An optimization algorithm for the inference of biological networks

Paola Vera-Licona [1,*], Abdul Jarrah [2], John McGee [3], Luis David Garcia-Puente [4] and Reinhard Laubenbacher [2*]

[1]BioMaPS and DIMACS Institute, Rutgers University, Piscataway, NJ 08854-8087
[2]Virginia Bioinformatics Institute, Bioinformatics Facility I (0477), Virginia Tech, Blacksburg, VA 24061
[3]Mathematics and Statistics Department, Radford University, Radford, VA 24142
[4]Department of Mathematics and Statistics, Sam Houston State University, Huntsville, TX 77341-2206

ABSTRACT
Motivation:
The inference of molecular networks from "omics" data is one of the central problems in systems biology, the so-called reverse-engineering problem. Particular emphasis is on the reverse-engineering of gene regulatory networks from DNA microarray data. There is a growing body of literature focused on the development of efficient algorithms for this purpose. Important features of such algorithms should be robustness to data noise and the ability to incorporate prior biological knowledge.
Results: This paper presents an algorithm that takes as input one or more time courses of microarray data and produces an optimal Boolean network model of the underlying gene regulatory network. The algorithm is able to incorporate both wildtype and perturbation data, as well as prior biological knowledge. In addition to a wiring diagram for the network, the algorithm also provides qualitative information about the network dynamics. The method consists of an evolutionary algorithm (EA) that optimizes the dynamic network structure with respect to data fit and a novel numerical measure of model complexity. It is shown that the algorithm is robust to noise levels expected in microarray data. The algorithm is validated with data generated from a published model of the segment polarity gene network in D. Melanogaster. ROC curves are used to measure the performance and robustness of the method.
Availability: The algorithm has been implemented in C++ and is available at http://dimacs.rutgers.edu/ mveralic
Contact: mveralic@math.rutgers.edu

1 INTRODUCTION

The complexity of biological systems is achieved through networks comprising thousands of interacting molecular species, including DNA, RNA, proteins, and smaller molecules. One of the goals of systems biology is thus to map these networks in ways that may provide fundamentally new understanding of cell biology at the molecular level (Stolovitzky *et al.*, 2007). Modern biotechnology can provide rich data sets by the simultaneous monitoring of thousands of different types of molecules and the challenge now is to unravel the complex interactions behind the data. Success in reverse-engineering the underlying network depends strongly on the characteristics of the modeling framework used, the availability of appropriate molecular data, and the level of noise present in the data (Noman and Iba, 2007).

Some reverse-engineering algorithms based on evolutionary computation have been proposed, e.g.,(Liu and Wang, 2008; Tsai and Wang, 2005; Kikuchi *et al.*, 2003; Repsilber *et al.*, 2002), and several studies have been performed of the robustness of these algorithms to noise (Noman and Iba, 2005; Sugimoto *et al.*, 2005). A typical drawback of evolutionary algorithms is the requirement of large amounts of computational resources. Some approaches have been proposed to overcome this issue by means of the use of some strategies such as problem decomposition, search space reduction, and *a priori* biological information on the network structure (Kimura *et al.*, 2005; Swain *et al.*, 2005; Tsai and Wang, 2005).

Because network inference is inherently ill-posed due to measurement error and the impact of latent variables, which are either not measurable or simply not included in the data, robustness to noise of inference methods is desirable (Xiao and Dougherty, 2007). The implicit assumption of noise-free measurements, made in many of the methods proposed, may lead to unreliable results (Lahdesmaki *et al.*, 2003). For some reverse-engineering methods their robustness to noise has been tested (Tegner *et al.*, 2003; Margolin *et al.*, 2006), and other methods have been proposed with the explicit goal of robustness in mind for linear models (van Someren *et al.*, 2001) and so-called noisy Boolean networks (BNs) (Akutsu *et al.*, 2000).

Boolean network models have a long tradition in biochemical network modeling, and several very sophisticated models have been published recently (*e.g.* Albert and Othmer, 2003; Cotta and Troya, 2003; Fangting *et al.*, 2004). In this type of model, the state of a molecular species is represented by a Boolean variable (ON or OFF), and interactions between the molecular species are represented by Boolean functions, which determine the presence or absence of a given molecular on the basis of the presence or absence of some others. Previous work, for instance in gene regulation networks, suggests that even when gene expression data are analyzed entirely in the binary domain, meaningful biological information can be successfully extracted (Shmulevich and Zhang, 2002; Tabus

*to whom correspondence should be addressed *et al.*, 2002). One of the appealing properties of BNs is that they are inherently simple, emphasizing generic network behavior rather than quantitative biochemical details, but are able to capture some of the complex dynamics present.

In this paper we introduce a reverse-engineering method based on an evolutionary algorithm that takes as input one or more time courses of (continuous) experimental measurements and possibly prior biological information and gives as output an optimized Boolean network as a model for the experimental system from which the data set was generated.

The paper is organized as follows. We first introduce our modeling framework and the reverse-engineering problem in this framework; then we discuss the algorithm proposed by first providing an overview of its basic features and a brief description of the different metrics incorporated in the fitness function of the evolutionary algorithm. We then validate the algorithm by applying it to data generated from a published model of the segment polarity gene network in D. Melanogaster; we develop a thorough study of the parameter space to reveal the ability of the algorithm to recover the network with different levels of noise applied to the input data. Then we generated Receiving Operator Curves (ROC) to compute their area under the curve (AUC) to measure algorithm performance under variation of several EA parameters at the different levels of noise added to the data.

