Complexity of Inferring Local Transition Functions of Discrete Dynamical Systems

20th International Conference on
Implementation and Application of Automata (CIAA)
18-21 August 2015, Umea, Sweden

A. Adiga, C. J. Kuhlman, M. V. Marathe

Network Dynamics and Simulation Science Laboratory (NDSSL)
Virginia Bioinformatics Institute and Virginia Tech, Blacksburg, VA
Email: {abhijin, ckuhlman, mmarathe}@vbi.vt.edu

S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns

Department of Computer Science
University at Albany – SUNY, Albany, NY
Email: {sravi@albany.edu, drosenkrantz@gmail.com, thestearns2@gmail.com}

Talk Outline

1. Basics of Discrete Dynamical Systems
2. Previous Work on Discrete Dynamical Systems
3. Threshold Inference Problems and Results
4. Summary and Future Work

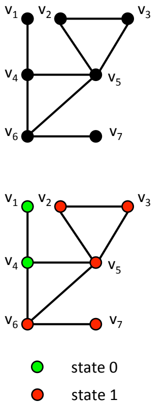
Part 1
Basics of Discrete Dynamical Systems

Discrete Dynamical Systems: Basics

A **Discrete Dynamical System** $\mathcal{S}$ consists of

- An underlying (undirected or directed) **graph** $G(V, E)$.
  1. **Nodes:** Agents in the system.
  2. **Edges:** Permissible local interactions.
- State values for nodes from a finite domain $\mathbb{B}$ (e.g. $\mathbb{B} = \{0, 1\}$).
- A **local transition function** for each node (e.g. a threshold function).
- **Update mechanism:** **synchronous**, sequential, block sequential, etc.

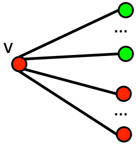
Notation: SyDS (Synchronous Dynamical System)



Thresholds and Node State Transitions

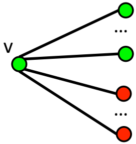
Informally, transitions for v based on threshold t_v:

Vertex v transitions 1 (red) to 0 (green); "down transition"



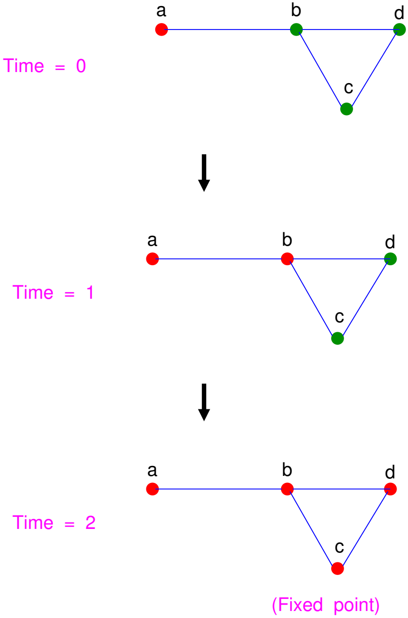
If (number of reds) < t_v,
then v becomes **green**.
Else, v stays **red**.

Vertex v transitions 0 (green) to 1 (red); "up transition"



If (number of reds) ≥ t_v,
then v becomes **red**.
Else, v stays **green**.

Time Evolution of a SyDS



Notation:

● -- State 0

● -- State 1

Threshold
$t_v = 1$.

Some Definitions

- **Configuration** at time t: Vector specifying the state of each node at time t.
- **Successor** of a configuration $\mathcal{C}$: The configuration that **immediately follows** $\mathcal{C}$ in time evolution.
- **Predecessor** of a configuration $\mathcal{C}$: A configuration that **immediately precedes** $\mathcal{C}$ in time evolution.

Note: In any SyDS (a deterministic system),

- each configuration has a **unique successor**;
- a configuration may have **zero or more predecessors**.

Some Definitions (continued)

- **Stable Configuration or Fixed Point**: A configuration $\mathcal{C}$ whose successor is $\mathcal{C}$ itself.
- **Unstable Configuration**: A configuration $\mathcal{C}$ whose successor is different from $\mathcal{C}$.
- **Garden of Eden Configuration**: A configuration $\mathcal{C}$ which does not have a predecessor.

