article is a generalization of BNs and was first introduced in (Laubenbacher and Stigler, 2004). It allows for a finer representation of gene expression states while maintaining the analytical tractability of BN models by imposing a special form on the update functions f_i. More specifically, a gene is assumed to be in a finite number of states q that represent different expression levels. The number of states is restricted to be a power of a prime p, $q=p^s$, $s\in\mathbb{N}$, so that the states of the genes correspond to elements of a finite field, denoted by $\mathbb{F}_q$, i.e. $v_i(t)\in\mathbb{F}_q$. This is not an overly restrictive assumption since many small integers such as $2,3,4,5,7,8,9,11$ can be expressed in the form p^s, and since very fine quantization schemes are not of practical interest. Analogous to BNs, the dynamics of the system are specified through the equivalent mapping of the state vector $\boldsymbol{v}(t)\in\mathbb{F}_q^n, t\in\mathbb{N}_0$:

$$\mathcal{F}_q:\ \mathbb{F}_q^n\to\mathbb{F}_q^n,\ \boldsymbol{v}(t)\mapsto\boldsymbol{v}(t+1)=\mathcal{F}_q(\boldsymbol{v}(t)). \tag{2}$$

A crucial point for the methods used in this article is the well-known fact that any possible function $f:\mathbb{F}_q^m\to\mathbb{F}_q$ defined over an arbitrary finite field $\mathbb{F}_q$ is of the form of a multivariate polynomial, i.e. every function f can be expressed as

$$f=\sum_{i_1,\ldots,i_m} a_{i_1i_2\ldots i_m}x_1^{i_1}\ldots x_m^{i_m},\quad a_{i_1i_2\ldots i_m}\in\mathbb{F}_q. \tag{3}$$

We use $\mathbb{F}_q[x_1,x_2,\ldots,x_m]$ to represent the set of all possible multivariate polynomials given by Equation (3). Hence, in our model, a function f_i associated with a node v_i in the GRN is a multivariate polynomial in the variables $\breve{\boldsymbol{v}}_i$, the regulators of v_i. In what follows, we refer to this model as a *polynomial dynamical system* (PDS).

2.3 Reverse engineering frameworks

We consider the following reverse engineering problem: one is given the topology of the network and a limited set of time-course expression points for all the genes in the network. In this setting, a reverse engineering algorithm has to infer a PDS that *explains* the observed time-course data. Let us first assume that the expression data is error-free and that the PDS model perfectly characterizes the real dynamics of the GRN. Then we have to solve the following problem, which we henceforth refer to as the *noiseless reconstruction problem*. Given a time series of $T+1$ transitions for each gene v_i, i.e. pairs of observed inputs $\breve{\boldsymbol{v}}_i(t)$ and corresponding outputs $v_i(t+1)$

$$\begin{aligned}\mathcal{V}_i&=\{(\breve{\boldsymbol{v}}_i(0),v_i(1)),\ldots,(\breve{\boldsymbol{v}}_i(T),v_i(T+1))\}\\&\dot{=}\{(\breve{\boldsymbol{v}}_i(t),v_i(t+1))\}_{t=0}^{T},\end{aligned}$$

find the functions f_i that reproduce the time series exactly.

This is the framework originally considered in (Laubenbacher and Stigler, 2004) and the followup work (Aviñó *et al.*, 2004;

Delgado-Eckert, 2009; Dimitrova *et al.*, 2007; Just, 2006). In (Laubenbacher and Stigler, 2004), the selection process in the noiseless reconstruction problem consists in identifying a *minimal* solution. Such a solution is obtained by first interpolating a polynomial $f_{i,(0)}(\check{\boldsymbol{v}}_i)$ through the $T+1$ points. Here, $f_{i,(0)}$ satisfies

$$f_{i,(0)}(\check{\boldsymbol{v}}_i(t))=v_i(t+1),\ \forall t=0,\ldots,T.$$

Next, the set $I(\mathcal{V}_i)$ of polynomials vanishing on the input points in $\mathcal{V}_i$, i.e.

$$I(\mathcal{V}_i)=\{g_i\in\mathbb{F}_q[x_1,\ldots,x_m]: g_i(\check{\boldsymbol{v}}_i(t))=0,\ t=0,\ldots,T\},$$

is constructed. Such a set, usually called a vanishing ideal, has the property that any interpolator $f_{i,(0)}$ lies in the coset

