Synchronous Dynamical Systems on Directed Acyclic Graphs: Complexity and Algorithms

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

[1]Biocomplexity Institute and Initiative, University Virginia, Charlottesville, VA 22904. [2]Computer Science Dept., University at Albany – SUNY, Albany, NY 12222. [3]Computer Science Dept., University Virginia, Charlottesville, VA 22904.

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

Abstract

Discrete dynamical systems serve as useful formal models to study diffusion phenomena in social networks. Motivated by applications in systems biology, several recent papers have studied algorithmic and complexity aspects of diffusion problems for dynamical systems whose underlying graphs are directed, and may contain directed cycles. Such problems can be regarded as reachability problems in the phase space of the corresponding dynamical system. We show that computational intractability results for reachability problems hold even for dynamical systems on directed acyclic graphs (dags). We also show that for dynamical systems on dags where each local function is monotone, the reachability problem can be solved efficiently.

1 Introduction

Background and Motivation: Discrete dynamical systems are formal models for the study of diffusion phenomena in networks. Application areas in which these models have been used include the study of social contagions (e.g., information, opinions, fads, epidemics) and energy demand modeling (e.g., adaptation of solar energy) (Adiga et al. 2019; Chistikov et al. 2020; Ogihara and Uchizawa 2020; Gupta et al. 2018). Informally, such a dynamical system[4] consists of an underlying (social or biological) network, with each node having a state value from a domain $\mathbb{B}$. In this paper, we assume that the underlying graph is directed and that the domain is binary (i.e., $\mathbb{B}$ = $\{0,1\}$). The propagation of the contagion is modeled by a collection of Boolean local functions, one per node. For any node v, the inputs to the local function f_v at v are the current states of v and those of its in-neighbors (i.e., the nodes from which v has an incoming edges) and the output of f_v is the state of v at the next time instant. We consider the **synchronous** update model, where all nodes evaluate their local functions and update their states *in parallel*. These dynamical systems are referred to as **synchronous dynamical systems** (SyDSs) in the literature (e.g., (Adiga et al. 2019; Rosenkrantz et al. 2018)). In applications involving systems biology, such systems are also referred to as **synchronous Boolean networks** (e.g., (Kauffman et al. 2003; Ogihara and Uchizawa 2020; Akutsu et al. 2007)). Throughout this paper, we will use the term SyDS to denote such a system. In general, the local functions may be deterministic (e.g., threshold models used in social systems (Granovetter 1978)) or stochastic (e.g., the SIR model of disease propagation (Easley and Kleinberg 2010)). We consider deterministic local functions throughout this paper.

The **configuration** of a SyDS with n nodes at at time τ is a vector $(s_1^\tau, s_2^\tau, \ldots, s_n^\tau)$, where s_i^τ is the state value of node v_i at time τ, $1 \leq i \leq n$. As the nodes evaluate and update their local functions, the configuration of the system evolves over time. If a SyDS transitions from a configuration $\mathcal{C}$ to a configuration $\mathcal{C}'$ in one time step, we say that $\mathcal{C}'$ is the **successor** of $\mathcal{C}$. A configuration $\mathcal{C}$ which is its own successor is called a **fixed point**. Thus, once a SyDS reaches a fixed point, no further state changes occur at any node.

When using SyDSs as models of social or biological phenomena, it is of interest to study whether a given SyDS starting from a specified initial configuration may reach certain (desirable or undesirable) configurations. For example, in the context of information propagation, where the state value 1 indicates that a node has received the information propagating through the network, one may be interested in configurations where many nodes are in state 1. On the other hand, in contexts such as disease propagation where the state of 1 indicates that a node has been infected, configurations where many nodes are in state 1 are undesirable. One can study such **configuration reachability** problems by considering another directed graph, called the **phase space**, of the dynamical system (Mortveit and Reidys 2007). Each node in the phase space is a configuration and there is an edge from a node X to node Y if the SyDS can transition from the configuration represented by X to that represented by Y in one step. Thus, the configuration reachability problem becomes a directed path problem in phase space. For a SyDS with n nodes where the state of each node from $\{0, 1\}$, the number of possible configurations is 2^n; thus, the size of the phase space of a SyDS is *exponential* in the size of the SyDS. There has been a considerable amount of research whose goal is to understand when such reachability problems are efficiently solvable and when they are computationally intractable. Specifically, several papers (e.g., (Ogihara and Uchizawa 2017; Akutsu et al. 2007)) have presented computational intractability results for reachability

