Algebraic Reverse Engineering of Biological Networks

Mike Stillman (mike@math.cornell.edu)

Department of Mathematics
Cornell

22 May 2015 / UConn Health

Includes work with or by Laubenbacher, Stigler, Jarrah, Dimitrova, Vera-Licona, Veliz-Cuba, Murrugarra, Aguilar, Hoops, Arat, Ha and others.

Polynomial Dynamical Systems

Pioneered by Laubenbacher and Stigler (2004).

Given a biological network with n nodes (each node corresponds to some biological product, often expression levels of genes)

Polynomial dynamical systems

- Let $[p] = \{0, 1, 2, \ldots, p-1\}$ be a finite set. This will be the possible states of each node.
- Model the dynamics of the system by a finite dynamical system :

$$\phi = (f_1, \ldots, f_n) : [p]^n \longrightarrow [p]^n$$

- If p is a prime number (or power of one), then every ϕ can be represented by polynomials $f_i \in k[x_1, \ldots, x_n]$, where $k = \mathbb{Z}/p$.

Large literature on boolean dynamical systems (p=2).

Power of this method is availability of powerful methods in computational algebra. And : why I got excited and involved.

Goals

Our goal

Given discretized time series data (to p possible states for each node), and possibly knockout data, and possibly other biological information, determine

- Possible **wiring diagrams** : A wiring diagram is a directed graph with vertices x_i, where edges incoming to a vertex x_i correspond to the variables which appear in f_i.
- Possible dynamics : the actual functions f_i.

As a mathematician, my personal goal is : total enlightenment and determination of the regulatory network.

But : the actual goal is to help to cheaply narrow down what are the most likely edges in the wiring diagram. Because : wet lab experiments are expensive and time consuming.

Goals

Our goal

Given discretized time series data (to p possible states for each node), and possibly knockout data, and possibly other biological information, determine

- Possible **wiring diagrams** : A wiring diagram is a directed graph with vertices x_i, where edges incoming to a vertex x_i correspond to the variables which appear in f_i.
- Possible dynamics : the actual functions f_i.

As a mathematician, my personal goal is : total enlightenment and determination of the regulatory network.

But : the actual goal is to help to cheaply narrow down what are the most likely edges in the wiring diagram. Because : wet lab experiments are expensive and time consuming.

Goals

Our goal

Given discretized time series data (to p possible states for each node), and possibly knockout data, and possibly other biological information, determine

- Possible **wiring diagrams** : A wiring diagram is a directed graph with vertices x_i, where edges incoming to a vertex x_i correspond to the variables which appear in f_i.
- Possible dynamics : the actual functions f_i.

As a mathematician, my personal goal is : total enlightenment and determination of the regulatory network.

But : the actual goal is to help to cheaply narrow down what are the most likely edges in the wiring diagram. Because : wet lab experiments are expensive and time consuming.

Goals

Our goal

Given discretized time series data (to p possible states for each node), and possibly knockout data, and possibly other biological information, determine

