Hard questions about simple finite dynamical systems

Vinny Just

Mathematical Biosciences Institute and Department Mathematics, Ohio University

May 25, 2006

Transcriptional gene regulation

The best understood mechanism of gene regulation is transcriptional *cis*-regulation. As an example, consider a situation where gene number 7 is being transcribed iff transcription factor number 12 or transcription factor number 28 bind to the promotor, and repressor number 13 does not bind. If gene number 7 is transcribed at time step t and its product does itself play a role in gene regulation, *e.g.*, is transcription factor number 7, then this product will play its regulatory role at time step $t + 1$.

Boolean dynamical systems

Boolean dynamical systems (BDS's) have been proposed and studied as models of gene regulation. The state space is $\{0,1\}^n$; for example, a state vector $[0,1,1]$ means that the product of gene 1 is absent or at low concentration while the products of genes 2 and 3 are present at high concentration. The updating function $f : \{0,1\}^n \to \{0,1\}^n$ computes the state of the system at the next time step. In the example from the previous slide, we would have

$$f_7(\bar{x}) = (x_{12} \vee x_{28}) \wedge \neg x_{13}.$$

Questions about BDS's:

- How to build a BDS model from data? ("Reverse Engineering")

- Given a BDS by a formula for its updating function, deduce the dynamic properties.

- If we treat BDS models as approximations of continuous differential equation models of gene regulatory networks, how well does the dynamics of the BDS reflect the dynamics of the underlying system of differential equation?

Some questions about dynamics of BDS's:

- Does there exist a steady state?
- Is every limit cycle a steady state?
- Conversely, does there exist a limit cycle of length > 1?
- Does there exist a limit cycle of length 2?

A simple-minded answer:

For answering any of these questions, just compute $f(\bar{x})$, $f(f(\bar{x}))$, ... for all $\bar{x} \in \{0,1\}^n$ and keep track if the length of the limit cycles you found.

This is guaranteed to work **but** we may need to look at 2^n initial states $\bar{x}$ which becomes infeasible for large n.

(When) can we find a faster algorithm?

The class P:

Questions like "Does this BDS have a limit cycle?" are known as *decision problems.* An *instance* of this problem is a given BDS. The *description length* of an instance is the number of bits needed to specify the instance, in this case, the updating function f. If each component of f is relatively simple, *e.g.,* linear, a monomial, a threshold function, then the description length will be on the order of $O(n^2)$, where n is the number of variables.

A decision problem is in the class **P** if there exists an algorithm for correctly deciding this problem in all instances whose running time on instances of size n is bounded by a polynomial in n.

We will refer to the problem of existence of limit cycles of length > 1 as the *LC-problem* and to the problem of existence of limit cycles of length 2 as the *LC2-problem.*

Theorem 1. *(Elspas 1959, Hernández-Toledo 2005, Jarrah, Laubenbacher, Vera-Licona 200?) The LC-problem for BDS's with linear updating functions is in the class* **P**.

Theorem 2. *(Colón-Reyes, Laubenbacher, Pareigis 2004) The LC-problem for BDS's where each component of the updating function is a monomial,* i.e., *is of the form* $x_{i_1}x_{i_2}\ldots x_{i_k}$*, is in the class* **P**.

Question 3. *How about more general BDS's? What if we allow also functions of the form* $x_{i_1}x_{i_2}\ldots x_{i_k}+1$ *or threshold functions? What if we allow only functions of these forms that depend on at most two variables?*

Witnesses

Suppose $\bar{x}$ is a state that is part of a limit cycle of length 2. Such $\bar{x}$ will be called a *witness* for (a "yes" answer to) the LC2-problem. Verifying that a given $\bar{x}$ is a witness for the LC2-problem can be done by an algorithm that is polynomial in the description length of the given BDS.

In contrast, let $\bar{x}$ be a state that is part of a limit cycle of length > 1. Then $\bar{x}$ is a witness for the LC-problem, but since limit cycles may have length that is exponential in n, it may not be possible to verify in polynomial time that $\bar{x}$ is a witness for the LC-problem.

The class NP

A decision problem is in the class **NP** if:

- There is a notion of "potential witness" and "witness" for an instance of the problem.

- There exists a witness for an instance I iff the correct answer for instance I is "yes."

- The problem of deciding whether a potential witness is an actual witness is in the class **P**.

Thus the LC2-problem is in the class **NP**, but it is not clear whether the LC-problem is in the class **NP**.