Note: We will use "Stable Configuration" instead of "Fixed Point".

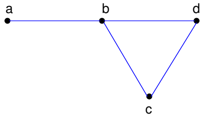
Phase Space of a Discrete Dynamical System

The **phase space** of a discrete dynamical system $\mathcal{S}$ is a **directed graph** $\mathcal{P}$.

- Each node of $\mathcal{P}$ represents a configuration of $\mathcal{S}$.
- Each directed edge (x, y) indicates that y is the successor of x.

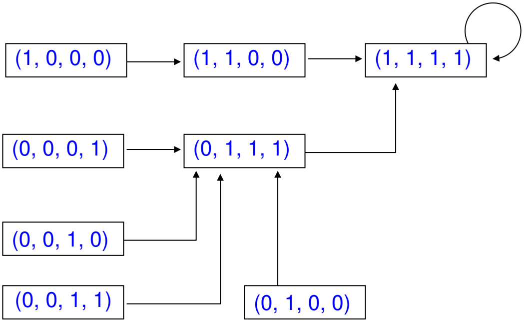
Note: The size of $\mathcal{P}$ is **exponential** in the size of $\mathcal{S}$.

Example – Phase Space of Dynamical System $\mathcal{S}$



Dynamical System $\mathcal{S}$

(1-threshold function at each node)



Note: Only a portion of the phase space is shown.

(1,0,0,0) is a configuration; there are 8 shown above.
(1,0,0,0) is a Garden of Eden configuration.
(1,1,1,1) is a stable configuration.
(0,1,1,1) is an unstable configuration.
(0,1,0,0) is a predecessor of (0,1,1,1).
(0,1,1,1) is a successor of (0,1,0,0).

Motivation: Diffusion Phenomena in Networks

- **Contagion** processes model many social phenomena (e.g. propagation of information, influence, diseases, trends, etc.).
- Usual modeling assumptions:
  - Agents in the system have states that vary with time.
  - The next state of an agent depends on its current state and those of its neighbors (i.e., agent interactions are **local**).
- **Threshold-based** mechanisms commonly used to capture behavior; the behavior of an agent depends on how many of its neighbors are in certain states (e.g. [Granovetter 1978, Easley & Kleinberg 2010]).
- **Discrete Dynamical Systems:** A formal model for analyzing contagion phenomena.

Part 2
Previous Work on Discrete Dynamical Systems

Analysis Problems for Dynamical Systems

Analysis Problem (Informal Definition):

- **Given:** A **fully specified** system $\mathcal{S}$ and a behavioral property P.
- **Goal:** Determine whether $\mathcal{S}$ has the specified property P.

Examples of Properties:

- $\mathcal{S}$ have a stable configuration.
- $\mathcal{S}$ has a Garden of Eden configuration.

Note: The above questions concern **subgraphs** of the phase space $\mathcal{P}(\mathcal{S})$ of $\mathcal{S}$.

Previous Work on Analysis Problems

- Computational intractability results for fixed point existence and counting problems (e.g. [Barrett et al. 2001], [Kosub & Homan 2007], [Tosic 2010]).
- Computational intractability results for Garden of Eden existence [Barrett et al. 2001].
- Computational intractability results for finding a predecessor of a given configuration [Barrett et al. 2007].
- Polynomial time algorithms for some of the above problems for restricted classes of SyDSs (e.g. treewidth-bounded graphs and restricted local functions) [Barrett et al. 2001, 2007].

Part 3
Threshold Inference Problems and Results

Inference Problems for Dynamical Systems

Inference Problem (Informal Definition):

- **Given:** A **partially specified** system $\mathcal{S}'$ and some observed behavior (e.g. a set of stable configurations).
- **Goal:** Infer the other parts of $\mathcal{S}'$ to obtain a fully specified system $\mathcal{S}$ which exhibits the observed behavior.

Some Applications:

- Important step in model calibration (see for example, [Trucano et al. 2006]).
- Estimating model parameters from observed data on the spread of epidemics and information (see for example, [Gonzalez-Bailon et al. 2011]).