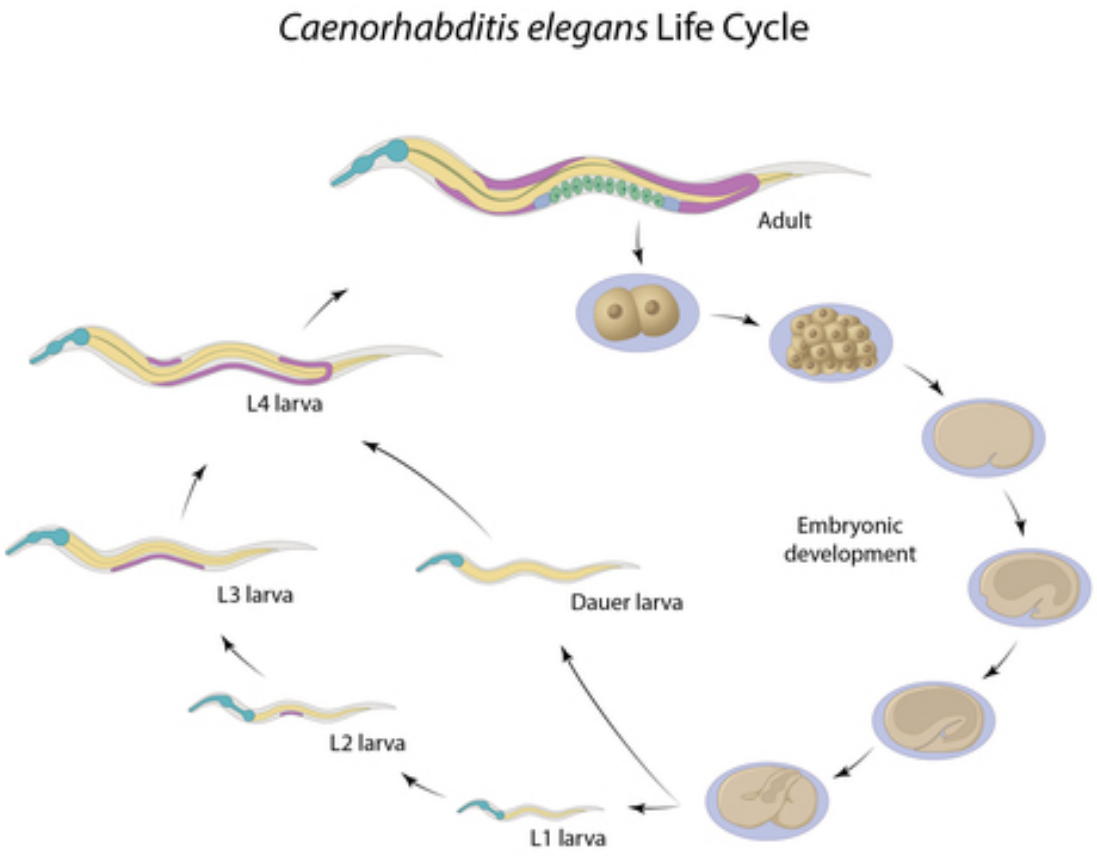
- Possible **wiring diagrams** : A wiring diagram is a directed graph with vertices x_i, where edges incoming to a vertex x_i correspond to the variables which appear in f_i.
- Possible dynamics : the actual functions f_i.

As a mathematician, my personal goal is : total enlightenment and determination of the regulatory network.

But : the actual goal is to help to cheaply narrow down what are the most likely edges in the wiring diagram. Because : wet lab experiments are expensive and time consuming.

C.elegans : an example (Stigler-Chamberlin 2012)

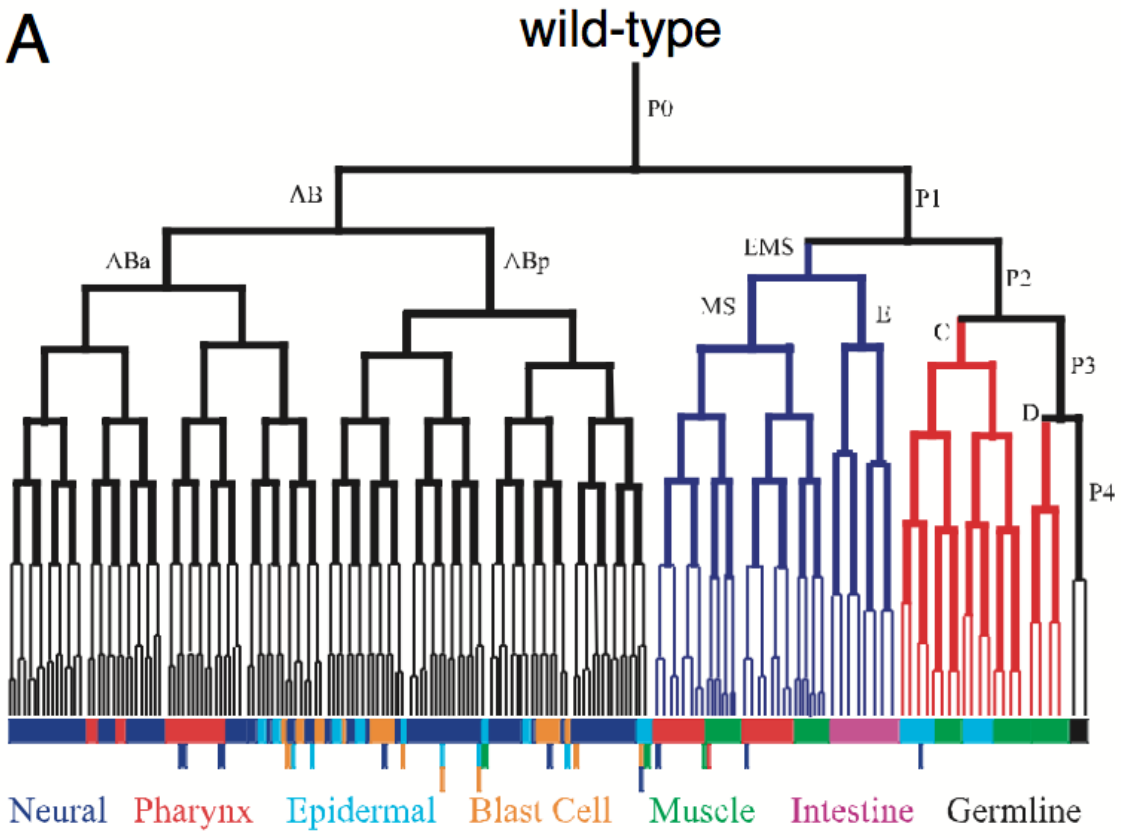
Consider the nemotode C. elegans.