2 THE ALGORITHM

A Boolean network can be described as follows. Let $k = \{0, 1\}$, with the arithmetic operations of addition and multiplication defined modulo 2, that is, $1 + 1 = 0$. Any Boolean function can then be described as a multivariate polynomial function on k, via the translation

$$x \wedge y = x \cdot y, x \vee y = x + y + x \cdot y, \neg x = x + 1.$$

A Boolean network on n variables is then a function

$$f = (f_1, \ldots, f_n) : k^n \longrightarrow k^n,$$

where each $f_i : k^n \longrightarrow k$ is a polynomial function in n variables. That is, if $\mathbf{x} = (x_1, \ldots, x_n) \in k^n$ is a state, then $f(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_n(\mathbf{x}))$. Since $x^2 = x$ in Boolean arithmetic, each f_i can be expressed uniquely as a polynomial in which each variable appears to the first power, that is, a square-free polynomial. Thus, a Boolean network is a special case of a finite polynomial dynamical system, the modeling framework considered for the reverse-engineering algorithm in (Laubenbacher and Stigler, 2004). Polynomial dynamical systems can be viewed as multistate generalizations of Boolean networks.

The reverse-engineering problem can then be formulated as follows. Given one or more time courses of state transitions generated by a biological system with n varying quantities, choose a family of Boolean networks $f : k^n \rightarrow k^n$ which best fit the data and "best describe" the biological system, according to a specified fitness function. It is assumed that a set of state transitions of the network is given in the form of one or more time courses of network states of the form

$$\mathbf{t}_1 = (t_{11}, \ldots, t_{n1}), \ldots, \mathbf{t}_m = (t_{1m}, \ldots, t_{nm}).$$

These may consist of wildtype and/or perturbation data, including those coming from knock-out mutants. After preprocessing the (continuous) experimental data by choosing an appropriate threshold we may assume that the data points consist of vectors with binary entries.

Since we want to optimize Boolean networks through an evolutionary algorithm context, they play the role of individuals in the population, or *chromosomes*, and each of the n coordinate polynomials represents a *gene*. A "gene" mutates by changing some of its terms. The algorithm takes as input a collection of discretized time courses, together with optional biological information (described in section **??**) and the output is a set of optimal polynomial models with respect to a given fitness function.

In addition to the input data set, the algorithm can also accept a collection of seed models. These can be used as the first population to initialize the algorithm. Seed models can consist, for instance, of interpolating polynomial models generated by other reverse-engineering methods, such as (Laubenbacher and Stigler, 2004; Dimitrova *et al.*, 2007). In the absence of "seed" models, the algorithm uses randomly generated models for the first generation.

If available, it is also possible to input prior information about the wiring diagram of the network, in the form of an $n \times n$ interaction matrix where a_{ij} denotes the probability of the existence of an edge from node j to node i in the static network.

Two additional strategies are employed to reduce the complexity of the search. Given the model framework, it is possible to optimize each coordinate function f_i separately and later assemble them into a polynomial model. This leads to a considerable reduction of the search space dimension. In addition, as shown in appendix 4, the use of computational algebra tools allows us to use only those terms for the purpose of mutation that result in a different "phenotype" of the new model on the input data. That is, we use only terms that do not vanish identically on the input data.

2.0.1 *Algorithm Summary*

Input: Discretized time courses and parameter set. Optional input: *seed* polynomial models and a priori network knowledge.

OutputPolynomial models with the best fitness scores

1. If no 'seed was provided, generate random polynomial models until the pool size is reached. Otherwise, use input models and complete pool size with randomly generated models;
2. Evaluate and sort models according to their fitness;
3. Subdivide *population* into n *sub populations*, each corresponding to one of the n coordinate functions;
4. Select polynomials from each subpopulation corresponding to each of the coordinate functions and assemble them into new polynomial models to form the generation's *offspring*;
5. Mutate the assembled candidate models with inverse probability to the fitness of the generation, i.e. fewer mutations will take place in a generation that shows higher fitness scores;
6. *Clone* polynomial models selected to be preserved in the next generation;
7. Build new population from candidate models, *cloned* models and randomly generated models;

Continue this process until either the specified number of generations is reached or until the fitness score has not improved for a pre-determined number of generations.

Figure 1 shows a schematic representation of the algorithm.

2.0.2 *The Fitness Function*

The fitness function used is a multi-objective function consisting of a weighted sum of the different optimality criteria. We explain here its main features; for details see the appendix. The fitness of a polynomial model is computed as the average of the fitness of its coordinate functions. Each coordinate function is evaluated based on three criteria:

- How well the coordinate function f_i is able to reproduce the i-th coordinate of the input data set. This is measured using Hamming distance.