Inference Problems: Related Work

- Inferring sources of infection given the network and diffusion traces (e.g. [Shah et al. 2011]).
- Inferring network structure given traces of a diffusion process (e.g. [Abraho et al. 2013], [Gomez-Rodriguez et al. 2010], [Soundarajan & Hopcroft, 2010]).
- Inferring a Boolean function given the class to which it belongs and its values on some inputs (see for example, [Kearns & Vazirani, 1994]).

Threshold Inference for Synchronous Systems

Focus:

- Synchronous systems with threshold functions at each node.
- Use observed behavior to infer a threshold value for each node.

Inference Problem: An Example

- **Given:** The underlying graph $G(V, E)$ of a synchronous dynamical system $\mathcal{S}$ and a set Q of configurations of $\mathcal{S}$.
- **Requirement:** Find a threshold value for each node, if one exists, so that for the resulting (fully specified) dynamical system, each configuration in Q is a **stable** configuration.

Types of Behavior Specifications

I. Homogeneous Specifications: The input consists of only one type of behavior.

Examples:

- A collection of stable configurations.
- A collection of unstable configurations.
- A collection of Dendrograms.
- A collection of Garden of Eden configurations.

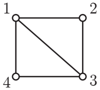
II. Heterogeneous Specifications: The input consists of two or more types of behavior.

Example: A collection Q_1 of stable configurations and another collection Q_2 of unstable configurations.

More examples follow.

Example: Collection of Stable Configurations

Graph

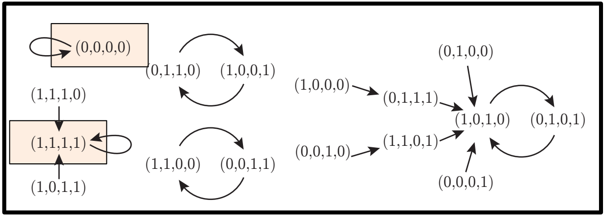


Node state set K={0,1}.

Bithreshold system $(k_{01},k_{10})=(1,3)$.

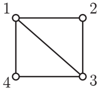
Parallel (synchronous) update: F.

Phase Space



Example: Collection of Unstable Configurations

Graph

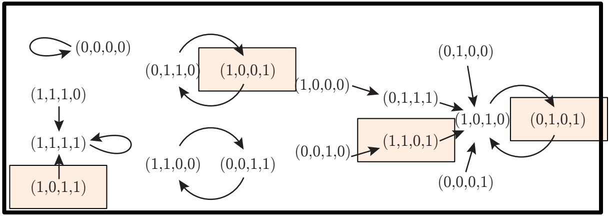


Node state set K={0,1}.

Bithreshold system $(k_{01},k_{10})=(1,3)$.

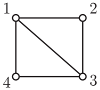
Parallel (synchronous) update: F.

Phase Space



Example: Collection of Dendrograms

Graph

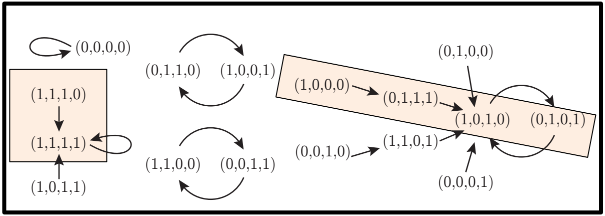


Node state set K={0,1}.

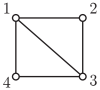
Bithreshold system $(k_{01},k_{10})=(1,3)$.

Parallel (synchronous) update: F.

Phase Space



Example: Collection of Garden-of-Eden Configurations

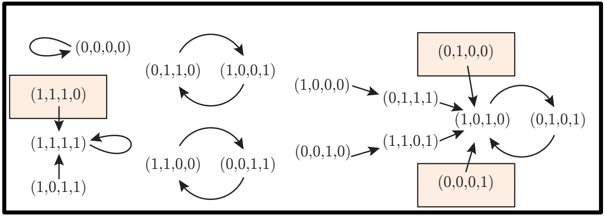
Graph 

Node state set K={0,1}.

Bithreshold system (k_{01},k_{10})=(1,3).

Parallel (synchronous) update: F.