(random picture from web)

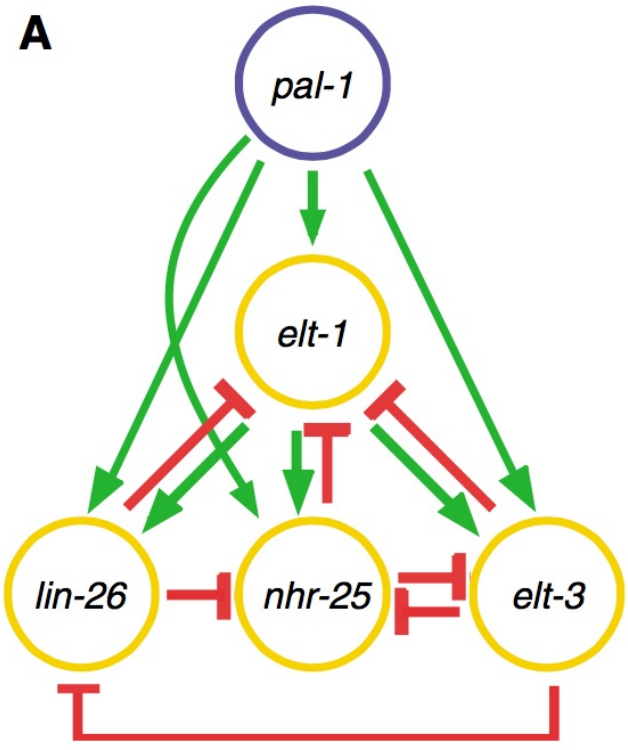
During embryonic development, the egg cell splits a number of times, and after maybe 4 splits (to 16 cells) : the cells start getting differentiated into different cell types.

C. elegans : "C" type cells



Some of the "C" type cells eventually become muscle (mesoderm) cells, some become skin (ectoderm) cells. The gene PAL-1 plays a strong role regulatory role in this differentiation. (Picture from Baugh et al, 2005)

C. elegans : (probable) ectoderm regulatory genes



A potential wiring diagram (Yanai et al, 2008).

The players :

- pal-1 : mainly comes from the mother, but some is generated in the cell too.
- elt-1 : expressed earlier than the genes below.
- elt-3
- nhr-25
- lin-26 : totally missed in earlier studies (e.g. Baugh et al).

Time series data

Baugh et al generated data for the approximately 5000 genes in C. elegans. 22 of these are relevant for "C" type cells. We will zero in on the 5 genes mentioned above.

*pal*1	*elt*1	*elt*3	*nhr*25	*lin*26	*pal*1	*elt*1	*elt*3	*nhr*25	*lin*26
356	16	7	12	213	342	14	8	12	226
275	17	6	11	228	357	12	8	14	166
165	18	7	12	110	458	13	8	15	182
134	21	7	13	71	339	16	8	17	90
106	22	7	13	59	185	16	8	18	62
116	140	8	14	68	235	59	8	13	56
115	184	8	14	67	447	76	9	18	70
156	270	13	22	135	638	185	8	60	122
160	200	18	79	195	290	82	11	61	205
104	104	15	51	370	323	110	34	93	635

discretize the data

Discretize both matrices at the same time : each column will have different scales. This is a crucial, not completely understood step (at least by me). Method used : Dimitrova et al (2010).
In this case we use 7 possible states : $[7] = \{0,1,2,3,4,5,6\}$.

*pal*1	*elt*1	*elt*3	*nhr*25	*lin*26
356	16	7	12	213
275	17	6	11	228
165	18	7	12	110
134	21	7	13	71
106	22	7	13	59
116	140	8	14	68
115	184	8	14	67
156	270	13	22	135
160	200	18	79	195
104	104	15	51	370

*pal*1	*elt*1	*elt*3	*nhr*25	*lin*26
4	1	0	0	2
3	1	0	0	3
1	1	0	0	1
1	2	0	0	0
0	2	0	0	0
0	4	1	0	0
0	5	1	0	0
1	6	3	2	1
1	6	4	4	2
0	3	3	3	4

Minsets

Fix the node i. Our plan is to determine the possible incoming edges for this node, given data :

$$\mathbb{D}(i) = \{(p, a) \in k^n \times k\}$$

Definition (Minset)

An **edge set** for the data $\mathbb{D}(i)$ is a set of variables $x_L := \{x_\ell \mid \ell \in L\}$, such that there is a function f only involving these variables, satisfying the data : $f(p) = a$, for all $(p, a) \in \mathbb{D}(i)$.