[4] A formal definition is provided in Section 2.

problems for SyDSs over directed graphs. Recently, Chistikov et al. (2020) studied two reachability problems[5] (called Convergence and Convergence Guarantee) in the context of opinion propagation. They show that these problems are PSPACE-complete for general directed graphs. They also show that the Convergence problem can be solved efficiently when the underlying directed graph is *acylic* (i.e., it does not have any directed cycle), regardless of the local functions at each node. Following standard graph theoretic terminology, we refer to directed acylic graphs as **dags** (Cormen et al. 2009). Several computational problems for discrete dynamical systems on dags have been addressed in the literature; these problems will be discussed in the related work section. Here, we study the complexity of reachability problems for SyDSs whose underlying graphs are dags. We refer to such SyDSs as DAG-SyDSs. Our results are summarized below.

Summary of Results and Their Significance:

(1) Results on the structure of the phase space: For any DAG-SyDS, we show that the length of every phase space cycle is a power of 2. If the number of levels in the underlying DAG of a given SyDS is L, then no phase space cycle is longer than 2^L, and no transient (i.e., a directed path leading to a directed cycle) is longer than $2^L - 1$ (Theorem 3.1). Moreover, for each L, we observe that there is an L-level DAG SyDS that achieves these bounds (Theorem 3.2). Such structural properties are useful in solving reachability problems; if the maximum lengths of transients and cycles are bounded by polynomial functions of the size of a given SyDS, then reachability problems for that SyDS can be solved efficiently (by running the SyDS for a polynomial number of steps).

(2) Complexity of Reachability for DAG-SyDSs: It was shown in (Chistikov et al. 2020) that the Convergence problem (i.e., given a SyDS $\mathcal{S}$ and an initial configuration, does $\mathcal{S}$ reach a fixed point?) is PSPACE-complete for SyDSs on directed networks. They also showed that the convergence problem can be solved efficiently for DAG-SyDSs. We show that the reachability problem (i.e., given a SyDS $\mathcal{S}$ and two configurations $\mathcal{C}$ and $\mathcal{D}$, does $\mathcal{S}$ starting from $\mathcal{C}$ reach $\mathcal{D}$?), which is similar to the Convergence problem, is PSPACE-complete for DAG-SyDSs even when each local function is symmetric (Crama and Hammer 2011). To our knowledge, no hardness result is currently known for the Reachability problem for DAG-SyDSs. Our proof of this result involves two major steps. The first step uses a reduction from the Quantified Boolean Formulas (QBF) problem (Garey and Johnson 1979) to show that the Reachability problem for DAG-SyDSs is PSPACE-complete even when each local function is r-symmetric[6] for some constant r (Theorem 4.1). For the second step, we define the concept of an **embedding** of one SyDS into another, and use this concept to show that for any fixed value of r, the reachability problem for DAG-SyDSs with r-symmetric local functions is polynomial-time reducible to the Reachability problem for DAG-SyDSs with symmetric local functions (Theorem 4.3). The PSPACE-completeness of the Reachability problem for DAG-SyDSs with symmetric local function (Theorem 4.4) follows directly from Theorems 4.1 and 4.3.

The difference between the complexities of Convergence and Reachability problems for DAG-SyDSs is due to the following. For any DAG-SyDS with L levels, regardless of the local functions, the length of any transient leading to a fixed point is bounded by L. (This is a slight restatement of Proposition 1 in (Chistikov et al. 2020).) Thus, the Convergence problem for DAG-SyDSs is efficiently solvable. On the other hand, we show that one can construct DAG-SyDSs with specific local functions such that the length of a transient and that of the cycle the transient leads to are both exponential in L (Theorem 3.2). The idea behind this construction enables us to establish our PSPACE-hardness result for the reachability problem for DAG-SyDSs.

