Using Gröbner bases to reverse-engineer biochemical networks

Winfried Just[1,2] and Brandilyn Stigler[1]

1) Mathematical Biosciences Institute, Ohio State University, Columbus, Ohio, U.S.A.

2) Department Mathematics, Ohio University, Athens, Ohio, U.S.A.

This material is based upon work supported by the National Science Foundation under Agreement No. 0112050.

Biochemical networks

- gene regulatory networks
- metabolic networks
- signal transduction networks
- the number n of chemicals in such networks can be of order 10^3 or even 10^4

Network dynamics

Concentration of chemicals in a network change over time.
Future concentration levels are determined by the present concentration vector.

Thus networks can be modeled as dynamical systems, either as

- continuous flows, or

- finite dynamical systems when both time and concentration levels have been suitably discretized.

Reverse engineering

Contemporary experimental methods make it possible to collect data on the concentration levels of all chemicals in a network simultaneously; even for networks with $n > 10^4$ chemicals.

In *reverse engineering* of a network, one tries to answer the following questions, simultaneously for all chemicals in the network:

1. Which chemicals directly influence the future concentration levels of chemical i? *(network topology)*

2. How to compute future concentration levels of chemical i from the present concentration vector?
   (regulatory functions)

Polynomial dynamical systems

The data collected on large networks are snapshots in time and typically noisy. Thus at present the most promising modeling paradigm for reverse engineering is treating the network as a **finite dynamical system** with state space F^n for some finite set F, and updating function $H = (h_1, \ldots, h_n)$.

If the cardinality of F is a prime number, then we can treat F as a finite field, and every component h_i of the updating function becomes a polynomial in $F[x_1, \ldots, x_n]$.

In this case the (unknown) true model of the network becomes a *polynomial dynamical system (PDS).*

Models of regulatory functions

Let $[m] := \{1, \ldots, m\}$. The data for reverse engineering one regulatory function h_i take the form

$$D = \{< \bar{x}(t), y(t) >: \ t \in [m]\}.$$

Any polynomial function $h \in F[x_1, \ldots, x_n]$ with $h(\bar{x}(t)) = y(t)$ for all $t \in [m]$ will be called a *model for* D.

Note that D has $|F|^{|F|^n - m}$ models.
Enumerating all models is prohibitive in most cases, and we need reverse-engineering algorithms that *select* one of the many models consistent with the data.

The set of models

Let

$$I_D = \{g \in F[x_1, \ldots, x_n] : \forall t \in [m]\, g(\bar{x}(t)) = 0\}.$$

Then I_D is an ideal in $F[x_1, \ldots, x_n]$.

Lemma 1 (Laubenbacher-Stigler)**.** *Let D be a data set and h a model for D. Then the set of all models for D is the set $h + I_D$.*

Review: Term orders

A *term order* is a well-order $\prec$ on the set of all monomials in $F[x_1, \ldots, x_n]$ such that $x^\alpha \prec x^\beta$ implies $x^\alpha x^\gamma \prec x^\beta x^\gamma$ for all multiexponents α, β, γ.

Examples

1. *lex(icographical)* orders generated by a variable order $x_1 \prec x_2 \prec \cdots \prec x_n$

2. *graded* term orders, in which $\sum \alpha < \sum \beta$ always implies $x^\alpha \prec x^\beta$.

For the variable order above, $x_1 x_2 \prec_{lex} x_3$, but for any graded term order, $x_3 \prec x_1 x_2$.

Review: Gröbner bases

For any ideal $I \in F[x_1, \ldots, x_n]$ and term order $\prec$, there exists a basis $G_\prec$ for I called a *(reduced) Gröbner basis.*

This basis has the property that for every $f \in F[x_1, \ldots, x_n]$ there exists a unique polynomial $f\%G_\prec$, called the *normal form of f with respect to $G_\prec$*, such that

$$f - f\%G_\prec \in I$$

and no term of $f\%G_\prec$ is divisible by the leading term of any polynomial in $G_\prec$.