We call a minimal edge set (under inclusion) a **minset**.

the pessimistic view

Krupa, 2002

- The problem of finding one minset is "NP-hard".
- Gives a greedy algorithm to get one set that is close to a minset, but possibly larger.

This is bad news! Any problem that is that hard will be impossible for large inputs, it seems.

minsets : translate to algebra

Definition

If $(p,a),(q,b) \in \mathbb{D}(i)$, and $a \neq b$, define

$$m(p,q) = \prod_{p_j \neq q_j} x_j,$$

a monomial in $\mathbb{Q}[x_1,\ldots,x_n]$.

Example

If $p = (0,3,1,2)$ and $q = (1,2,1,2)$, then $m(p,q) = x_1 x_2$.

Key point : Suppose that $f(0,3,1,2) = 1$ and $f(1,2,1,2) = 3$. The f must involve either x_1 or x_2 (or both).

Definition

Given data $\mathbb{D}(i)$ as above, define the monomial ideal

$$J(\mathbb{D}(i)) := \langle m(p,q) \mid (p,a),(q,b) \in \mathbb{D}, \text{ and } a \neq b \rangle,$$

a monomial ideal in $\mathbb{Q}[x_1,\ldots,x_n]$.

Wiring diagrams and minimal primes

Definition

A **minimal prime** of a monomial ideal J is a (prime) ideal generated by a set of variables x_L, such that each proper subset $L' \subset L$ doesn't contain J.

Theorem (Jarrah-Laubenbacher-Stigler-S, 2007)

The set x_L is a minset for $\mathbb{D}(i)$ if and only if the ideal $\langle x_L \rangle$ is a minimal prime of $J(\mathbb{D}(i))$.

How can we find minimal primes of an ideal ? This is a solved problem in computational algebra !

Irreducible decompositions

Definition

An (irredundant) irreducible decomposition of a squarefree monomial ideal J is $x_{L_1},\ldots,x_{L_N}$ such that

$$J = \bigcap_{j=1}^{N} \langle x_{L_j} \rangle$$

and no smaller set of the $L's$ satisfies this property.

Theorem (From Commutative Algebra)

An irreducible decomposition is unique up to permutation, and the set of all minimal primes of J is $\langle x_{L_1} \rangle,\ldots,\langle x_{L_N} \rangle$

The upshot : if we can find the irreducible decomposition of $J(\mathbb{D}(i))$, then we have determined *all* of the minsets !

Computing an irreducible decomposition

Given a (square-free) monomial $\alpha = x_{\ell_1} x_{\ell_2} \dots x_{\ell_s}$, the *support* of α is $supp(\alpha) := \{x_{\ell_1}, \dots, x_{\ell_s}\}$.

Definition

The **Alexander dual** of a squarefree monomial ideal $J = \langle \alpha_1, \dots, \alpha_\ell \rangle$ is

$$J^* = dual(J) := \bigcap_{j=1}^{\ell} \langle supp(\alpha_j) \rangle.$$

Cool thing : $(J^*)^* = J$.

Theorem (from Commutative Algebra)

If $dual(J) = \langle \beta_1, \dots, \beta_N \rangle$ is a minimal set of generators for the Alexander dual of J, then the minimal primes of J are exactly the N primes :

$$\langle supp(\beta_1) \rangle, \dots, \langle supp(\beta_N) \rangle.$$

Algorithm for the possible wiring diagrams from $\mathbb{D}$

Minsets($\mathbb{D}$, x_i)

input : Data $\mathbb{D}(i)$ for variable x_i

output : a list of the minsets for $\mathbb{D}(i)$

$J(\mathbb{D}(i)) := \langle m(p,q) \mid (p,a),(q,b) \in \mathbb{D}(i), a \neq b \rangle$

$J^* := \text{dual}(J(\mathbb{D}(i))) = \langle \beta_1, \ldots, \beta_N \rangle$

return $\{supp(\beta_j) \mid 1 \leq j \leq N\}$.