$$f_i+I(\mathcal{V}_i)=\{f_i+g_i: g_i\in I(\mathcal{V}_i)\},$$

where f_i denotes an interpolator that cannot be further decomposed— i.e. a *reduced interpolator*. A reduced interpolator is defined by the following property: $\nexists(f_i^{'}\in\mathbb{F}_q[x_1,\ldots,x_m],\ g_i^{'}\in I(\mathcal{V}_i))$: $f_i=f_i^{'}+g_i^{'}$. In other words, g_i represents the part of $f_{i,(0)}$ that lies in $I(\mathcal{V}_i)$ and f_i represents the reduction of $f_{i,(0)}$ with respect to $I(\mathcal{V}_i)$. The reduced solution f_i subsequently serves as an update model for the node v_i. The ideal of vanishing polynomials $I(\mathcal{V}_i)$ has a finite polynomial Gröbner basis (Cox *et al.*, 1992). This basis can be computed by the Buchberger–Möller algorithm, and then used for identifying minimal solutions (Dimitrova *et al.*, 2007; Laubenbacher and Stigler, 2004). The method described above has the following drawback: as noted by the authors in Laubenbacher and Stigler (2004), the interpolation method is extremely sensitive to measurement errors and it does not consider missing expression observations. These problems arise due to the fact that in the first step of the modeling process, an interpolation polynomial is found that passes through *all* data points, which may cause overfitting in the presence of noise. An alternative approach to the method above is to approximate the time series by using lists of update functions, thereby accounting for missing samples and read-out errors caused by noise. To implement this approach, we describe next how this reverse engineering problem is connected to coding theory, and introduce a reconstruction method based on list-decoding. List-decoders can identify the exact update functions in the presence of both missing values and noise, provided that the total number of missing values and errors is properly bounded. Robustness with respect to measurement errors is achieved by bounding the degree of the update polynomial, thus allowing the interpolated functions in the list to agree only with a fraction of the observed transitions.

2.4 Stochastic probabilistic models and coding theory

2.4.1 Additive noise We now turn our attention to a more rigorous formulation of the reverse engineering problem. In this context, we address two important issues: availability of only a limited amount of time-course data, even for the best studied organisms; and errors in the measured data. Noise in the data arises both due to measurement errors and the fact that biological systems are inherently stochastic and influenced by factors that cannot be measured. In this case, the purely deterministic view within the framework of the noiseless reconstruction problem represents an inadequate approach. We therefore propose to recast the reverse engineering problem into a probabilistic framework and consider the following *noisy reconstruction problem:* given a time series of $T+1$ noisy transitions for each gene v_i, i.e. a sequence of pairs of inputs and corresponding outputs

$$\mathcal{V}_i'=\{(\check{\boldsymbol{v}}_i(t),v_i(t+1))\}_{t=0}^{T},$$

find a *set of functions* f_i that jointly provide the best approximation for the observed time series. Here, it is assumed that $v_i(t+1)=f_i(\check{\boldsymbol{v}}_i(t))+\epsilon_i(t+1)$, where $\epsilon_i(t)\in\mathbb{F}_q$, and $P(\epsilon_i(t)\neq 0)=p_e$ denotes the noise samples. Note that in this setting, small sample sets and measurement errors are accounted for by forming a *list* of possible node update functions, similarly as for the case of probabilistic BN models (Shmulevich *et al.*, 2002), but with some major differences summarized in the exposition to follow.

Measurement noise affects *both* the inputs and the outputs of the time series. However, one can assume that only an 'output noise' component exists, since the effect of input noise can be transformed into an equivalent output noise component. To clarify this issue, assume that the noise-free input is $\check{\boldsymbol{v}}_i'(t)$, with corresponding noise free output $v_i'(t+1)$, and that a noise sample $\check{\boldsymbol{\epsilon}}_i'(t)$ is added to the input, and a noise sample $\epsilon_i'(t+1)$ is added to the output. Let $\check{\boldsymbol{v}}_i(t):=\check{\boldsymbol{v}}_i'(t)+\check{\boldsymbol{\epsilon}}_i'(t)$ and let $\epsilon_i(t+1):=f_i(\check{\boldsymbol{v}}_i(t))-v_i'(t+1)+\epsilon_i'(t+1)$ denote the 'transformed noise' affecting only the output. Then, $f_i(\check{\boldsymbol{v}}_i'(t)+\check{\boldsymbol{\epsilon}}_i'(t))+\epsilon_i'(t+1)=v_i'(t+1)+\epsilon_i(t+1)$, as claimed.

2.4.2 Reed–Muller codes As shown above, the noiseless and noisy reverse engineering problems are essentially interpolation and approximation problems for polynomials over finite fields. This is a well-studied subject in coding theory, due to its application in various decoding algorithms. The coding-theoretic objects of interest are generalized q-ary Reed–Muller (RM) codes, defined as follows. Let a multivariate polynomial f be of the form (3). The total degree of f is defined as

$$\text{totdeg}=\max\{i_1+i_2+\cdots+i_m: a_{i_1i_2\ldots i_m}\neq 0\}.$$

A q-ary RM code $\mathcal{RM}_q(u,m)$ is the set of all m-variate polynomials $f\in\mathbb{F}_q[x_1,\ldots,x_m]$ with $\text{totdeg}(f)\leq u$, evaluated at the q^m distinct elements of the finite field $\alpha_j\in\mathbb{F}_{q^m}$, i.e.

$$\mathcal{RM}_q(u,m)=\{(f(\alpha_0),\ldots,f(\alpha_{q^m-1})):\\ f\in\mathbb{F}_q[x_1,\ldots,x_m],\text{totdeg}(f)\leq u\}. \quad (4)$$

RM codes are used to encode messages into codewords, in order to subsequently reconstruct the message from an observed noisy codeword. The reconstruction process is called decoding. As already pointed out, in RM codes, the encoded messages—codewords—are multivariate polynomials of bounded degree. Given noisy samples

$$\boldsymbol{c}_\epsilon=(f(\alpha_0)+\varepsilon_0,\ldots,f(\alpha_{q^m-1})+\varepsilon_{q^m-1}) \quad (5)$$

an optimal *maximum-likelihood* RM decoder has to find the polynomial f of bounded degree u that most likely led to the observation. However, this problem is computationally intractable for long codeword lengths and suboptimal decoders must be used instead. Recently, a class of algorithms was described in coding literature that can closely approach the optimal performance with polynomial computational times. Among these decoding algorithms, list-decoders are of special interest, since they can operate in extremely noisy regimes. From the simple discussion above, it is apparent that the decoding problem is equivalent to the noisy reconstruction problem.