The P = NP problem

Question 4. *Is* **P = NP***?*

This is one of the major open problems in mathematics.

NP-hard and NP-complete problems

A decision problem is *NP-hard* if the existence of a polynomial-time algorithm for solving this problem would imply **P = NP**. An NP-hard problem that is itself in the class **NP** is *NP-complete.*

More than a thousand decision problems are known to be NP-complete.

Two examples

Theorem 5. *(Akutsu, Kuhara, Maruyama, Miyano 1998) The problem of deciding whether a BDS has a steady state is NP-complete.*

Suppose $x_1, \ldots, x_n$ are Boolean variables and

$$\psi = (y_{11} \vee y_{12} \vee y_{13}) \wedge \ldots \wedge (y_{m1} \vee y_{m2} \vee y_{m3}),$$

where y_{kj} is either $x_{i_{kj}}$ or $\neg x_{i_{kj}}$ for some $i_{kj} \in [n]$. Then ψ is *satisfiable* if there exists a Boolean vector $\bar{x}$ of length n such that $\psi(\bar{x})$ evaluates to 1. The 3-SAT problem takes as instances formulas ψ as above and asks whether ψ is satisfiable.

By a classical theorem (Karp 1972), 3-SAT is an NP-complete problem.

Polynomial-time reducibility

Let E and D be two decision problems. We say that D is *polynomial-time reducible to* E if there exists an algorithm A such that:

- A takes instances of D as inputs and always outputs the correct answer for each instance of D.

- A uses as a subroutine a hypothetical algorithm B for solving E.

- There exists a polynomial p such that for every instance of D of size n the algorithm A terminates in at most $p(n)$ steps if each call of the subroutine B is counted as only one step.

How to prove NP-completeness?

The formal definition of NP-hardness of a decision problem E requires that every decision problem D in the class **NP** be polynomial-time reducible to E. Thus if we want to show that a given decision problem F is NP-hard, it suffices to show that some decision problem E that is known to be NP-hard is polynomial-time reducible to F. In other words, we need to show that if there were a polynomial-time algorithm for deciding F, there would also be such an algorithm for E.

3-SAT is polynomial-time reducible to LC2

Suppose $\{x_1, \ldots x_n\}$ is a set of Boolean variables, and suppose

$$\psi = (y_{11} \vee y_{12} \vee y_{13}) \wedge \ldots \wedge (y_{m1} \vee y_{m2} \vee y_{m3}),$$

where y_{kj} is either $x_{i_{kj}}$ or $\neg x_{i_{kj}}$ for some $i_{kj} \in [n]$. We construct a Boolean dynamical system $< \{0,1\}^{n+2}, f >$ as follows:

- $f_i = x_i$ for $i \in [n]$.
- $f_{n+1} = \psi(x_1, \ldots, x_n) \wedge x_{n+2}$.
- $f_{n+2} = x_{n+1}$.

The dynamics of our system

Let us see what happens to the updates of our system.

Case 1: $\psi(x_1, \ldots, x_n) = 0$.

$$[x_1, \ldots, x_n, ?, ?] \mapsto [x_1, \ldots, x_n, 0, ?] \mapsto [x_1, \ldots, x_n, 0, 0].$$

Thus the system reaches a steady state from any initial state in at most two steps.

Case 2: $\psi(x_1, \ldots, x_n) = 1$.

$$[x_1, \ldots, x_n, 1, 0] \mapsto [x_1, \ldots, x_n, 0, 1] \mapsto [x_1, \ldots, x_n, 1, 0].$$

This is a limit cycle of length 2. Such limit cycles exist iff ψ is satisfiable.

3-SAT is polynomial-time reducible to LC2

Here is an algorithm for 3-SAT that uses a hypothetical algorithm for LC2:

- Build the BDS we just described. This can be done by a polynomial-time algorithm in the description length of ψ.

- Run our hypothetical algorithm for deciding whether this BDS has a limit cycle of length 2.

- If yes, conclude that ψ is satisfiable. If not, conclude that ψ is not satisfiable.

Generalized monomial systems

Theorem 6. *The LC2-problem is NP-complete and the LC-problem is NP-hard for BDS's in which every component of the updating function is of the form* $x_i x_j$ *or of the form* $x_i + 1$ *for some (possibly equal)* $i, j \in [n]$.

The proof uses essentially the same idea that we have just presented, but one needs to work a little harder.