The LS-algorithm

The reverse-engineering algorithm developed by Laubenbacher and Stigler (LS-algorithm) takes as **input a data set** D for one regulatory function h^* and a term order $\prec$, computes one model h for D and a Gröbner basis $G_\prec$ for I_D, and **returns the normal form** $h\%G_\prec$.

This output is the most parsimonious model for D in the sense specified by $\prec$.

Laubenbacher and Stigler report a success rate of 82% of their algorithm on a synthetic data set generated from a well-characterized network in *Drosophila melanogaster.*

The first bottleneck: Data requirements

In typical reverse-engineering problems, $m << n$.
So the question arises:

Question 2. *How many data points are needed on average before we can expect a given reverse-engineering algorithm,* e.g., *the LS-algorithm, to return the correct model?*

To turn this into a mathematical question, we will assume that the data inputs $\bar{x}(t)$ are chosen randomly and independently from the set F^n of all possible data inputs.

The expected number of data points needed

Now assume h^* is the true, but unknown regulatory function for a chemical in our network, and h^* depends on k of the n variables. In practice, k will be small, so we treat it as fixed.

$E(h^*, \prec)$:= the expected number of data with random inputs that need to be collected before the LS-algorithm with parameter $\prec$ returns h^*.

We are interested in the question how $E(h^*, \prec)$ scales with the number n of chemicals in the network.

Theorems about $E(h^*, \prec)$

Theorem 3 (Just)**.** *If* $\prec$ *is a graded term order, then* $E(h^*, \prec)$ *scales as a polynomial in* n *and it doesn't matter much which particular graded order* $\prec$ *is used.*

Theorem 4 (Just)**.** *If* $\prec$ *is a randomly chosen lex order, then* $E(h^*, \prec) > c^n$ *for some constant* $c > 1$*.*

Theorem 5 (Just)**.** *If* $\prec$ *is an optimally chosen lex order, then* $E(h^*, \prec)$ *scales as* $log\, n$*. This performance is achieved by a recent modification of the LS-algorithm developed by Jarrah, Laubenbacher, Stigler, and Stillman.*

The second bottleneck: Run-time

The LS-algorithm requires computing a Gröbner basis for the ideal in $F[x_1, \ldots, x_n]$ of a variety with m points. The standard method via the Buchberger-Möller Algorithm (BMA) has a time-complexity of $O(n^2 m^3)$.

To date no algorithms had been know that were optimized for the case when $m << n$, which is typical for experimental data on biochemical networks.

Just and Stigler developed a new algorithm, called the *EssBM algorithm*, that computes reduced Gröbner bases in running time $O(nm^3 + m^6)$. Empirical comparisons show that for $n > 200$ our algorithm runs **faster** than the standard implementation of the BMA when m is small (between 5 and 15).

The EssBM algorithm in a nutshell

The EssBM algorithm considers the variables one at a time and classifies them as either *essential* or *inessential* variables.

For each inessential variable, one polynomial that has this variable as leading term is added to the Gröbner basis.

The BMA is called at most m times to identify additional elements of the Gröbner basis that are relations between the essential variables.

Directions of ongoing research

- Investigate the performance of the LS-algorithm on data sets with nonrandom data inputs.

- Investigate data requirements for other reverse engineering algorithms.

- Further improve the run-time of the EssBM algorithm (we have another algorithm, but need to implement it).

- Modify the LS-algorithm so that it preferentially finds models that are similar to regulatory functions in experimentally characterized biochemical networks.

References

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

Just, W. (2006).
Reverse engineering discrete dynamical systems from data sets with random input vectors. *To appear in J. Comput. Biol.*

Jarrah, A., Laubenbacher, R., Stigler, B. and Stillman, M. (2006). Reverse-engineering of polynomial dynamical systems. *Submitted.*

Just, W. and Stigler, B. (2006).
Computing Groebner Bases when $m << n$. *Submitted.*