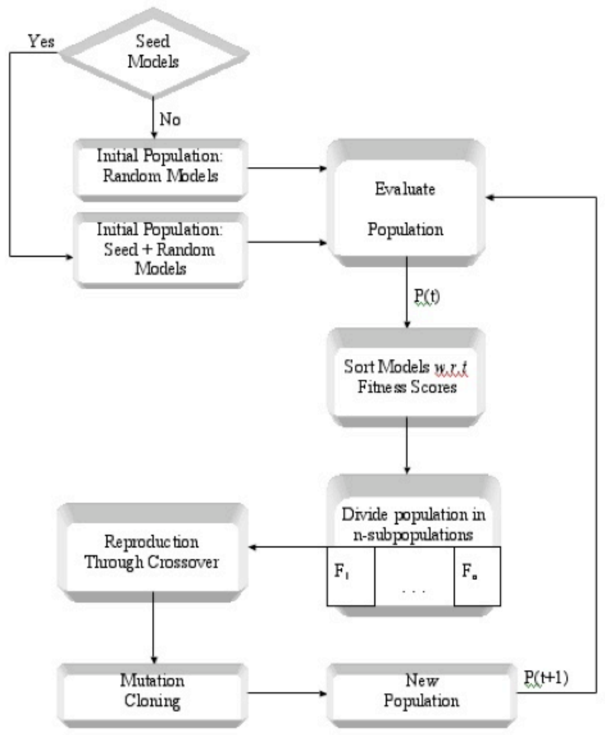
Seed Models
Yes
No
Initial Population: Random Models
Initial Population: Seed + Random Models
Evaluate Population
P(t)
Sort Models w.r.t Fitness Scores
Divide population in n-subpopulations
F1
. . .
Fn
Reproduction Through Crossover
Mutation Cloning
New Population
P(t+1)

Fig. 1. Evolutionary Algorithm Loop

- The complexity of the polynomial. This is measured as the average of the size of the support for all the terms in the polynomial. In other words, how many other variables influence a given variable through their appearance in the corresponding coordinate function?
- How well the part of the wiring diagram induced by the polynomial agrees with prior biological information, if applicable.

The output of the algorithm is the set of all polynomial models that have the highest fitness score.

3 RESULTS

3.1 Reverse Engineering of Segment Polarity Gene Network

Validation of reverse-engineering algorithms faces the challenge that for most available experimental data sets there is only incomplete knowledge of the underlying network that generated the data. One commonly chosen alternative is to use simulated data sets for which the network is known. For algorithms within a discrete modeling framework, such as Boolean or Bayesian networks, there is the additional challenge that performance depends on the choice of data discretization. To address both issues we validate the algorithm using a data set generated from a published Boolean model of the segment polarity gene network in *Drosophila melanogaster* . The segment polarity genes are responsible for pattern formation in the *D. melanogaster* embryo. In (Albert and Othmer, 2003), a Boolean model based on the binary ON/OFF representation of mRNA and protein levels of five segment polarity genes was proposed and analyzed. This model was constructed based on the known topology and it was validated using published gene expression data.

The expression of the segment polarity genes occurs in stripes that encircle the embryo, captured in the model rather as a one-dimensional representation that consists of a line of 12 interconnected cell, grouped into 3 parasegment primordia, in which the genes are expressed in every fourth cell. In (Albert and Othmer (2003)), parasegments are assumed to be identical, so only one parasegment of four cells is considered: the variables are the expression levels of the segment polarity genes and proteins (listed above) in each of the four cells. Hence, one stripe is represented as a $15 \times 4 = 60$ node network ; furthermore, by considering each cell as one subnetwork of 15 molecular species, with input from 6 external ones, a polynomial version of the network was introduced in (Laubenbacher and Stigler, 2004) reproduced in Table 3.1.

Table 1. Polynomial representation of the Boolean subnetwork of interest

$$
\begin{aligned}
F_1 &= x_1 \\
F_2 &= (x_{15}+1)[x_1x_{14}+x_2(x_{14}+x_1+x_1x_{14}] \\
F_3 &= x_2 \\
F_4 &= (x_{16}+x_{17}+x_{16}x_{17}(x_1+1) \\
F_5 &= x_4 \\
F_6 &= x_5(x_{15}+1) \\
F_7 &= x_6 \\
F_8 &= x_{14}(x_5+1)(x_{15}+1) \\
F_9 &= x_8+x_9(x_{18}+1)(x_{19}+1)+x_8x_9(x_{18}+1)(x_{19}+1) \\
F_{10} &= x_9(x_{18}+x_{19}+x_{18}x_{19}) \\
F_{11} &= [(x_9+1)+x_{52}+(x_9+1)x_{18}]+x_{19}+[(x_9+1)+ \\
&\quad x_{52}+(x_9+1)x_{18}]x_{19} \\
F_{12} &= (x_5+1) \\
F_{13} &= x_{12} \\
F_{14} &= x_{13}[(x_{11}+x_{20}+x_{11}x_{20})+x_{21}+(x_{11} \\
&\quad +x_{20}+x_{11}x_{20})x_{21}] \\
F_{15} &= x_{13}(x_{11}+1)(x_{20}+1)(x_{21}+1)
\end{aligned}
$$

$F_1 = CIR_i$	$F_2 = wg_i$	$F_3 = WG_i$	$F_4 = en_i$
$F_5 = EN_i$	$F_6 = hh_i$	$F_7 = HH_i$	$F_8 = ptc_i$
$F_9 = PTC_i$	$F_{10} = PH_i$	$F_{11} = SMO_i$	$F_{12} = ci_i$
$F_{13} = CI_i$	$F_{14} = CIA_i$	$F_{15} = CIR_i$	$F_{16} = WG_{i-1}$
$F_{17} = WG_{i+1}$	$F_{18} = HH_{i-1}$	$F_{19} = HH_{i+1}$	$F_{20} = hh_{i-1}$
$F_{21} = hh_{i+1}$			

Using this model, we generate several time courses which are used as input to the reverse-engineering algorithm.

3.1.1 Data Generation We generated 20 time courses, including wild-type and knock-out mutant data (obtained by setting the corresponding variable equal to 0), with a total of 202 time points ($<< 1\%$ of the 2^{21} possible states in the system).