Phase Space



Decision and Maximization Versions

- Considered for homogeneous specifications.
- **Decision version of threshold inference:** Is a threshold value for each node so that the specified property (e.g. stable configuration) is satisfied for **all the objects** in the input set?
- **Maximization version of inference problems:** Find a threshold value for each node so that the specified property (e.g. stable configuration) is satisfied for a **largest subset** of the objects in the input set.
- Maximization version is useful when the answer to the corresponding decision problem is "No".

Results for Homogeneous Specifications – Decision Versions

Behavior Spec.	**Our Result**
Stable configurations	Efficiently solvable.
Unstable configurations	Efficiently solvable.
Dendrograms	Efficiently solvable.
Garden of Eden configurations	Efficiently solvable.

Note: For each decision problem above, when the answer is "Yes", a corresponding threshold assignment can also be obtained efficiently.

Inference from Stable Configurations

Problem Name: Inferring Thresholds from Stable Configurations (ITSC).

Statement of ITSC:

- **Given:** The graph $G(V, E)$ of a synchronous dynamical system and a set of Q_1 configurations.
- **Requirement:** Find a threshold value for each node $v \in V$ such that for the resulting system, each configuration in Q_1 is stable.

Algorithm for ITSC: Basic Ideas

- Let $Q_1 = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_r\}$ be a set of **stable** configurations.
- The problem can be considered for each node $v \in V$ **separately**.
- For each v, let $Q^0_{1,v}$ ($Q^1_{1,v}$) be the subset of Q_1 s.t. for each configuration in $Q^0_{1,v}$ ($Q^1_{1,v}$), the state of v is 0 (1).
- For t_v^{low}. If $Q^0_{1,v}$ is empty, then $t_v^{low} = 0$. Else, do the following. For each $\mathcal{C}_i \in Q^0_{1,v}$, we must have $t_v > \mathcal{C}_i^v$. Thus, $t_v \geq \mathcal{C}_i^v + 1$. And so $t_v^{low} = 1 + \max_{\mathcal{C}_i \in Q^0_{1,v}} \mathcal{C}_i^v$.
- For t_v^{high}. If $Q^1_{1,v}$ is empty, then $t_v^{high} = d_v + 2$. Else, analogous reasoning gives $t_v^{high} = \min_{\mathcal{C}_i \in Q^1_{1,v}} \mathcal{C}_i^v$.
- When there is a solution, any t_v satisfying $t_v^{low} \leq t_v \leq t_v^{high}$ is valid.
- The feasibility of the set of constraints for each node can be checked efficiently.

Results for Homogeneous Specifications – Maximization Versions

Behavior Spec.	**Our Result**
Stable configurations	**NP**-hard even to approximate.
Unstable configurations	Efficiently solvable.
Dendrograms	**NP**-hard even to approximate.
Garden of Eden configurations	Efficiently solvable.

Notes:

- For stable configuration and dendrograms, there is no $O(n^{1-\epsilon})$ approximation unless $\mathbf{P} = \mathbf{NP}$.
- The hardness results hold even when the underlying graph of the SyDS is a **simple path**.

Results for Heterogeneous Specifications

Behavior specification: A set of stable configurations and another set of unstable configurations.

Our Results:

- **NP**-hard even when the underlying graph of the SyDS is a **simple path**.
- The problem is **fixed parameter tractable** with respect to **the number of unstable configurations** with no restriction on the underlying graph.

Fixed Parameter Tractability: Definition

Definition: A problem Π is **fixed parameter tractable** (FPT) with respect to parameter k if there is an algorithm for the problem with a running time of $O(h(k)\, N^r)$, where

- $h(k)$ is a function that depends *only* on k,
- N is the size of the problem instance and
- r is a constant *independent* of k.

Fixed Parameter Tractability: Examples

Example (Vertex Cover): Given a graph $G(V, E)$ and an integer k, determine whether G has a vertex cover of size k.

Straightforward algorithm: Time $= O(|V|^k |E|)$.

More sophisticated algorithm: Time $= O(2^k |V|^2)$.

- So, the Vertex Cover problem is **fixed parameter tractable** (FPT) with respect to the **size of the vertex cover**.
- Other examples appear in the book [Niedermeier, 2006].

Inference Under Heterogeneous Behavior Specifications

Problem Name: Inferring Thresholds with Stable and Unstable Configurations (ITSUC).