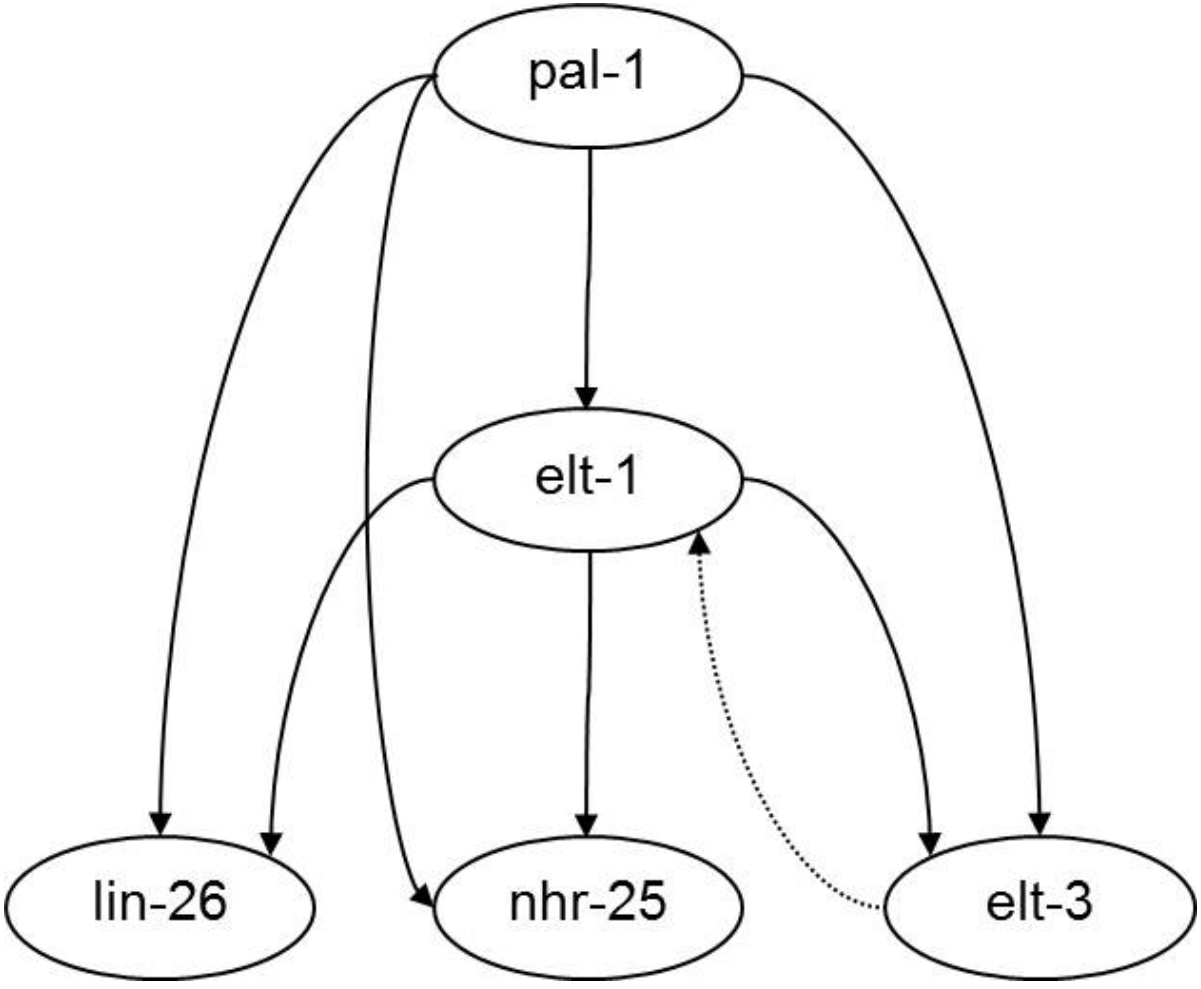
OK, the real complexity is in computing $dual(J)$. How to do that ?

Bjarke Roune, now at google, www.broune.com

- The slice algorithm for irreducible decompositions of monomial ideals (2007).
- Frobby : software for monomial ideals (library included in *Macaulay2*). Current fastest, can handle large size problems.

C. elegans example

The 22 node network : each node has between 352 and 1209 minsets, sizes ranging from 2 to 7. Can use biological information to reduce the number of meaningful minsets.
The 5 node subnetwork deduced from this :



Some important issues and questions

- Arbitrary functions don't appear in biological networks. How to modify the minsets algorithm to only consider functions of a given type ?
- Edges in the wiring network for a biological system almost always have a *sign* associated with them : these edges are either positive (activators) or negative (inhibitors). How to incorporate or find this kind of information ?
- Given many minsets, how do we determine which ones to consider ? (i.e. how to score edges or minsets).
- Data is noisy, and this method is very sensitive to even small changes in the discretized data. How to handle this ?