Prior Knowledge. In order to simulate a real case study, we had considered only basic prior information to input in our algorithm:

- Biological information: this consist of only 5 dependencies in the static network from each one of the 5 genes in the network, with their corresponding products, and
- Prior static network information from reverse engineering: we used the algorithm from Jarrah *et al.*, 2007 and use their $(S1, T1)$ to obtain an interaction matrix of the static network.

EA Parameters. One main issue when applying evolutionary algorithms (or in general any heuristic algorithm) is the choice of appropriate parameters. Hence, we have randomly generated 60 sets of parameters from which we will analyze our results after running triplicates of these runs.

All information about data and parameters can be found as part of the Supplementary Data for this paper.

3.1.2 Numerical Results In order to perform a systematic evaluation of algorithm performance, we use the standard methods of Receiving Operator Curve (ROC). In Appendix 4 we give a description on this metric's performance adapted to the type of multi-parameter algorithms we use.

We study the performance of our method in three different settings:

- Case 1: No Noise

  For this first case, we verify that the proposed method performs well when inferring our network of interest when noise-free data is given. We generate triplicates for each one of our runs (60 runs from randomly generated sets of parameters), to have a total of 180 runs for noise-less data. In the Supplementary Data all the runs and their outputs are available.
- Case 2: Noise

  Biological data are typically noisy, particularly microarray data Hassibi and Vikalo, 2005. Although there are various techniques that increase the accuracy of microarray measurements, the data contain errors due to the probabilistic characteristics of the detection process, sample extraction, and mRNA purification to hybridization and imaging. Consequently, it is crucial to study how robust the proposed reverse engineering algorithm is to noise. The noise applied was sampled from a normal distribution with mean 0 and a standard deviation of 1% and 5% of the actual values. Since data discretization may filter some amount of noise contained in the experimental data, it is important to emphasize that in our setting, we added noise to our already discretized data, meaning a larger amount of noise than if we would have added the noise to the continuous data and then discretized. Here too we run computational experiments in triplicate for the different randomly generated parameters. Table 2 contains the best and worst results for all $3 \times 60 \times 3 = 540$ runs (60 different parameter sets, 3 noise levels, and triplicate computations).

  ¿From table 2 we observe that our method's performance does not degenerate as the noise level is increased, that is, our method is robust with respect to significant levels of noise in the data.
- Effect of Parameter Values

  The efficiency of evolutionary algorithms is greatly dependent on the parameters used. They are typically chosen through pre-training or tuning steps. We believe that there is value in a detailed study of the performance of different parameter sets, in order to understand the effect of the different optimization criteria on algorithm performance. We perform sensitivity analysis on the different parameters, in particular the parameters relating to the different weights in the fitness function.

  In this particular type of experiments, it is possible to calculate the Area Under the Curve (AUC) of the ROC curve to compare how close our method is to the perfect AUC value 1, for each one of the studied parameters. The ROC curves below the ROC points, including the points $(0, 0)$ and $(1, 1)$, were fit to a function $f_r(x) = \frac{ax+b}{x+c}$, using standard least squares techniques. The area under the curve is then estimated as $AUC = \int_0^1 f_r(x)dx$.

  We have carried out a sensitivity analysis on the individual parameters, using parameters sets as initializations, which give good results in both cases 1 and 2. Based on preliminary runs, we selected the parameter set tagged as 41 to make this parameter sensitivity study. As in the previous experiments, we develop our computations with the different levels of data noise, to observe that in all the cases, the different values of AUC vary between .86 and .88; in figure 2 we have displayed some of the ROCs generated with their corresponding AUCs.

Table 2. Results on EA Inference Method's Performance. Highest and lowest normalized ROC scores for all the 60 parameter sets for each one of the three different levels of noise.

Noise Level	ROC Score	1−Specificity	Sensitivity	Param
0% Best R1	.8390	0.103943	0.77778	41
0% Worst R1	.7617	0.200717	0.72222	34
0% Best R2	.8330	0.096774	0.75000	55
0% Worst R2	.7679	0.121186	0.63888	13
0% Best R3	.8257	0.082437	0.72222	11
0% Worst R3	.7597	0.136201	0.63888	58
.01% Best R1	.8428	0.096774	0.77778	07
.01% Worst R1	.7658	0.125448	0.67388	09
.01% Best R2	.8389	0.129032	0.80556	31
.01% Worst R2	.7536	0.168459	.666667	10
.01% Best R3	.8333	0.114695	0.77778	29
.01% Worst R3	.7636	0.150538	0.66666	31
.05% Best R1	.8540	0.100358	0.805560	37
.05% Worst R1	.7620	0.132616	0.638889	58
.05% Best R2	.8352	0.111111	0.777780	39
.05% Worst R2	.7468	0.204301	0.694444	31
.05% Best R3	.8321	0.093190	0.750000	42
.05% Worst R3	.7661	0.168459	0.694444	47

4 CONCLUSIONS

We have proposed an effective evolutionary algorithm for inferring gene regulation networks. Beginning with time course experimental data and available prior biological knowledge, the algorithm outputs a dynamic model of the network, rather than just a static wiring diagram. The algorithm was validated with simulated gene expression