2.4.3 List-decoding List-decoding algorithms can handle very large noise levels and missing sample points, while allowing the decoder to generate a list of possible solutions, rather than a unique candidate output. With small sample sizes encountered in gene network reconstruction problems, one inevitably has to employ list-decoding and allow for non-unique solutions. A list-decoder takes the noisy samples $\boldsymbol{c}_\epsilon$ and produces a list $\{g_1,\ldots,g_L\}$ that contains all polynomials $g\in\mathbb{F}_q[x_1,\ldots,x_m]$ of total degree less than or equal to u, such that the Hamming distance (i.e. the number of positions in which two words disagree) of g and $\boldsymbol{c}_\epsilon$ is smaller than or equal to a predefined constant e_θ, called the *decoding radius*. Alternatively, the list includes polynomials g such that

$$|\{j: g(\alpha_j)\neq f(\alpha_j)+\epsilon_j\}|\leq e_\theta. \tag{6}$$

Note that the parameters L and e_θ are intimately connected to the error rate, and that each list-decoding algorithm provides different operational regions for these parameters. Detailed overviews of list-decoding algorithms can be found in Guruswami and Sudan (1999), Gaborit and Ruatta (2006), Gopalan *et al.* (2007) and (Santhi, 2007). We focus our attention on a list-decoding method for RM codes recently described by Pellikaan and Wu (2004), which exploits the fact that $\mathcal{RM}_q(u,m)$ codes defined over $\mathbb{F}_q$ are subfield–subcodes of the well-studied generalized Reed–Solomon (RS) codes, defined over $\mathbb{F}_{q^m}$. Note that the Pelilikaan–Wu (PW) algorithm is just one of many possible list-decoding methods for RM codes: some very recent results that offer excellent performance for small field sizes include Gopalan *et al.* (2008). We focus on the former technique because of its easy implementation and simple operational principles. Exactly the same procedure can be applied for reverse engineering stochastic polynomial dynamical systems (SPDSs) models for GRNs. In this case, the inputs $\breve{\boldsymbol{v}}_i(t)$ of the $m(i)$ regulators are interpreted as points from the field $\mathbb{F}_q^{m(i)}$ corresponding to the evaluation points α_i of the RM code with $m=m(i)$ in Equation (4). The node function $f_i\in\mathbb{F}_q[x_1,\ldots,x_{m(i)}]$ corresponds to the encoded message and the outputs $v_i(t+1)$ represent (noisy) codeword symbols of the RM code. Note that the assumption of bounded degree of RM codewords does not impose a severe restriction on the biological properties of gene networks, since this parameter can be freely chosen and fairly large values for the degree bound can be tested in a sequential manner.

A more detailed description of the PW algorithm can be found in the Supplementary Material.

3 THE REVERSE ENGINEERING ALGORITHM

3.1 Microarray data preprocessing

We quantized gene expressions using q levels, in terms of K-means clustering. Quantization of microarray data is a critical step in the reverse engineering process, and the influence of quantization on network dynamics inference is still not completely understood. Still, quantization is an integral step of all finite state gene network modeling approaches (Brazma and Schlitt, 2003; de Jong, 2002; Schlitt and Brazma, 2007) including Bayesian network techniques. Quantization is most often performed using simple thresholding and clustering techniques or fitting Gaussian mixture models (Bulashevska and Eils, 2005; Gat-Viks *et al.*, 2006; Pe'er *et al.*, 2001; Shmulevich and Zhang, 2002). To the best of our knowledge, there still does not exist a quantization technique that takes into account error models for DNA microarray measurements (Ideker *et al.*, 2000). As a consequence, different quantizers may lead to quite different outputs of the reverse engineering algorithm. By using a sufficiently large number of quantization levels, this problem can be avoided to a certain degree, and this motivates the use of non-binary network dynamics models. A rigorous analysis of the influence of the number of quantization levels on the accuracy of the reconstruction method is beyond the scope of the present work, and will be presented elsewhere.