Statement of ITSUC:

- **Given:** The graph $G(V, E)$ of a synchronous dynamical system, two sets of configurations Q_1 and Q_2.
- **Requirement:** Find a threshold value for each node $v \in V$ such that for the resulting system, each configuration in Q_1 is stable and each configuration in Q_2 is unstable.

Notation: Let $|V| = n$, $|Q_1| = r$ and $|Q_2| = q$.

An Easy Algorithm for ITSUC

1. Consider each combination of possible threshold assignments to nodes.
2. If some combination satisfies the behavior specifications, output "Yes"; else output "No".

Running Time:

- No. of possible threshold assignments $= O(\Delta^n)$, where Δ is the maximum node degree.
- Time to test whether a given combination of threshold assignments satisfies the behavior specification $=$ $O(n\Delta(q+r))$.
- Overall running time $= O(\Delta^{n+1}\, n(q+r))$.

An Algorithm to show that ITSUC is FPT

Recall: Q_1 and Q_2 are respectively the given set of **stable** and **unstable** configurations.

Definition: A node $v \in V$ is **compatible with a configuration** $\mathcal{C} \in Q_2$ if there is a threshold value t_v for v such that

- t_v satisfies all the constraints imposed by all the configurations in Q_1, and
- the value t_v makes $\mathcal{C}$ an unstable configuration regardless of the thresholds assigned to the other nodes.

Definition: A node $v \in V$ is **compatible with a subset** R of Q_2 if v is compatible with every configuration in R.

Observation: Testing whether a node v is compatible with a subset R of Q_2 can be done in polynomial time.

An Algorithm to show that ITSUC is FPT (continued)

Notation: $\pi(Q_2)$ is the collection of all **partitions** of Q_2.

Algorithm for ITSUC:

1. **for** each partition P in $\pi(Q_2)$ **do**
   - Let $B_1, \ldots B_k$ denote the **blocks** of P.
   - Construct the bipartite graph $H_P(V, V_P, E_P)$, where V_P has a node for each block in P and E_P consists of each edge $\{x, y\}$ such that node $x \in V$ is compatible with the block of P represented by y.
   - If H_P has a matching with k edges, then output "Yes" and **stop**.
2. Output "No".

An Algorithm to show that ITSUC is FPT (continued)

Correctness: Discussed in the paper.

Running Time:

- Time τ for each iteration of the loop in Step 1 = $O(nq^2 + nq\sqrt{n+q})$. (Details in the paper.)
- Overall time = $O(|\pi(Q_2)|\,\tau)$.
- $|\pi(Q_2)| = O((q/\log q)^q)$, where $q = |Q_2|$ ([Graham et al. 1994]).

Theorem: ITSUC is fixed parameter tractable with respect to $q = |Q_2|$, the number of unstable configurations.

Part 4
Summary and Future Work

Summary of Results

- Considered Threshold Inference Problems for SyDSs.
- Motivated by applications to model calibration and parameter estimation.
- Considered homogeneous and heterogeneous behavior specifications.
- Presented hardness or easiness results for various problems.
- Also presented a fixed parameter tractability result for heterogeneous behavior specifications.

Future Work

- Study inference problems for
  - other forms of homogeneous behavior specifications,
  - other combinations of heterogeneous behavior specifications and
  - other classes of local transition functions.
- Consider inference problems for stochastic dynamical systems.

Acknowledgments

- We thank the anonymous reviewers for their helpful comments.
- This work has been funded partially by
  - DTRA Grant HDTRA1-11-1-0016,
  - DTRA CNIMS Contract HDTRA1-11-D-0016-0010,
  - NSF NetSE Grant CNS-1011769,
  - NSF SDCI Grant OCI-1032677, and
  - NIH MIDAS Grant 5U01GM070694-11.

End

END

Algorithm for ITSC: Basic Ideas

- The problem can be considered for each node $v \in V$ **separately**.
- For each node v, each configuration $\mathcal{C} \in Q_1$ imposes a constraint of the form $t_v \geq k_1$ or $t_v \leq k_2$, where t_v is the threshold for v and k_1 and k_2 are integers.
- The feasibility of the set of constraints for each node can be checked efficiently.