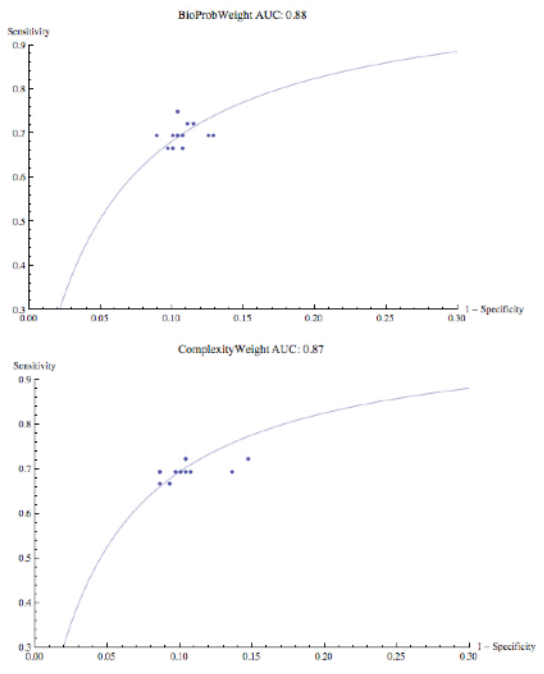

Fig. 2. ROC curves and their corresponding areas (AUC) with .05 data noise for two of the studied parameters: Prior Biological Knowledge and Complexity Weight Parameters

data, generated using a published Boolean network model. It is possible to apply the algorithm to each node of the network separately, thereby greatly reducing the complexity of the problem. Another tool we employ to reduce the complexity of the search problem is a result from computational algebra that provides a computationally efficient characterization of those polynomial terms that change the phenotype of the model on the given data set, thereby limiting significantly the evolutionary space of the algorithm. We found that the algorithm is robust with respect to significant levels of experimental noise.

One the input options is information about the static wiring diagram. One source of such information is biology. Another possible scenario providing such information is through application of other reverse-engineering methods to data that are not suitable as input for our algorithm, for instance steady-state data (*e.g.* Andrec *et al.*, 2005.

It is worth mentioning that the algorithm presented here can be generalized to multistate discrete dynamical models. One aspect of the algorithm that limits its performance on multistate models is that the characterization of polynomial terms that induce a different phenotype on the given input data fails to characterize all allowable mutations. It is an open problem in computational algebra to find a complete characterization. (See Appendix 4.)

ACKNOWLEDGEMENT

The first and fifth authors have been partially supported by NIH Grant Nr. RO1GM068947-01. The authors thank P. Mendes and B. Stigler for contributing insights in the problem. We thank Bill Marragas from the Advanced Research Computing (ARC) at Virginia Tech for providing access to the high performance computing platform SGI Enterprise ALTIX 3700 Supercluster at Virginia Tech.

APPENDIX

COMPUTATIONAL ALGEBRA TOOLS: MAXSUPPORT

In Section 2, we saw that given a time course $t_1, \ldots, t_m$ of length m of state transitions with n varying quantities, the reverse engineering problem requires finding a function $f : k^n \to k^n$ satisfying $f(t_i) = t_{i+1}$. It was also shown that we can reduce this problem to finding each coordinate (Boolean) interpolating function $f_j : k^n \to k$ satisfying

$$f_j(t_i) = t_{ij}, \quad \text{for each } 1 \le j \le n \text{ and } 1 \le i \le m. \qquad (4.1)$$

Furthermore, each interpolating function can be uniquely expressed as a square-free polynomial in the ring $R = k[x_1, \ldots, x_n]$ of polynomials in n variables with coefficients in k. Given an interpolating function f satisfying the conditions (4.1), any other interpolating function satisfying the same conditions can be written as $f + g$ for some polynomial g in R satisfying $g(t_i) = 0$, for each $1 \le i \le m$. The reverse engineering problem consists in choosing a suitable interpolating function $f + g$ that "best describes" the dynamics of the biological system.

Given a time course $t_1, \ldots, t_m$ of length m, let $I = I(t_1, \ldots, t_m)$ denote the set of all polynomials g in R satisfying $g(t_i) = 0$, for each $1 \le i \le m$. The set I is an ideal in the polynomial ring R, the so-called *ideal of points* $t_1, \ldots, t_m$. In this context, finding an interpolating function $f + g$ corresponds to choosing a representative of the class $[f]$ in R/I, where

$$R/I = \{h + I \mid h \in R\} \text{ and } h + I = \{h + g \mid g \in I\}.$$

In the 1-variable case, this is accomplished by finding the remainder r upon the Euclidean division algorithm of f modulo the unique (up to sign) generating polynomial g of I. This result extends to several variables, but in this case, different orderings in the monomials of R may result in distinct remainders (In 1 variable there is only one sensible term order $1 < x < x^2 < \cdots$ in two variables we need to decide whether $x < y$ or $x > y$ while performing the division algorithm).

Any monomial $x^a = x_1^{a_1} \cdots x_n^{a_n}$ in R can be identified with a lattice point $a = (a_1, \ldots, a_n)$ in $\mathbb{N}^n$. A *term order* $<$ on $\mathbb{N}^n$ is a total order where the zero vector 0 is the unique minimal element and

$$a < b \text{ implies } a + c < b + c, \quad \text{for all } a, b, c \in \mathbb{N}^n.$$

Given a term order $<$, every non-zero polynomial $f \in R$ has a unique *initial monomial*, denoted $\text{in}_<(f)$. If I is an ideal in R, then its *initial ideal* is the monomial ideal

$$\text{in}_<(I) = \langle \text{in}_<(f) \mid f \in I \rangle.$$

The monomials which do not lie in $\text{in}_<(I)$ are called *standard monomials.* The following known result is crucial in our development.