Improvement : Signed minsets, Veliz-Cuba (2012)

Goal : want a directed graph as wiring diagram, but want to **sign** the edges : positive or negative.
Most functions don't occur in biology. We want to restrict to a reasonable class of functions. Here is the most basic :

Definition

A function $f(x_1,\ldots,x_n)$ is called **positive** for x_i (resp. **negative** for x_i) if for every possible $c_1,\ldots,c_{i-1},c_{i+1},\ldots,c_n$, and $a<b$, then

$$f(c_1,\ldots,c_{i-1},a,c_{i+1},\ldots,c_n) \leq f(c_1,\ldots,c_{i-1},b,c_{i+1},\ldots,c_n)$$

(respectively : $\geq$).

Definition

A function is $f(x_1,\ldots,x_n)$ is called **unate** if for each i, either f doesn't depend on x_i, is positive for x_i, or is negative for x_i.

If P is the set of variables where f is positive, N is the set where it is negative, then f is called **unate with signs** (P,N).

Example

If $k = \mathbb{Z}/3$, and $f(x,y)$ is a function $k^2 \longrightarrow k$, then we can write f's values as an array :

$$\begin{array}{ccc} 2 & 1 & 0 \\ 2 & 2 & 0 \\ 2 & 2 & 1 \end{array}$$

(For example, $f(0,2) = 0$). f is unate with signs $\{1^+, 2^-\}$. In general, if the values in the rows always increase or always decrease, same with columns, the function is unate.

Definition (signed minset)

A **signed minimal edgeset (signed minset)** for data $\mathbb{D}(i)$ is a pair (P,N) such that there exists a unate function with signs (P,N).

Current goal : find all of the signed minsets compatible for data $\mathbb{D}(i)$.

Definition

If $(p,a),(q,b) \in \mathbb{D}(i)$, and $a < b$, define

$$m_{signed}(p,q) = \prod_{p_j<q_j} x_j \prod_{p_j>q_j} y_j,$$

a monomial in $\mathbb{Q}[x_1,\ldots,x_n,y_1,\ldots,y_n]$.

Example

If $p=(0,3,1,2)$ and $q=(1,2,1,2)$, and $a=1<b=3$, then $m_{signed}(p,q) = x_1 y_2$.

Key point : Suppose that $f(0,3,1,2)=1$ and $f(1,2,1,2)=3$. If f is unate, then f must be either positive on x_1, or negative on x_2 (or both).

Definition

Given data $\mathbb{D}(i)$ as above, define the monomial ideal

$$J_{signed}(\mathbb{D}(i)) := \langle m_{signed}(p,q) \mid (p,a),(q,b) \in \mathbb{D}, \text{ and } a < b \rangle,$$

a monomial ideal in $\mathbb{Q}[x_1,\ldots,x_n,y_1,\ldots,y_n]$.

Theorem (Alan Veliz-Cuba (2012))

The minimal signed edge sets of $\mathbb{D}(i)$ are in 1-1 correspondence with the minimal primes of the monomial ideal

$$J_{signed}(\mathbb{D}(i)) = \langle m_{signed}(p,q) \mid (p,a),(q,b) \in \mathbb{D}(i), a < b \rangle.$$

which do not contain both x_j and y_j (for any j).