3.2 Constructing lists of approximating polynomials

As described in the previous section, we are given a set of noisy transitions for each node in the network. The first step in the reconstruction algorithm is to reduce the observations to a set of *unique transitions*. The observed inputs $\breve{\boldsymbol{v}}_i(t)$ in the set $\mathcal{V}_i'$ may not be unique, i.e. there may exist pairs of indices $(t,t')\in\{0,\ldots,T\}^2$, $t\neq t'$, for which $\breve{\boldsymbol{v}}_i(t)=\breve{\boldsymbol{v}}_i(t')$. Note that due to noise, the existence of such pairs does not imply that the corresponding outputs $v_i(t+1)$ and $v_i(t'+1)$ are equal. Hence, we can encounter 'multiple transitions' for the same input. In order to analyze whether the source of such inconsistent data is the discretization method, we applied four different quantization schemes to the expression data described in Section 4.2.1, and analyzed their effect on the multiple transition problem using the RegulonDB transcription factor network (Gama-Castro *et al.*, 2008). Data were quantized using $q=3,5,7$ expression levels. All genes in the network with in-degree smaller or equal to three were considered. Ideally, quantization schemes should take into account error models for microarray measurements and the distribution of gene expression. However, determining such models is an open research problem. A quantization scheme that tries to fit a certain type of error model was proposed in Di Camillo *et al.* (2005), but it requires redundant gene expression data. Other methods model expression values as mixtures of Gaussian variables (Chung *et al.*, 2006). Both approaches are not suitable for the considered dataset, and we therefore focus on model-free approaches. The different quantization schemes tested include K-means clustering, maximum entropy quantization, uniform (linear) quantization and a method presented by Shmulevich and Zhang (2002) for binary discretization which we generalized for the $q$ary case. The methods are described in more detail in the Supplementary Material. Multiple transitions were observed for all quantization schemes—which may imply that this effect is not caused by quantization only.

We used mutual information between the input stimulus and the corresponding output of a gene as an evaluation measure for the quantization methods. The mutual information is large if the input and output states are always consistent and small if the input and output states seem to occur independently. The averaged mutual information per gene, obtained for the different quantization schemes we tested, is shown in Table 1.

Multiple transitions are coped with according to the procedure described next. First, the observed set $\mathcal{V}_i'$ is reduced to a set of unique transitions $\mathcal{U}_i$ as follows: assume we observe the same input to gene v_i exactly r times, $\breve{\boldsymbol{v}}_i(t_1)=\breve{\boldsymbol{v}}_i(t_2)=\cdots=\breve{\boldsymbol{v}}_i(t_r)$, with corresponding outputs $v_i(t_1+1),\ldots,v_i(t_r+1)$. We retain only one transition pair in $\mathcal{U}_i$, with input $\breve{\boldsymbol{v}}_i(t_1)$ and corresponding output equal to the majority vote of the observed outputs. Ties are broken arbitrarily. The inputs in $\mathcal{U}_i$ are subsequently regarded as the sample points of an RM code of unknown dimension, i.e. unknown bounded

Table 1. Mutual information between the input stimulus and the corresponding output of a gene for different quantization schemes

Method	Mutual Information		
	$q=3$	$q=5$	$q=7$
K-means	0.2479	0.5895	0.9360
Method by Shmulevich and Zhang (2002)	0.0555	0.1988	0.3491
Maximum entropy	0.3110	0.8738	1.3912
Uniform	0.2323	0.5483	0.8776

degree u of the encoder function f_i; then, the corresponding outputs represent noisy sample points of this polynomial. Upon reduction of the set $\mathcal{V}_i'$, we have $q^m - |\mathcal{U}_i|$ missing transitions at node v_i that we model as 'erasures' during decoding. List-decoding is used to explore the space of low-degree polynomials that approximate the above described time series: one starts with the smallest possible degree $u=1$, and then tries to find polynomials that approximate the transitions in $\mathcal{U}_i$. In other words, the goal is to try to find interpolating functions that can disagree with the pairs in $\mathcal{U}_i$ up to a fraction of points. In the next iteration, we increase u by one and repeat the decoding process. For each value of u, the list of decoded codewords is stored in $\mathcal{C}_i^{(u)}$. The output of the algorithm for node v_i is the list $\mathcal{C}_i = \bigcup_u \mathcal{C}_i^{(u)}$. Optionally, the solution found by the Gröbner basis method as described in Section 2.3 can be added to the list as well. When the algorithm produces a non-empty list for a given node, its output is replaced by one of the decoded words and the measurements of its regulatees are updated. In this way parent nodes can be used to aid the decoding of their children by correcting their input measurements. Our algorithm accounts for this feature through an iterative refinement technique that is used until no changes in the regulatee's lists are observed. The complexity of the algorithm is determined by the complexity of the list-decoding algorithm at hand, which for the PW method is roughly $O(|\mathcal{U}_i|^2\theta^4)$ operations per gene. Here, $\theta \geq 1$ denotes an adjustable integer parameter that determines the designed decoding radius e_θ given in Equation (6). As each node in the network is decoded separately, the complexity grows approximately linear with the number of nodes in the network (approximately because of the iterative update scheme). The steps of this algorithm are summarized in the Supplementary Material. A flowchart of the overall procedure is shown in Figure 1.