The system used in Theorem 6

- $f_i = x_i$.
- $f_{n+i} = x_i + 1$.
- If $y_{k,1} = \neg x_{i_{k,1}}$ and $y_{k,2} = \neg x_{i_{k,2}}$, then $f_{2n+k} = x_{i_{k,1}} x_{i_{k,2}}$.
- If $y_{k,1} = x_{i_{k,1}}$ and $y_{k,2} = \neg x_{i_{k,2}}$, then $f_{2n+k} = x_{n+i_{k,1}} x_{i_{k,2}}$.
- If $y_{k,1} = \neg x_{i_{k,1}}$ and $y_{k,2} = x_{i_{k,2}}$, then $f_{2n+k} = x_{i_{k,1}} x_{n+i_{k,2}}$.
- If $y_{k,1} = x_{i_{k,1}}$ and $y_{k,2} = x_{i_{k,2}}$, then $f_{2n+k} = x_{n+i_{k,1}} x_{n+i_{k,2}}$.

- If $y_{k,3} = \neg x_{i_{k,3}}$, then $f_{2n+m+k} = x_{2n+k}x_{i_{k,3}}$.
- If $y_{k,3} = x_{i_{k,3}}$, then $f_{2n+m+k} = x_{2n+k}x_{n+i_{k,3}}$.
- $f_{2n+2m+k} = x_{2n+m+k} + 1$.
- $f_{2n+3m+1} = x_{2n+2m+1}$.
- $f_{2n+3m+k+1} = x_{2n+3m+k}x_{2n+2m+k+1}$ for $k \in [m-1]$.
- $f_{2n+4m+1} = x_{2n+4m+2} + 1$.
- $f_{2n+4m+2} = x_{2n+4m}x_{2n+4m+1}$.

Monotone systems

A Boolean function f is *monotone* if $\bar{x} \leq \bar{y}$ implies $f(\bar{x}) \leq f(\bar{y})$. We say that a BDS is a *monotone system* if every component f_k of the updating function can be written as a combination of functions $x_i \vee x_j$ and $x_i \wedge x_j$. Note that the use of negations is not allowed in monotone systems. Moreover, note that the vectors $[0, \ldots, 0]$ and $[1, \ldots, 1]$ are steady states for every monotone system. Thus the result of Akutsu *et al.* does not apply to such systems.

Theorem 7. *The problem of deciding whether a given monotone BDS has at least three steady states is NP-complete.*

Proof: Suppose $\{x_1, \ldots x_n\}$ is a set of Boolean variables, and suppose

$$\psi = (y_{11} \vee y_{12} \vee y_{13}) \wedge \ldots \wedge (y_{m1} \vee y_{m2} \vee y_{m3}),$$

where y_{kj} is either $x_{i_{kj}}$ or $\neg x_{i_{kj}}$ for some $i_{kj} \in [n]$. We construct a Boolean dynamical system $< \{0,1\}^{2n+3}, f >$ as follows:

- $f_i = (x_i \wedge x_{2n+1} \wedge x_{2n+3}) \vee x_{2n+2}$ for $i \in [2n]$.
- $f_{2n+1} = (x_1 \vee x_{n+1}) \wedge \cdots \wedge (x_n \vee x_{n+n})$.
- $f_{2n+2} = (x_1 \wedge x_{n+1}) \vee \cdots \vee (x_n \wedge x_{n+n})$.
- $f_{2n+3} = \psi^*(x_1, \ldots, x_n, x_{n+1}, \ldots, x_{2n})$, where ψ^* is obtained from ψ by replacing every occurrence of $\neg x_i$ by x_{n+i}.

The dynamics of our system

Let $\bar{x}$ be an initial state of our system.

- If $x_i = x_{n+i} = 1$ for some $i \in [n]$, then $f(f(\bar{x}))$ takes values $x_j = 1$ for all $j \in [2n]$ and $f^3(\bar{x}) = [1, \ldots, 1]$.

- If $x_i = x_{n+i} = 0$ for some $i \in [n]$ and the system does not reach the steady state $[1, \ldots 1]$ from $\bar{x}$, then $f(f(\bar{x}))$ takes values $x_j = 0$ for all $j \in [2n]$ and $f^3(\bar{x}) = [0, \ldots, 0]$.

- If the system does not reach the steady state $[1, \ldots, 1]$ from $\bar{x}$ and ψ is not satisfiable, then $f^3(\bar{x}) = [0, \ldots, 0]$.