Signed minsets : example

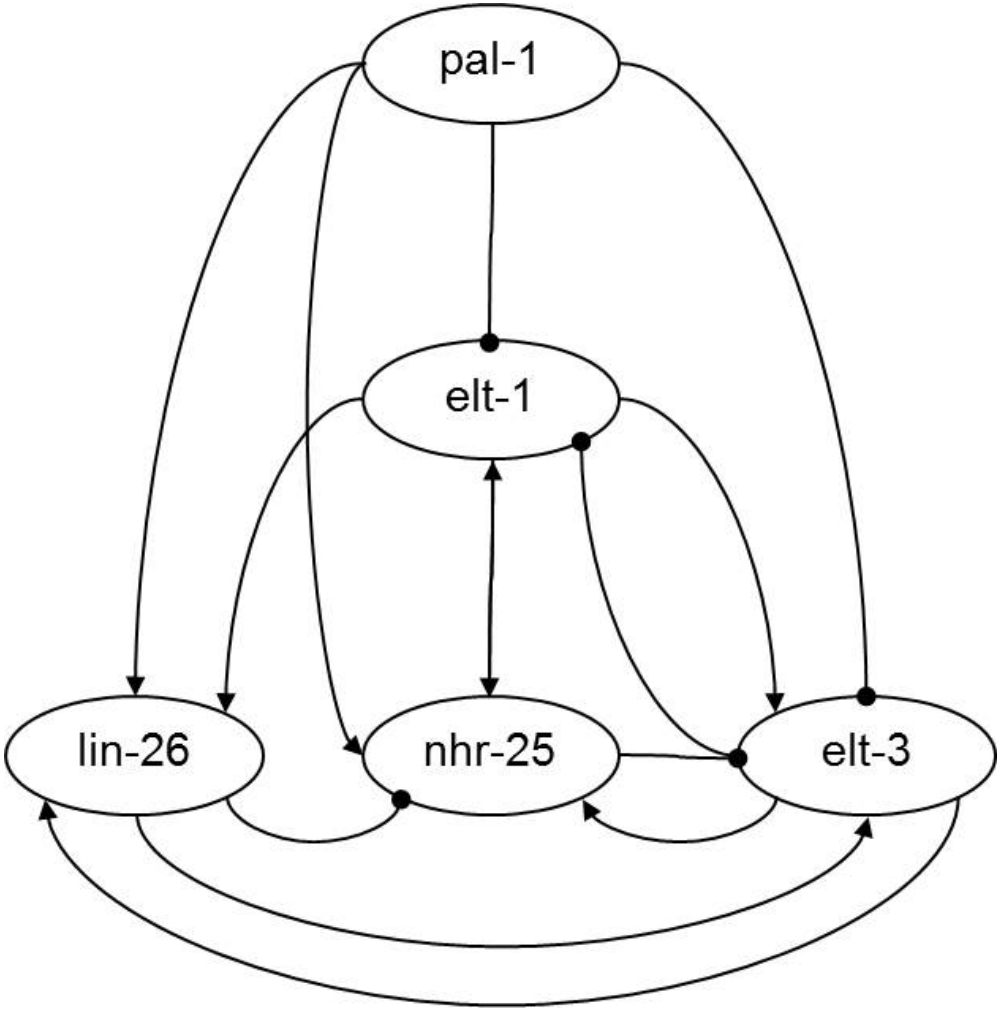
Upshot : For the price of using twice as many variables, we obtain more information, **assuming** that the functions at a node are unate, often exactly what we want.
Caveat : Many more signed minsets, and the number of variables is often larger.

Example (22 node C. elegans network)

node	signed	mindeg	unsigned	mindeg
1	60548	4	1055	3
2	58873	5	980	3
3	47725	3	676	2
4	35887	3	934	2
5	29894	3	771	3
6	34187	4	956	3
7	16442	3	622	2
8	41306	4	1033	3
9	29416	3	1042	3
10	33140	3	648	2
11	18143	3	612	2
12	35715	4	919	2
...				

The 5 node subnetwork prediction by signed minsets

Signed minsets : There is a unique signed minset for each node, except for lin-26 where there are 2 signed minsets.



Reverse engineering of the dynamics

Problem : Given data $\mathbb{D}$, and a wiring diagram for the network (a directed graph G, possibly with signs on each edge), find functions f_i which only involve variables whose edges come into node x_i, which interpolate $\mathbb{D}$
First solution, which got everything rolling, and which got me interested in the whole project :

(Laubenbacher, Stigler, 2004)

Let $I \subset k[x_1, \ldots, x_n]$ be the ideal of polynomials vanishing at the input points of $\mathbb{D}$.
Then if f_i is one possible function for the ith node, then **all** possible functions at node i, that match the data exactly are in the set

$$f_i + I := \{f_i + g \mid g \in I\}.$$

Take a Gröbner basis of the ideal I, and apply the division algorithm to f_i to obtain a normal form $rem(f_i, I) \in f_i + I$. Use this as the dynamics for node i.

Gröbner fan approach : Dimitrova et al (2007, 2010, 2011)

Each Gröbner basis gives a possible dynamics for a node.

Idea : sample from the set of all possible Gröbner bases, choose the normal forms which occur most often, and use these as tentative dynamical functions.

What is known about the collection of Gröbner bases :

- There are only finitely many Gröbner bases of I.
- All term orders arise from a weight vector in $\mathbb{R}^n$.
- Partition $\mathbb{R}^n$ according to what Gröbner basis you get.
- The resulting partition is a polyhedral fan (collection of cones).
- There is software (gfan, Anders Jensen) for computing this fan (and all the Gröbner bases), included with Macaulay2.

Open problem : do the relative sizes of Gröbner cones give a biologically meaningful probability distribution on the set of possible dynamical systems ?