3.3 Reconstruction bounds

Consider the collection of noisy samples in Equation (5). There are two types of errors: an 'erasure' occurs when a sample is erased from the observations. This is equivalent to an unobserved transition at a given network node that arises due to small sample set sizes. An error due to noise occurs at positions j corresponding to $\epsilon_j \neq 0$. Note the important difference between erasures and errors: in the former case, the positions of inaccurate symbols are known a priori to the algorithm, whereas in the latter case those positions are unknown. Based on the list-decoding framework, it is possible to describe *exact analytical bounds* for the minimum required number of measured transitions for reconstructing update function lists of a given size. It is easy to show that the polynomial f can be *uniquely* recovered from the noisy samples as long as the combined number of

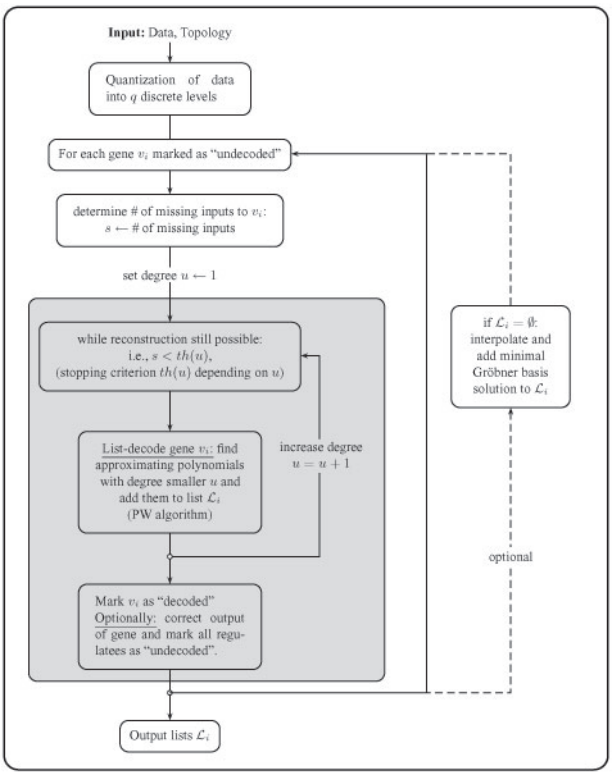

Fig. 1. Flowchart of our method.

errors e and erasures s satisfies $s + 2e < d_{\min}$, where $d_{\min}$ denotes the minimum Hamming distance between any two codewords. Note that this bound applies to both the *noiseless* problem (for which $e=0$), as well as the *noisy* reconstruction problem. It is known that for an RM code, $d_{\min}$ depends on the total degree of the polynomials, and $d_{\min}$ decreases when u increases (Pellikaan and Wu, 2004). This is a very intuitive result because it basically states that the more non-linearity is present in our network the more samples must be taken in order to infer the functions and less noise can be tolerated in this case.

When list-decoding is employed, the number of errors and erasures that can be accounted for is significantly larger—linear in the size of the complete dataset. For illustration, several examples of performance characteristics for list-decoders of RM codes with small values of q are shown in Table 2. These numerical values are based on the bounds described in the previous section.

4 RESULTS

4.1 Synthetic networks

We tested the performance of our algorithm on random synthetic networks. We simulated 1000 PDS over the alphabet $\mathbb{F}_5$, with 20 nodes and random topologies. Nodes had either in-degree 0 (30%), 2 (50%) or 3 (20%). The degrees of the node update functions were randomly chosen from $\{1,2,3,4,5\}$. We restricted our attention to small degrees only, since biological networks have similar properties. The network was initialized with a random state $\boldsymbol{v}(0) \in \mathbb{F}_5^{20}$, and five transitions of the network were recorded.

Table 2. Decoding radius e_θ for different total degrees u and list size upper bound L_θ

In-degree 2			In-degree 3		
u	e_θ	L_θ	u	e_θ	L_θ
1	13	9	1	66	11
2	8	3	2	44	11
3	5	5	3	27	11
4	2	1	4	12	1
5	1	1	5	9	1

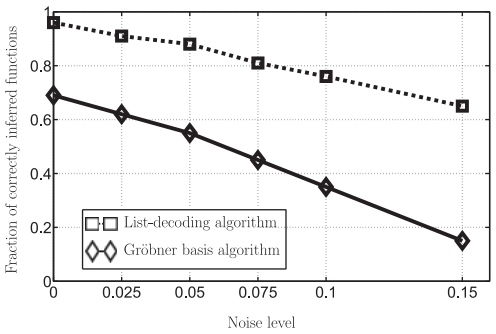

Fig. 2. Comparison of the performance of the proposed algorithm and the Gröbner bases method *in silico*.

This was repeated 50 times, producing a set of 250 synthetic expression samples. Different additive white Gaussian noise samples were added to these 'measurements'. Figure 2 shows the fraction of correctly inferred node functions of our reverse engineering approach for the simulated network, given different noise levels. For comparison, the performance of the Gröbner basis method described in Section 2.3 is shown as well. It can be observed that our algorithm significantly outperforms the latter. Even in the noiseless case, we achieve a significantly higher reconstruction rate and the performance of the Gröbner approach drops quite fast when noise is introduced to the measurements. Note that for the simulations, noise was also added to the inputs $\check{\mathbf{v}}_i(t)$ and the probability that any symbol in the data was changed is termed 'the noise level'.

4.2 Application to the *E.coli* network

4.2.1 Data In order to facilitate the analysis of expression data compiled from multiple laboratories, Faith *et al.* (2007) recently developed M^{3D}, a unified microarray database for several microbial organisms. M^{3D} contains only single-channel arrays using the same platform, and the raw expression data is uniformly normalized to enable the analysis across different experiments without any additional user-dependent processing. We used the M^{3D} microarray data for *E.coli.* and filtered time-course experiments providing at least one transition of the network. We found a total of 90 time points from 21 different experiments, resulting in a total of 69 transitions.