- If $\psi(x_1, \ldots, x_n) = 1$, then $\bar{x} = [x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n, 1, 0, 1]$ is a steady state.

LC and LC2 for monotone systems

Theorem 8. *The LC2-problem is NP-complete and the LC-problem is NP-hard for BDS's in which every component of the updating function is of the form* $x_i \vee x_j$ *or of the form* $x_i \wedge x_j$ *for some (possibly equal)* $i, j \in [n]$.

The proof combines ideas of the proofs of our previous theorems.

The system used in Theorem 8

- $f_{x_{i,1}} = x_{i,L} \vee t_{n+1}$ for $i \in [n]$.
- $f_{c_{i,1}} = c_{i,L} \vee t_{n+1}$ for $i \in [n]$.
- $f_{x_{i,\ell+1}} = x_{i,\ell}$ for $i \in [n]$ and $\ell \in [L-3]$.
- $f_{c_{i,\ell+1}} = c_{i,\ell}$ for $i \in [n]$ and $\ell \in [L-3]$.
- $f_{x_{i,L-1}} = x_{i,L-2} \wedge u_{L-4}$ for $i \in [n]$.
- $f_{c_{i,L-1}} = c_{i,L-2} \wedge u_{L-4}$ for $i \in [n]$.
- $f_{x_{i,L}} = x_{i,L-1} \wedge e_{L-4}$ for $i \in [n]$.
- $f_{c_{i,L}} = c_{i,L-1} \wedge e_{L-4}$ for $i \in [n]$.

- $f_{v_{i,1}} = x_{i,1} \vee c_{i,1}$ for $i \in [n]$.
- $f_{v_{i,j+1}} = v_{i,\ell}$ for $i \in [n]$ and $j \in [i-1]$.
- $f_{u_1} = v_{1,1}$.
- $f_{u_{i+1}} = u_i \wedge v_{i+1,i+1}$ for $i \in [n-1]$.
- $f_{u_{n+r}} = u_{n+r-1}$ for $r \in [L-4-n]$.
- $f_{w_{i,1}} = x_{i,L-2} \wedge c_{i,L-2}$ for $i \in [n]$.
- $f_{w_{i,j+1}} = w_{i,j}$ for $i \in [n]$ and $j \in [i-1]$.
- $f_{t_1} = w_{1,1}$.
- $f_{t_{i+1}} = t_i \vee w_{i+1,i+1}$ for $i \in [n-1]$.

- $f_{t_{n+1}} = t_n \vee t_{n+1}$.

- $f_{d_{k,1}} = x_{i_{k,1},1} \vee x_{i_{k,2},1}$ if $k \in [m]$ and $y_{k,1} = x_{i_{k,1}}$ and $y_{k,2} = x_{i_{k,2}}$.

- $f_{d_{k,1}} = x_{i_{k,1},1} \vee c_{i_{k,2},1}$ if $k \in [m]$ and $y_{k,1} = x_{i_{k,1}}$ and $y_{k,2} = \neg x_{i_{k,2}}$.

- $f_{d_{k,1}} = c_{i_{k,1},1} \vee x_{i_{k,2},1}$ if $k \in [m]$ and $y_{k,1} = \neg x_{i_{k,1}}$ and $y_{k,2} = x_{i_{k,2}}$.

- $f_{d_{k,1}} = c_{i_{k,1},1} \vee c_{i_{k,2},1}$ if $k \in [m]$ and $y_{k,1} = \neg x_{i_{k,1}}$ and $y_{k,2} = \neg x_{i_{k,2}}$.

- $f_{d_{k,2}} = d_{k,1} \vee x_{i_{k,3},2}$ if $k \in [m]$ and $y_{k,3} = x_{i_{k,3}}$.

- $f_{d_{k,2}} = d_{k,1} \vee c_{i_{k,3},2}$ if $k \in [m]$ and $y_{k,3} = \neg x_{i_{k,3}}$.

- $f_{d_{k,\ell+2}} = d_{k,\ell+1}$ for $k \in [m]$ and $\ell \in [k-1]$.
- $f_{e_1} = d_{1,2}$.
- $f_{e_{k+1}} = e_k \wedge d_{k,k+1}$ for $k \in [m]$.
- $f_{e_{m+r+1}} = e_{m+r}$ for $r \in [L-5-m]$.