Synchronous Dynamical Systems on Directed Acyclic Graphs (DAGs): Complexity and Algorithms

Daniel J. Rosenkrantz[1,2] **Madhav V. Marathe**[1,3]
S. S. Ravi[1,2] **Richard E. Stearns**[1,2]

[1]Bioinformatics Institute and Initiative, University of Virginia, Charlottesville, VA

[2]Department of Computer Science, University Albany – State University of New York, Albany, NY

[3]Department of Computer Science, University of Virginia, Charlottesville, VA

Email: drosenkrantz@gmail.com, marathe@virginia.edu, ssravi0@gmail.com, thestearns2@gmail.com

Acknowledgment: This work was partially supported by NSF Grants IIS-1633028 (BIG DATA), CMMI-1745207 (EAGER), OAC-1916805 (CINES), CCF-1918656 (Expeditions) and IIS-1908530.

Talk Outline

1. Basics of Discrete Dynamical Systems on DAGs
2. Previous Work
3. Problems Considered
4. Our Main Contributions
5. Future Work

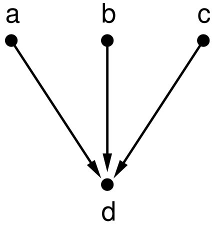
Discrete Dynamical Systems on DAGs: Basics

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

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

Notation: DAG-SyDS (Synchronous Dynamical System on a DAG)

Local Transition Functions



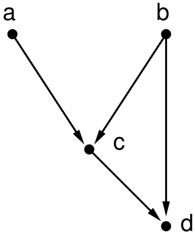
Local function f_d :

- Inputs: States of a, b, c and d.
- Output: Next state of d.

- The only input to the local function f_a is the state of a.
- A similar comment applies to the local functions f_b and f_c.

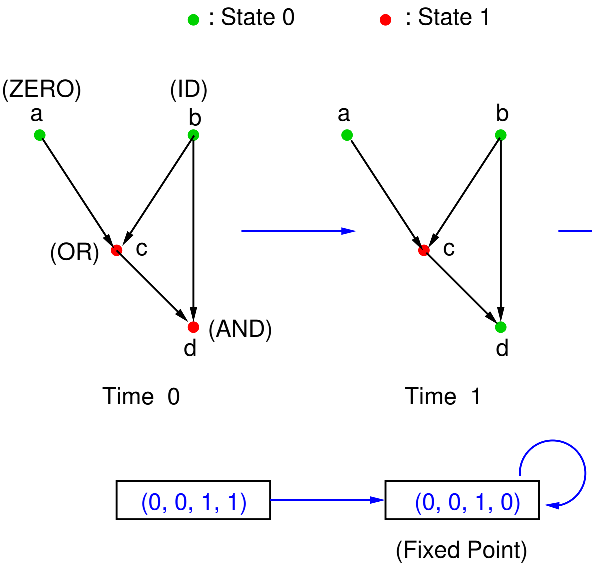
Note: The local functions are assumed to be **deterministic**.

An Example of a DAG-SyDS



- $\mathbb{B} = \{0, 1\}$
- f_a : Zero function
- f_b : Identity function
- f_c : OR function
- f_d : AND function

Time Evolution of a DAG-SyDS

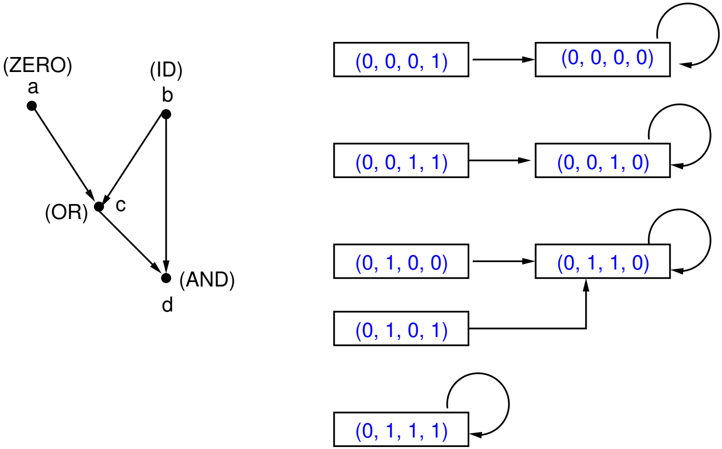


Some Definitions

- **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.
- **Fixed Point**: A configuration $\mathcal{C}$ whose successor is $\mathcal{C}$ itself.
- 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 a DAG-SyDS $\mathcal{S}$

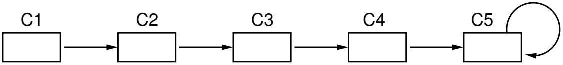


Note: Only a part of the phase space is shown.

A Few More Definitions

Transient: A directed path in the phase space ending in a directed cycle.

Example: The sequence ⟨C1, C2, C3, C4⟩ in the following figure.



Symmetric Boolean Function: A Boolean function whose value remains the same when inputs are permuted. (Example: OR, AND, XOR)

***r*-Symmetric Boolean Function:** A Boolean function whose inputs can be partitioned into r classes so that the function value remains the same when inputs in any class are permuted.

Note: Each symmetric function is 1-symmetric.