4.2.2 Escherichia coli SOS network We extracted a small subnetwork consisting of nine genes whose topology is given in Gardner *et al.* (2003). Our nine gene 'test network' includes the transcription factor *lexA* and *recA*, a gene that catalyzes DNA strand exchange and renaturation. Both genes are known to interact with each other and to regulate many genes directly and tens or possibly hundreds indirectly. The four genes (*ssb, recF, dinI* and *umuDC*) are regulatory genes involved in the SOS response system, while the three genes (*rpoD, rpoH* and *rpoS*) code for sigma factors. The topology of the network is depicted in Figure 3. We applied our algorithm using different numbers of quantization levels q. At quantization level $q=5$, we found statistically significant low-degree approximating polynomials for three of the nine genes in the network. The inferred input/output responses are shown in Figure 4. Statistical significance was evaluated by randomly choosing nine genes from the available set of 4292 genes, and by applying the reverse engineering algorithm assuming the same topology as in Figure 3. Functions found in this way are counted as false positives. This experiment was repeated 1000 times and the number of non-empty lists was counted. Only in six out of the 1000 iterations, three nodes were simultaneously inferred in the same network. The overall observed false positive rate per node was <3.8%. The sigma factors *rpoS* and *rpoH* exhibit identical responses; both are regulated by *rpoD* and include a self-loop, which may explain this finding. All inferred polynomials are linear ($u=1$), indicating that non-linear functions could not be found with the provided number of samples and quality of data.

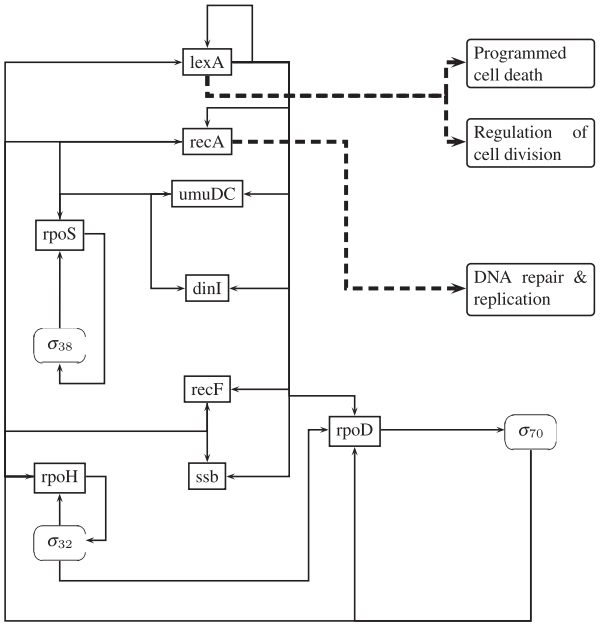

Fig. 3. Diagram of interactions in the SOS network as described in (Gardner *et al.*, 2003). Angled boxes denote genes and small rounded boxes proteins.

4.2.3 Known E.coli transcription factor network We also applied our method to the complete transcription factor gene network available at RegulonDB (Gama-Castro *et al.*, 2008). We removed all genes from this network for which there were no entries in the M^{3D} database, so that the reduced network consisted of 1384 genes. Our algorithm was applied to all nodes in this network with in-degree 2 or 3 (a total of 526 genes), by setting $q=3$ for all nodes. Additionally, polynomials inferred by the Gröbner basis method were added to the lists. In order to verify that the new constructive

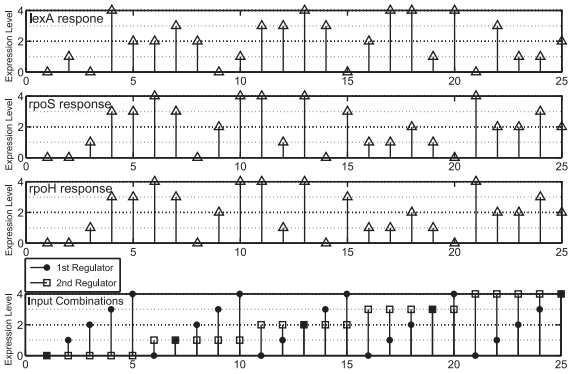

Fig. 4. Inferred responses of the *E.coli.* genes *lexA, rpoS, rpoH*. Each of these genes has two regulators that can take five different values resulting in $5^2=25$ possible input combinations $\vec{\boldsymbol{v}}_i(t)$ that are shown in the last row. Rows 1–3 show the quinternary response $v_i(t+1)$ of the genes to the corresponding input at time t depicted in row 4.