(3) Complexity of Convergence Guarantee for DAG-SyDSs: It was shown in (Chistikov et al. 2020) that the Convergence Guarantee problem (i.e., given a SyDS $\mathcal{S}$ on a directed graph, does $\mathcal{S}$ reach a fixed point from every initial configuration?) is PSPACE-complete for SyDSs on general directed graphs. For DAG-SyDSs, we show that the problem is Co-NP-complete, even when restricted to dags with at most three levels (Theorem 5.1). Thus, our result points out that the problem remains computationally intractable even for DAG-SyDSs.

(4) Reachability for Monotone DAG-SyDSs: We show that the Reachability problem is efficiently solvable for DAG-SyDSs whose local functions are monotone. This is done by showing that for such DAG-SyDSs, the length of each transient is at most the number of levels in the dag and that every cycle in the phase space is a fixed point (Theorem 6.1). These two properties directly imply the efficient solvability of the reachability problem for monotone DAG-SyDSs. In contrast, the reachability problem is known to be PSPACE-complete monotone SyDSs when the underlying directed graph has cycles (Ogihara and Uchizawa 2017).

For space reasons, proofs of several results are either sketched or omitted; detailed proofs can be found in (Rosenkrantz et al. 2020).

Related Work: Reachability problems for various models of discrete dynamical systems have been widely studied in the AI literature. Examples include Hopfield neural nets (Floréen and Orponen 1989; Orponen 1993, 1994) and Petri nets (Esparza and Nielsen 1994). Problems related to the diffusion of opinions and other contagions have also been studied under various discrete dynamical systems models (see e.g., (Auletta, Ferraioli, and Greco 2018; Botan, Grandi, and Perrussel 2019; Chistikov et al. 2020; Bredereck and Elkind 2017)).

As stated earlier, synchronous dynamical systems over directed graphs serve as useful models for biological phenomena (Kauffman et al. 2003). Many researchers have studied reachability and related problems for such dynamical systems (see e.g., (Ogihara and Uchizawa 2020, 2017)). Even though the PSPACE-completeness result for the reachabil-

[5] These problems will be defined shortly.

[6] The definition of r-symmetric functions is given in Section 2. Symmetric Boolean functions are 1-symmetric. As will be explained in Section 2, the local functions used in (Chistikov et al. 2020) are 2-symmetric.

ity problem considered in (Rosenkrantz et al. 2018) is stated in terms of undirected graphs, it can be easily extended to general directed graphs.

Many researchers have presented results for various computational problems for dynamical systems on dags. We now provide a summary of these results. Akutsu et al. (2007) showed that the problem of controlling a synchronous Boolean network so that it reaches a desired configuration is NP-hard for general directed graphs but is efficiently solvable when the underlying graph is a directed tree. This problem is different from the reachability problem considered here; in particular, the variables that are used to control a network are external to the network and the goal of the controller is to choose appropriate values for those variables at each time step. Materassi and Salapaka (2013) considered the problem of inferring from time-series data the edges in the underlying dag for a dynamical system. A similar problem is considered by Cliff et al. (2020) where the underlying dag is assumed to represent relationships between latent variables of a probabilistic graphical model. Creager et al. (2020) point out that dynamical systems on dags provide a unifying framework for studying fairness issues that arise when a learning algorithm must interact with a dynamically changing environment. Arnold et al. (2019) discuss methods for comparing the effectiveness of several modeling approaches when the underlying causal model for an epidemic can be represented as a dag. To our knowledge, the reachability problem for DAG-SyDSs was first considered in (Kuhlman et al. 2013). They showed that for DAG-SyDSs, when each local function is a bi-threshold function (i.e., each node has two threshold values to control the $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions), the reachability problem can be solved efficiently.