PROPOSITION 4.1. *The (images of the) standard monomials, denoted $\mathcal{B}_<$ form a k-vector space basis for the residue ring R/I.*

A nice consequence of this result is that every ideal I in R has only finitely many distinct initial ideals (and hence finitely many distinct sets $\mathcal{B}_<$), see Sturmfels, 1995. Let $\Lambda(I)$ denote the union of all distinct sets $\mathcal{B}_<$. Then any representative of $[f]$ under any term ordering is a linear combination of monomials in $\Lambda(I)$. Therefore, the universe of monomials in the E.A. ought to be $\Lambda(I)$. Unfortunately, there is still no simple criterion to check if a given monomial is in $\Lambda(I)$.

FITNESS FUNCTION METRICS

The fitness function for the EA is a multi-objective function consisting on the weighted sum of the different criteria for model optimality. Below is the description of each one of these criteria:

Model Data Fit. The ability of a candidate model M to reproduce input time courses is measured with the use of the Hamming distance. Let α_i be the length (or number of time steps) of the time course t_j, and let t'_j be the time course generated by iterating the model M until $t'_j = \alpha_i$.

In the particular case of a knockout time courses, let's say corresponding to the j^{th} variable, we iterate the model M but editing the j^{th} polynomial to zero.

Let

$D = |\{$number of entries where $t_j \neq t'_j\}|$ (number of places where $T_s \neq T'_s$

$\alpha = \sum_{j=1}^{n} \alpha_j$ (the total number of time steps or measurements).

The total Hamming distance H_M between the input time series $T_s = \{t_1, t_2, , t_s\}$ and $T'_s = \{t'_1, t'_2, , t'_s\}$ is:

$$H_M = \frac{D}{n \times \alpha} \tag{4.2}$$

Hence the model data fit score ($Model_{FIT}$) of M is given as:

$$Model_{FIT}(M) = W_H(1 - H_M) \tag{4.3}$$

where W_H is the weight assigned to the model data fit.

Assuming the existence of noise in the data, this data fit score can then accept models with scores below 1.

Polynomial Data Fit. This score forms part of the problem decomposition strategy; we can evaluate the data fit of a model component-wise, that is, one polynomial at a time. The Hamming distance for the i^{th} polynomial in M has for objective to evaluate the polynomial's ability to reproduce the i^{th} column of all the input time courses in T_s; in order to achieve this evaluation, we need to"isolate" the polynomial's performance independently of the other polynomials. Hence, we fix all the values of the j^{th} columns of the input time series, where $j \neq i$. That is, the Hamming distance of the i^{th} polynomial is given as :

$$H_{f_i} = \frac{D_i}{n \times \alpha} \tag{4.4}$$

where

$D_i = |\{$number of entries where $t_i^j \neq t_i^{j'}\}|$, the number of places where the j^{th} columns of T_s and T'_s differ

$\alpha = \sum_{j=1}^{n} \alpha_j$ (the sum of the number α_j of time steps on t_j).

Therefore the polynomial data fit score ($Polynomial_{FIT}$) of f_i is given by:

$$Polynomial_{FIT} = (1 - H_{f_i}) \tag{4.5}$$

Now, in order to evaluate the polynomial-wise performance for the whole model, we take the average of the n $Polynomial_{FIT}$ scores of the given model times W_{Hp}, the weight assigned to the polynomial time course reproducibility score.

Complexity score. It results of great relevance, the balancing of network's ability to explain the observed data with its ability to do so simply; scoring metrics with a penalty for unnecessary complexity are able to guard against the over-fitting of network models to observed data (Yu *et al.*, 2004).

The complexity score that we have developed uses some computational algebra results that have been developed in appendix 4 and that we have resumed in the search space reduction part of section refstrategies. The complexity score is then be given by:

$$\text{Complexity Ratio }(f_i) = \frac{MaxTotalDegree(\{m \in f_i\})}{MaxSupport}$$

where $MaxSupport := \sum_{j=1}^{s} d_j$, j=1,...,s.

Since the polynomial complexity score must reflect the simplicity of structure of the monomials, we chose constructed a probability distribution curve that allows to penalize complexity without giving preference to constant terms:

$$\text{PolyCompl }(f_i) = \text{Compl Distrib Func (Complexity Ratio }(f_i)$$

Therefore, the average Model Complexity Score is given by:

$$\text{Model Complexity} = W_{Cx} \sum_{i=1}^{n} \frac{\text{PolyCompl }(f_i)}{n} \tag{4.6}$$

where W_{Cx} is the weight assigned to the complexity score for a model, in the input set of parameters.

A Priori Biological Knowledge Score. When a *priori* biological knowledge about the existence (or absence) of links in the wiring diagram of the network is available, we have incorporated a scoring to measure the ability of a model to reproduce these partial pieces of the wiring diagram.

Let's consider the $n \times n$ input matrix BioProbScore Matrix $(a)_{ij}$, where the $a_i j$ represents the certainty (value between 0 and 1) that in the wiring diagram of the network, there is a causal relationship from node j to node i

Let $V(f_i) = (v_1, ..., v_j, ...v_n)$ be the n-tuple of 0 and 1's corresponding to the absence or existence, respectively, of the variable x_j in the polynomial f_i.