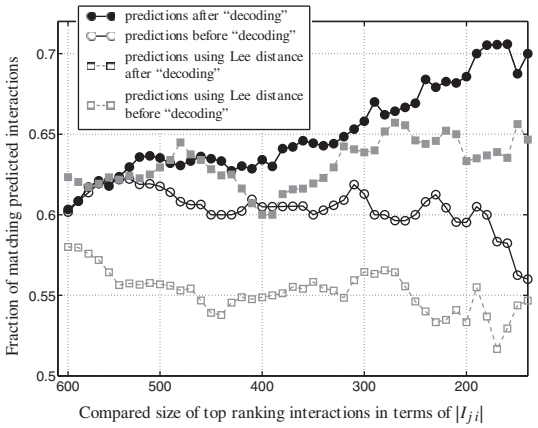

Fig. 5. Predictive power of influence values as obtained from Equation (7) for the *E.coli* transcription factor network before and after our algorithm. For comparison, we show the performance of a measure based on the Lee distance further described in the *Supplementary Material*.

approach can improve the prediction of the system dynamics, the following experiment was performed: first, an influence measure was developed. With the notation $\boldsymbol{x}^{(j,k)}=(x_1,\ldots,x_j=k,\ldots,x_m)$, the *influence* of gene v_j on v_i is defined as

$$I_{ji}=\mathbb{E}_{\boldsymbol{x}}\left\{\frac{1}{2}\sum_{\substack{(k,l)\\k\neq l}}\frac{f_i(\boldsymbol{x}^{(j,k)})-f_i(\boldsymbol{x}^{(j,l)})}{k-l}\right\}. \quad (7)$$

Equation (7) is a mathematical formalism for measuring the influence of a gene on its regulatees. For all input combinations, we calculate the change in the output of gene v_i when changing the input of v_j from l to k, while fixing all regulators other than v_j. Expression levels are regarded as integers and the expectation is taken with respect to all regulators except v_j. The resulting variable I_{ji} should have the following properties: the sign of I_{ji} must give the type of regulation (activating or repressing). The magnitude $|I_{ji}|$ should correlate with the strength of this interaction.[1] Second, the values of I_{ji} were used to indicate an activating influence ($I_{ji}>0$) of gene v_j on gene v_i, or repressing influence ($I_{ji}<0$). The magnitude I_{ji} reflects the strength of the influence. It can be seen that Equation (7) represents a generalization of the notion of influence developed for PBNs (Shmulevich *et al.*, 2002). We compared our predictions obtained from Equation (7) with the influences of regulators annotated in RegulonDB. Influences I_{ji} for all edges that connect to a gene with in-degree 2 or 3 (1225 edges) were calculated before and after applying list-decoding. Calculation of the influence before decoding was based on taking the consensus of a gene's output for multiple transitions and simply neglecting unobserved input combinations in Equation (7). We then ranked genes according to their influence magnitude $|I_{ji}|$ and compared the influence predictions based on sign(I_{ji}) to the RegulonDB annotation for different set sizes of high ranking genes. Figure 5 shows the fraction of the top $600,500,\ldots,200$ ranked genes in terms of $|I_{ji}|$ matching the annotation in RegulonDB. While there is no indication for correlation of matching predictions with high $|I_{ji}|$ before 'decoding', we observe that after 'decoding' the predictions could be significantly improved up to ∼70% matching for the 200 highest ranked genes.

[1]Note that several other influence measures can be used instead (see also the provided Supplementary Material). We focused on this influence model because it offered the best overall prediction performance for the genes tested. See Figure 5 for an illustration.

5 DISCUSSION

The presented approach to the reverse engineering of gene network dynamics builds upon two important principles: that the biological systems are inherently stochastic and that only small and possibly erroneous datasets are available for analysis. Our approach accounts for these facts by setting the problem into a probabilistic and noisy framework, and by allowing for non-unique solutions explaining each node dynamics with a whole list of functions. Our algorithm is *guaranteed* to find all approximating polynomials that generate time series within a predefined Hamming distance from the measured data, as long as the combined number of missing values (erasures) and errors does not exceed the decoding radius of the list-decoder. The decoding radius of the list-decoder is influenced by an adjustable parameter and can be increased at the expense of increased reconstruction complexity. In our simulations, we used the parameter setting that ensures lowest complexity. Thus, the performance of our algorithm represents a lower bound on the achievable performance. As expected, our algorithm outperforms the Gröbner basis approach significantly both in the noisy and noiseless scenario. We used the method based on Gröbner bases as a comparative reference; however, we would like to point out that this method was originally developed for joint inference of both the topology and the dynamics of the network. Due to the fact that there do not exist other known approaches within this modeling framework, and since both methods are based on different assumptions, we think that it is important to perform such a comparison within the proposed context. However, the results of this comparison must be interpreted with care. Many methods concentrating solely on the inference of the network topology are known (Dimitrova *et al.*, 2007), and it may be reasonable to separate the topology inference problem from the dynamics inference problem. The application of our algorithm to

time-course data from *E.coli* validates the usefulness of our approach even with the present small amount of data available. The analysis on the SOS network suggests that at least some gene functions can be approximated by low degree polynomials, while more data seems to be necessary for robust regression of other functions. Furthermore, our approach is constructive in the sense that it returns well-defined functions that specify the response to all possible combinations of inputs. The results regarding the transcription factor network of *E.coli* also show that our constructive approach can significantly improve the analysis of certain features of system dynamics, such as gene influence.

Funding: Deutsche Forschungsgemeinschaft (grant no. HA 1358/10-2 to J.D.); the German Academic Exchange Service (to J.D.); National Science Foundation Awards (CCF 0821910 and CCF 0809895 to O.M., in parts).

Conflict of Interest: none declared.

REFERENCES

Akutsu,T. *et al.* (2000) Algorithms for inferring qualitative models of biological networks. *Pac. Symp. Biocomput.*, 293–304.