2 Definitions and Notation

SyDSs on directed graphs: Let $\mathbb{B}$ denote the Boolean domain {0,1}. In this paper, a **Synchronous Dynamical System** (SyDS) $\mathcal{S}$ over $\mathbb{B}$ is assumed to be specified as a pair $\mathcal{S} = (G, \mathcal{F})$, where (i) $G(V, E)$, a directed graph with $|V| = n$, represents the underlying graph of the SyDS, with node set V and (directed) edge set E, and (ii) $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ is a collection of functions in the system, with f_i denoting the **local transition function** associated with node v_i, $1 \leq i \leq n$. Each node of G has a state value from $\mathbb{B}$. Each Boolean function f_i specifies the local interaction between node v_i and its in-neighbors in G. The inputs to function f_i are the state of v_i and those of the in-neighbors of v_i in G; function f_i maps each combination of inputs to a value in $\mathbb{B}$. This value becomes the next state of node v_i. It is assumed that each local function can be evaluated in polynomial time. In a SyDS, all nodes compute and update their next state *synchronously*. Other update disciplines (e.g., sequential updates) have also been considered in the literature (e.g., (Mortveit and Reidys 2007)). At any time τ, the **configuration** $\mathcal{C}$ of a SyDS is the n-vector $(s_1^\tau, s_2^\tau, \ldots, s_n^\tau)$, where $s_i^\tau \in \mathbb{B}$ is the state of node v_i at time τ $(1 \leq i \leq n)$. A **DAG-SyDS** is a SyDS whose underlying graph is a dag.

Example: An example of a DAG-SyDS is shown in Figure 1. Here, the local functions at nodes v_1 through v_6 are Identity, Complement, Identity, AND, OR and XOR respectively. (The Identity and Complement functions have only one input; they return respectively the input value and its complement.) Suppose the initial configuration of the SyDS is $(0, 0, 1, 0, 0, 0)$; that is, at time $\tau = 0$, v_3 is in state 1 and all other nodes are in state 0. Recall that the inputs to the local function at a node v are the current state of v and those of its in-neighbors (if any). Since the local functions at v_1 and v_3 are Identity functions, the states of v_1 and v_3 will remain 0 and 1 respectively in all subsequent time steps. Since the local function at v_2 is the Complement function, the state of v_2 at time $\tau = 1$ is 1. Using the local functions at v_4, v_5 and v_6, one can verify that their states at time $\tau = 1$ are 0, 1 and 0 respectively. Thus, the configuration at time $\tau = 1$ is $(0, 1, 1, 0, 1, 0)$. Similarly, the configuration at times τ = 2, 3, 4 and 5 are $(0, 0, 1, 0, 1, 0)$, $(0, 1, 1, 0, 1, 1)$, $(0, 0, 1, 0, 1, 1)$ and $(0, 1, 1, 0, 1, 0)$ respectively. Thus, the configuration at time $\tau = 5$ is the same as that at $\tau = 1$. In other words, the SyDS cycles among the four configurations at times $\tau = 1$ through $\tau = 4$.

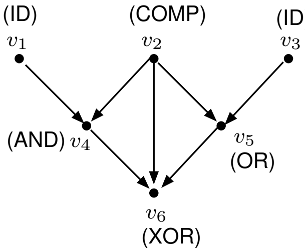

Figure 1: An example of a DAG-SyDS. The local functions at nodes v_1 through v_6 are Identity, Complement, Identity, AND, OR and XOR respectively.

Additional definitions regarding SyDSs: Some additional concepts related to SyDSs are defined below.

Definition 2.1 *For the local transition function f_v of a given node v of a directed SyDS, we call the variable corresponding to the source node of each incoming edge to v an* **incoming variable** *of f_v, and call the variable corresponding to v the* **self variable** *of f_v.*

The **phase space** of a SyDS was defined in Section 1. A **transient** in phase space is a simple directed path P whose last node is part of a directed cycle which does not contain any other node of P. The length of a phase space cycle is the number of edges in the cycle, and the length of a transient is the number of edges in the transient.

Let $\mathcal{S}$ be a SyDS. For a given configuration $\mathcal{C}$ and node v, we let $\mathcal{C}(v)$ denote the state of node v in $\mathcal{C}$. For a given configuration $\mathcal{C}$ and set of nodes Y, we let $\mathcal{C}[Y]$ denote the projection of $\mathcal{C}$ onto Y. We assume that the initial configuration of the system occurs at time 0. For a given initial configuration $\mathcal{C}$ and nonnegative integer i, we let $\mathcal{C}_i$ denote the configuration of $\mathcal{S}$ at time i.

For a given initial configuration $\mathcal{C}$, we say that a given node v is **stable** at time t if for all $i \geq t$, $\mathcal{C}_i(v) = \mathcal{C}_t(v)$.

Also, we say that a given node v is **alternating** at time t if for all $i \geq 0$, $\mathcal{C}_{t+2i}(v) = \mathcal{C}_t(v)$ and $\mathcal{C}_{t+2i+1}(v) = \overline{\mathcal{C}_t(v)}$ (i.e., the state of v alternates between 0 and 1).