REACT : (Vera-Licona et al (2014))

- Idea : Start with a boolean network and time series data $\mathbb{D}$. Search through the set of all networks to find networks which minimize (or nearly minimize) a cost function.
- Search method : Genetic algorithm, keeping sets of polynomial dynamical systems, and combining them at each generation.
- Could easily work for $p > 2$, but then the search is not so good. A theorem of Babson et al in computational algebra and combinatorics limits in $p = 2$ case the kinds of functions one needs to consider.
- Key point : This method is robust to small changes/errors in the data. It is one of the only methods so far that has this property.

Software : REACT.

Open problems

Minsets in presence of errors

Modify the minsets algorithm or the signed minsets algorithm to be robust in the presence of a small number of errors, or noise, in the data.

Behavior of minsets

Describe the behavior of the minsets (or signed minsets) algorithm under changes in the discretization method.

Add statistical methodology

Provide a statistical method, such as maximum likelihood, for the existence of a directed edge in the wiring diagram, given the complete set of minsets, or the answer to the previous open problem. Prove that the methods are better than random. Currently, the selection of edges/minsets is not biologically motivated. Make it so.

More open problems

Discretization

Find better (i.e. more adaptive) methods for discretization. Use user provided information when appropriate and use biological information.

Incorporate recent exciting biological information

Incorporate biological data into these algorithms. For example : use location in a cell, speed of different operations. Use 3-dimensional structure for regulatory motifs in DNA.

Relate algebra to biology

How to encode biological properties into term order selection in the Gröbner basis mehtod to find dynamics of a network ?

Software for discrete modelling and reverse engineering

Many different algorithms and methods for reverse engineering presented or fleetingly mentioned. Many have standalone implementations suitable mostly for researchers and experts in the software.

Reverse engineering modules

- discretize (Dimitrova, C++)
- minset (Stigler-S, Macaulay2)
- signed minsets (Veliz-Cuba, Macaulay2)
- scoring (signed or unsigned) minsets : Macaulay2
- Gröbner fan method : (Dimitrova, Macaulay2)
- REACT, a genetic algorithm to find better dynamics (Vera-Licona, C++)
- finding nested canalyzing functions (Hinkelmann)

Macaulay2 is open source software written by Dan Grayson and myself (1990s-present), for computing in algebraic geometry and related fields.

Similarly, many algorithms and methods available to analyze a dynamic model, once it has been constructed.

Analysis of a discrete model

- Fixed points of the system, limit cycles. Measures the "robustness" of a network. Methods include :
- small systems : enumeration of the state space. (Hoops, C++)
- stochastic simulation of the system. (Arat, Perl)
- model reduction : take a boolean model (network and functions). Veliz Cuba et al (2011, 2013, 2014) show how this can be done, and show that it gives a **dramatic** increase in the size of problems which can be handled.
- Gröbner basis methods (Hinkelmann, Macaulay2)

Turing : A discrete modelling toolbox suitable for non-mathematical biologists

Idea : combine this software, have interfaces for mathematical biologists, as well as non-mathematical biologists.

Software project : need a name yet !

Collaborators : Aguilar, Arat, DimitrovaFavre, Ha, Hoops, Laubenbacher, Murrugarra, Stigler, Stillman, Veliz-Cuba, Vera-Licona.

Components :

- Reverse engineering modules
- Modules for analysis of discrete networks and dynamics
- Visualization
- Interface for non-mathematicians (web app)
- Interface for modellers (similar, but with more options available)

More docker

Use docker

- Use docker (docker.com) to make containers of all of these modules.
- These images will be freely available on docker hub.
- Use json as a data format, and use data formats that promote ease of connecting modules together.
- Use docker compose to create a sequence of docker containers, one for each module, to create standard workflows usable (with the help of a friendly front-end) by non-experts.

Advantages to using docker

- Scientific reproducibility. This has become a serious problem in science. Docker helps since anyone (on a mac, PC, or linux) can run docker containers, and they will be freely available.
- Modules can be designed and implemented totally independently (given restriction on data input and output).
- With the proper glue functionality (designed and implemented in ruby by Thibauld Favre), the user will never even need to know about these containers.

More docker

Use docker

- Use docker (docker.com) to make containers of all of these modules.
- These images will be freely available on docker hub.
- Use json as a data format, and use data formats that promote ease of connecting modules together.
- Use docker compose to create a sequence of docker containers, one for each module, to create standard workflows usable (with the help of a friendly front-end) by non-experts.

Advantages to using docker

- Scientific reproducibility. This has become a serious problem in science. Docker helps since anyone (on a mac, PC, or linux) can run docker containers, and they will be freely available.
- Modules can be designed and implemented totally independently (given restriction on data input and output).
- With the proper glue functionality (designed and implemented in ruby by Thibauld Favre), the user will never even need to know about these containers.

Thanks !