Aviñó,M.A. *et al.* (2004) Applications of finite fields to dynamical systems and reverse engineering problems. In *SAC '04: Proceedings of the 2004 ACM Symposium on Applied Computing*. ACM, New York, pp. 191–196 .

Brazma,A. and Schlitt,T. (2003) Reverse engineering of gene regulatory networks: a finite state linear model. *Genome Biol.*, **4**, P5.

Bulashevska,S. and Eils,R. (2005) Inferring genetic regulatory logic from expression data. *Bioinformatics*, **21**, 2706–2713.

Chung,T.-H. *et al.* (2006) Quantization of global gene expression data. In *5th International Conference On Machine Learning and Applications*. Orlando, Florida, USA.

Cox,D. *et al.* (1992) *Ideals, Varieties, and Algorithms*. Springer, New York.

Delgado-Eckert,E. (2009) Reverse engineering time discrete finite dynamical systems: a feasible undertaking? *PLoS ONE*, **4**, e4939.

de Jong,H. (2002) Modeling and simulation of genetic regulatory systems: a literature review. *J. Comput. Biol.*, **9**, 67–103.

D'haeseleer,P. (2000) Reconstructing gene networks from large scale gene expression data. Ph.D. Thesis, The University of New Mexico, Albuquerque.

Dimitrova,E.S. *et al.* (2007) A Gröbner fan method for biochemical network modeling. In *Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation (ISSAC '07)*. ACM, New York, NY, USA, pp. 122–126.

Di Camillo,B. *et al.* (2005) A quantization method based on threshold optimization for microarray short time series. *BMC Bioinformatics*, **6** (Suppl. 4), S11.

Dougherty,J. *et al.* (2008) Inference of gene regulatory networks based on a universal minimum description length. *EURASIP J. Bioinform. Syst. Biol.*, **8**, 1–11.

Faith,J.J. *et al.* (2007) Many microbe microarrays database: uniformly normalized affymetrix compendia with structured experimental metadata. *Nucleic Acids Res.*, **36**, D866–D870

Gaborit,P. and Ruatta,O. (2006) Efficient erasure list-decoding of Reed-Muller codes. In *Proceedings of the 2006 IEEE International Symposium on Information Theory*, IEEE, Seattle, Wahsington,USA, pp. 148–152.

Gama-Castro,S. *et al.* (2008) Regulondb (version 6.0): gene regulation model of *Escherichia coli* k-12 beyond transcription, active (experimental) annotated promoters and textpresso navigation. *Nucleic Acids Res.*, **36**, D120–D124.

Gardner,T.S. *et al.* (2003) Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, **301**, 102–105.

Gat-Viks,I. *et al.* (2006) A probabilistic methodology for integrating knowledge and experiments on biological networks. *J. Comput. Biol.*, **13**, 165–181.

Gopalan,P. *et al.* (2007) Hardness of reconstructing multivariate polynomials over finite fields. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, ACM New York, NY, USA, pp. 349–359.

Gopalan,P. *et al.* (2008) List-decoding Reed-Muller codes over small fields. In *STOC '08: Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pp. 265–274.

Guruswami,V. and Sudan,M. (1999) Improved decoding of Reed-Solomon codes and algebraic geometry codes. *IEEE Trans. Inf. Theory*, **45**, 1757–1767.

Ideker,T. *et al.* (2000) Testing for differentially-expressed genes by maximum-likelihood analysis of microarray data. *J. Comput. Biol.*, **7**, 805–817.

Jarrah,A.S. *et al.* (2007) Reverse engineering of polynomial dynamical systems. *Adv. Appl. Math.*, **39**, 477–489.

Just,W. (2006) Reverse engineering discrete dynamical systems from data sets with random input vectors. *J. Comput. Biol.*, **13**, 1435–1456.

Kauffman,S. (1969) Homeostasis and differentiation in random genetic control networks. *Nature*, **224**, 177–178.

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

Liang,S. *et al.* (1998) Reveal, a general reverse engineering algorithm for inference of genetic network architectures. *Pac. Symp. Biocomput.*, 18–29.

Pe'er,D. *et al.* (2001) Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, **17** (Suppl. 1), S215–S224.

Pellikaan,R. and Wu,X.-W. (2004) List decoding of q-ary Reed-Muller codes. *IEEE Trans. Inf. Theory*, **50**, 679–682.

Santhi,N. (2007) On algebraic decoding of *q*-ary Reed-Muller and product Reed-Solomon codes. In *Proceedings of the IEEE International Symposium on Information Theory, ISIT07*. Nice, France.

Schlitt,T. and Brazma,A. (2007) Current approaches to gene regulatory network modelling. *BMC Bioinformatics*, **8** (Suppl. 6), S9.

Shmulevich,I. and Zhang,W. (2002) Binary analysis and optimization-based normalization of gene expression data. *Bioinformatics*, **18**, 555–565.

Shmulevich,I. *et al.* (2002) Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, **18**, 261–274.

Thieffry,D. and Thomas,R. (1998) Qualitative analysis of gene networks. *Pac. Symp. Biocomput.*, 77–88.

Wilkinson,D.J. (2006) *Stochastic Modelling for Systems Biology*, Vol. 11 of *Chapman & Hall/CRC Mathematical & Computational Biology*. CRC Press, London, UK.