Let $(p_{i1}..., p_{in})$ be the i^{th} row of the BioProbScore Matrix corresponding to the probability p_{ji} that the variable x_j appears in the polynomial f_i (i.e. that x_i depends on x_j), and let

$$(q_{i1}..., q_{in}) := (p_{i1} \wedge v_1, \cdots, p_{in} \wedge v_n)$$

and let

$$\alpha_{ij} := \begin{cases} 1 - p_{ij} & \text{if } v_j = 0 \\ p_{ij} & \text{otherwise} \end{cases}$$

Hence the biological score assigned to the polynomial f_i is given by:

$$\text{BioScore }(f_i) = \frac{1}{n} \sum_{j=1}^{n} \alpha_{ij} \qquad (4.7)$$

Therefore, the BioScore assigned to the model M is:

$$\text{BioScore } M = W_B \sum_{i=1}^{n} \frac{\text{BioScore }(f_i)}{n} \qquad (4.8)$$

where W_B is the weight assigned to the Model BioScore, in the input set of parameters.

It is important to notice that according to the scoring method, for those unknown causal interactions we can assign the value 0.5 in order to avoid penalizing lacking of a priori knowledge as well.

Reverse Engineering score. Analogously to the scoring process for the a priori biological knowledge, we can construct an $n \times n$ input matrix that contains knowledge obtained from a previous used reverse engineering method about the structure of the wiring diagram. Considering the advantage that represents to have all the information about all possible structures of wiring diagrams that explain the data, we have made use of the method described in (Jarrah *et al.*, 2007).

METHOD'S PERFORMANCE METRIC

For an unbiased evaluation of the performance of network inference methods it is common to quantify: Correct interactions inferred (true positives, TP), Incorrect interactions inferred (false positives, FP), Correct non-interactions inferred (true negatives, TN) and Incorrect non-interactions inferred (false negatives FN).

The number of TPs, TNs, FPs, and FNs can be estimated from the *confusion matrix.* This is a 2×2 matrix that provides an assessment of the classification procedure, or how "confused" the algorithm might be, by simply counting the interactions discovered. From the confusion matrix we compute

- Sensitivity SEN = TP / (TP + FN) in order to measure the proportion of recovered true edges
- Specificity SPE = TN / (TN + FP) which complement (*i.e.* 1-SPE) measures the proportion of erroneously recovered spurious edges.

Plotting the ensuing sensitivity against the corresponding complementary specificity gives the Receiver Operator Characteristic which area under the curve (AUC) measures the effectiveness of the network inference algorithm (area value from 0 to 1, where 1 corresponds to a perfect network inference). ROC curves are then plotted by $(1 - SPE, SEN)$ values of the networks inferred by a method when a given parameter value of the algorithm is varied (see study done insection 3.1.2)

However, when several parameters are involved in the inference algorithm, such as the case of the EA part of our network inference, the plotting of the ROC curves are difficult to interpret since they involved several parameter values changed simultaneously; hence, in order to perform our study for when several parameters are changed simultaneously, we use compare the magnitude of the vectors given by the $(1 - SPE, SEN)$ and placing the origin at the $(1, 0)$ coordinate; the measurements of the magnitudes then from 0 to $\sqrt{2}$ represent the extreme cases for coordinates $(1 - SPE, SEN) = (1, 0)$ and $(1 - SPE, SEN) = (0, 1)$ representing the worst and best possible predictions, respectively (see sections 3.1.2 and 3.1.2).

REFERENCES

Akutsu, T., Miyano, S., Kuhara, S. (2000) Inferring qualitative relations in genetic networks and metabolic pathways, *Bioinformatics*, **16**, 727734.

Albert,R. (2004) Boolean modeling of genetic regulatory networks, *Lec. Notes Phys. Springer-Verlag*, **650**, 459-481.

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

Allen,E., Fetrow,J., Daniel,S., Thomas,S., John,D. (2006) Algebraic dependency models of protein signal transduction networks from time-series data, *Journal of Theoretical Biology*, **238**, 317330.

Andrec,M., Kholodenko,B.N., Levy,R.M., Sontag,E.D. (2005) Inference of signaling and gene regulatory networks by steady-state perturbation experiments: structure and accuracy, *Journal of Theoretical Biology*, **232**(3), 427-441.

Babson,E., Onn,S., Thomas,R. (2003) The Hilbert zonotope and a polynomial time algorithm for universal Gröbner bases, *Advances in Applied Mathematics*, **30**, 529-544.

Baeck,T., Fogel,D.B., Michalewicz,Z. (2000) Evolutionary Computation 1: Basic Algorithms and Operators (Evolutionary Computation), *Taylor and Francis*, 1st edition.

Bardet,M., Faugère,J.C., Salvy,B. (2003) Complexity of Gröbner basis computation for Semi–regular Overdetermined sequences over $GF(2)$ with solutions in $GF(2)$, Research Report no. 5049, Inria, 19 pages.

Cotta,C., Troya, J. (2003) Reverse engineering of temporal Boolean networks from noisy data using evolutionary algorithm, *Neurocomputing*, **62**, 111-129.

Cull,P. (1971) Linear analysis of switching nets, *Kybernetik*, **8**, 3139.

D'Haeseleer,P., Liang,S., Somogyi,R. (2000) Genetic network inference: From co-expression clustering to reverse engineering, *Bioinformatics*, **16(8)**, 707-726.