Motivation: Diffusion Phenomena in Networks

- **Contagion** processes model many social phenomena (e.g., propagation of information, influence, diseases, trends, etc.).
- A well known example: diffusion of opinions in networks (e.g., [Auletta et al. 2018, Chistikov et al. 2020, Botan et al. 2019, Bredereck & Elkind 2017]).
- 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**).
- **Discrete Dynamical Systems:** A formal model for analyzing contagion phenomena.

Some Analysis Problems for SyDSs

Reachability:

Given: A SyDS $\mathcal{S}$ and two configurations $\mathcal{C}_1$ and $\mathcal{C}_2$.

Question: Does $\mathcal{S}$ starting from $\mathcal{C}_1$ reach $\mathcal{C}_2$?

Convergence:

Given: A SyDS $\mathcal{S}$ and a configuration $\mathcal{C}_1$.

Question: Does $\mathcal{S}$ starting from $\mathcal{C}_1$ reach a fixed point?

Convergence Guarantee:

Given: A SyDS $\mathcal{S}$.

Question: Does $\mathcal{S}$ reach a fixed point from **every** starting configuration?

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

Previous Work on Analysis Problems

- Computational intractability of Reachability and Convergence problems for dynamical systems on undirected graphs (e.g., [Barrett et al. 2006, Rosenkrantz et al. 2018]).
- Computational intractability results for SyDSs on directed graphs [Ogihara & Uchizawa: 2017 & 2020].
- Computational intractability results in the AI literature for other dynamical system models:
  - Hopfield neural nets (e.g., [Orponen: 1993 & 1994]).
  - Petri nets (e.g., [Esparza & Nielsen 1994]).

Previous Work on Analysis Problems (continued)

Results of [Chistikov et al. (AAAI-2020)]:

- SyDSs on directed graphs in the context of diffusion of opinions.
- Each local function is the **majority** function: an agent changes her $\{0,1\}$ opinion only when a majority of her neighbors have a different opinion. (This function is 2-symmetric.)
- Convergence and Convergence Guarantee problems are **PSPACE**-complete for SyDSs on directed graphs where each local function is the majority function.
- The Convergence problem is efficiently solvable for DAG-SyDSs regardless of local functions.

Other Work Related to DAG-SyDSs

- For DAG-SyDSs where each local function is a **bi-threshold** function, Reachability can be solved efficiently ([Kuhlman et al. 2013]).

- Problem of controlling a SyDS (with external inputs) so that the system reaches a desirable configuration can be solved efficiently when the underlying graph is a **directed tree** ([Akutsu et al. 2007]).

- Algorithm for inferring the edges of the underlying graph of a DAG-SyDS from time-series data ([Materassi et al. 2013], [Cliff et al. 2020]).

- Use of DAG-SyDSs in studying fairness issues for learning algorithms that interact with an environment ([Creager et al. 2020]).

Our Main Contributions

Definition: A **symmetric DAG-SyDS** is a DAG-SyDS where each local function is **symmetric**.

1. Reachability is **PSPACE**-complete for symmetric DAG-SyDSs.
   - **Approach:** Reduction from Quantified 3SAT (Q3SAT).
   - Proof involves two stages.
   - First stage: Use a reduction from Q3SAT to produce a DAG-SyDS where each local function is r-symmetric for some $r \leq 6$.
   - Second stage: Show that any SyDS where each local function is r-symmetric for a *fixed* r can be simulated by another SyDS where each local function is symmetric.

Our Main Contributions (continued)

Definition: A **monotone DAG-SyDS** is a DAG-SyDS where each local function is **monotone**.

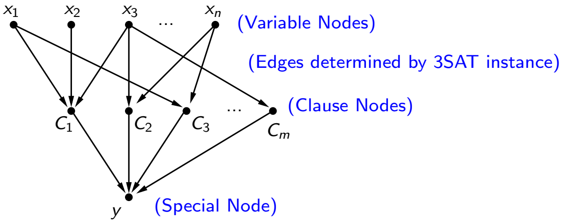
2. Reachability is efficiently solvable for monotone DAG-SyDSs.
   - **Idea:** Proof relies on two lemmas.
   - For any DAG-SyDS, the length of a transient to a fixed point is bounded by the number L of levels in the DAG ([Chistikov et al. 2020]).
   - For any DAG-SyDS with monotone local functions, every cycle in the phase space is a fixed point.

Note: For monotone SyDSs on general directed graphs, Reachability is **PSPACE**-complete [Ogihara & Uchizawa, 2017].

Our Main Contributions (continued)

3. Convergence Guarantee is **Co-NP**-complete for DAG-SyDSs with at most **three** levels.

   - **Idea:** Reduction from 3SAT.



   - Local functions ensure that the solution to Convergence Guarantee is true iff the given 3SAT instance is **not satisfiable**.

Note: For SyDSs on general directed graphs, Convergence Guarantee is **PSPACE**-complete [Chistikov et al. 2020].

Future Work

- Study Reachability and other problems for DAG-SyDSs for other classes of local functions (e.g., weighted threshold functions).
- Consider additional restrictions on DAGs (e.g., DAGs with bounded indegree).
- Study DAG-SyDSs where local functions are stochastic.

Thank You!