Dimitrova,E., Jarrah,A., Laubenbacher,R., Stigler,B. (2007) A Groebner-fan-based method for biochemical net- work modeling. In: Brown, C. W. (ed.), ISAAC07: Proceedings of the 2007 international Symposium on Symbolic and Algebraic Computation, 122126. ACM, NewYork.

Elspas,B. (1959) The theory of autonomous linear sequential networks, *IRE Transactions on Circuit Theory*, **6**, 4560.

Fangting,L., Tao,L., Ying,L., Qi,O., Chao,T. The yeast cell-cycle network is robustly designed, *Proceedings of the National Academy of Sciences*, **101**, 4781-4786.

Hassibi,A. and Vikalo,H. (2005) Probabilistic modeling and estimation of gene expression levels in microarray, *Proc. IEEE Workshop on Genomic Signal Processing and Statistics (GENSIPS).*

Jarrah,A., Laubenbacher,R., Stigler,B., Stillman,M. (2007) Reverse Engineering of Polynomial Dynamical Systems, *Advances in Applied Mathematics*, **39(4)**, 477-489.

Kikuchi, S., Tominaga,D., Arita,M., Takahashi,K., Tomita,M. (2003) Dynamic modeling of genetic networks using genetic algorithm and S-system, *Bioinformatics*, **19(5)**, 643-650.

Kimura,S., Ide,K., Kashihara,A., Kano,M., Hatakeyama,M., Masui,R., Nakagawa,N., Yokoyama,S., Kuramitsu,S., Konagaya,A. (2005) Inference of S-system models of genetic networks using cooperative coevolutionary algorithm, *Bioinformatics*, **21(7)**, 1154-1163.

Lahdesmaki,H., Shmulevich,I., Yli-Harja,O. (2003) On Learning Gene Regulatory Networks Under the Boolean Network Model, *Machine Learning*, Kluwer Academic Publishers, —bf 52, 147-167.

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

Liu,P.K., Wang,F.S. (2008) Inferenceof biochemical network models in S-system using multi-objective optimization approach, *Bioinformatics*, **24**(8):1085-92.

Marchand,H. and LeBorgne,M. (1998) On the optimal control of polynomial dynamical systems over Z/pZ. Proceedings of the Fourth Workshop on Discrete Event Systems (Cagliari, Italy), *IEE*, pp. 385390.

Margolin,A., Wang,K., Lim,W.K., Kustagi,M., Nemenman,I., Califano,A. (2006) Reverse engineering cellular networks, *Nature Protocols* , **1**, 662 - 671.

The behavior of affine Boolean sequential networks, *Connection Science*, **5(2)**, 153167.

Noman,N., Iba,H. (2005) Reverse engineering genetic networks using evolutionary computation, *Genome Informatics*, **16(2)**, 205-214.

Noman,N., Iba,H. (2007) Inferring Gene Regulatory Networks using Differential Evolution with Local Search Heuristics, *IEEE/ACM Transactions on Computational Biology and Bioinformatics* , **4**, 634-647.

Repsilber,D., Liljenstrom,H., Andersoon, G.E. (2002) Reverse engineering of regulatory networks: simulation studies on a genetic algorithm approach for ranking hypotheses, *BioSystems*, **66**, 31-41.

Shmulevich, I., Zhang, W. (2002). Binary analysis and optimization-based normalization of gene expression data. *Bioinformatics*, **18**, 555565.

Stolovitzky,G., Monroe,D., Califano,A. (2007) Reverse engineering biological networks: opportunities and challenges in computational methods for pathway inference , *Ann. N.Y. Acad. Sci.*, **1115**, 1-22.

Sturmfels, B. (1995) Gröbner bases and Convex Polytopes , *AMS University Lecture Series*, **8**.

Sturmfels, B. (2002) Solving Systems of Polynomial Equations, *CBMS Regional Conference Series in Mathematics*, **97**.

Sugimoto,M., Kikuchi,S., Tomita,M. (2005) Reverse engineering of biochemical equations from time-course data by means of genetic programming, *ByoSystems*, **80**, 155-164.

Swain,M., Hunniford,T., Dubitzky,W., Mandel,J., Palfreyman,N. (2005) Reverse engineering gene regulatory networks using evolutionary algorithms and grid computing, *Clinical Monitoring and Computing*, **19**, 329-337.

Tabus, I., Rissanen, J., Astola, J. (2002) Normalized maximum likelihood models for Boolean regression with application to prediction and classification in genomics. In W. Zhang, and I. Shmulevich (Eds.), Computational And Statistical Approaches To Genomics. Boston: *Kluwer Academic Publishers*.

Tegner,J., Yeung, S., Hasty,J., Collins,J. (2003) Reverse engineering gene networks: Integrating genetic perturbations with dynamical modeling, *PNAS*, **100(10)**, 5944 5949.

Tsai,K.Y., Wang,F.S. (2005) Evolutionary optimization with data collocation for reverse engineering of biological networks, *Bioinformatics*, **21(7)**, 1180-1188.

van Someren,E.P., Wessels,L.F.A., Reinders,M.J.T., Backer,E. (2001) Robust genetic network modeling by adding noisy data, *Proceedings of the 2001 IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing*, Baltimore, MA.

Xiao,Y., Dougherty, E. (2007) The impact of function perturbations in Boolean networks, *Bioinformatics*, **23(10)**, 1265-1273.

Yu,J., Smith,J., Hartemink,A., Jarvis,E.D. (2004) Advances to Bayesian network inference for generating causal networks from observational biological data, *Bioinformatics*, **20(18)**, 3594-3603.