Graph theoretic definitions: Given a dag, we say that node u **directly precedes** node v if the graph contains an edge from u to v, that u **precedes** v if the graph contains a path (possibly with no edges) from u to v, and that u **properly precedes** v if the graph contains a path with at least one edge from u to v. Note that the local transition function for a given node v in a directed SyDS is a function of v and the nodes that directly precede v.

Definition 2.2 *For a dag $G(V, E)$, the* ***level*** *of a node $v \in V$ is the maximum number of edges in any directed path to v.*

Suppose a given dag $G(V, E)$ has L levels. For each j, $0 \leq j < L$, we let $\mathcal{L}_j$ denote the set of nodes at level j, and let $\mathcal{L}'_j$ denote the nodes whose level is at most j.

Boolean function definitions: Here, we provide definitions of several classes of Boolean functions used in this paper. These definitions are from (Crama and Hammer 2011; Barrett et al. 2006).

Consider assignments α and β to a set of Boolean variables X. We say that $\alpha \leq \beta$ if for every variable $x \in X$, $\alpha(x) \leq \beta(x)$. A Boolean function f is **monotone nondecreasing** if for every pair of assignments α and β to its variables, $\alpha \leq \beta$ implies that $f(\alpha) \leq f(\beta)$, and is **monotone nonincreasing** if for every pair of assignments α and β to its variables, $\alpha \leq \beta$ implies that $f(\alpha) \geq f(\beta)$. We say that f is **monotone** if f is either monotone nondecreasing or monotone nonincreasing. Common examples of monotone functions are OR and AND. A **monotone SyDS** is a SyDS whose local transition functions are all monotone.

A **symmetric** Boolean function is one whose value does not depend on the order in which the input bits are specified; that is, the function value depends only on how many of its inputs are 1. For example, the XOR function is symmetric since the value of the function is determined by the parity of the number of 1's in the input.

In symmetric Boolean functions, we need not know whether a specific input is 0 or 1; it is enough to know the number of 1s in the input. A similar situation arises in anonymous symmetric pure strategy graphical games where each player's strategy is from {0,1}. A player X doesn't know whether a specific neighbor chose 0 or 1; X knows how many neighbors chose 1. Convergence in SyDSs corresponds to reaching a fixed point where everyone is satisfied with the outcome. This is similar to a Nash equilibrium in the game theoretic setting. In this sense, the convergence problem is related to the problem of Nash equilibria in such games and reachability is useful in understanding whether players can achieve certain equilibria (see e.g., (Blonski 1999; Daskalakis and Papadimitriou 2007, 2015; Jackson 2010)).

A Boolean function f is r**-symmetric** if the inputs to f can be partitioned into at most r classes such that the value of f depends only on how many of the inputs in each of the r classes are 1. Note that symmetric functions defined above are 1-symmetric. For each node v, the local (majority) function used in (Chistikov et al. 2020) to study opinion diffusion can be seen to be 2-symmetric: one class contains just the node v and the other class contains all the in-neighbors of v. The value of the function is determined by the number of 1's in these two classes.

A SyDS is r-symmetric if each of its local transition functions is r'-symmetric for some $r' \leq r$.

Problem definitions: We now provide formal specifications of the problems considered in this paper.

Reachability

Instance: A SyDS $\mathcal{S}$ specified by an underlying directed graph $G(V, E)$ and a local transition function f_v for each node $v \in V$ and two configurations $\mathcal{C}$ and $\mathcal{D}$.
Question: Starting from configuration $\mathcal{C}$, does $\mathcal{S}$ reach configuration $\mathcal{D}$?

Convergence

Instance: A SyDS $\mathcal{S}$ specified by a directed graph $G(V, E)$, a local function f_v for each $v \in V$ and a configuration $\mathcal{C}$.
Question: Starting from configuration $\mathcal{C}$, does $\mathcal{S}$ reach a fixed point?

Convergence Guarantee

Instance: A SyDS $\mathcal{S}$ specified by a directed graph $G(V, E)$ and a local function f_v for each $v \in V$.
Question: Does $\mathcal{S}$ reach a fixed point from *every* initial configuration?

The formulation of the Convergence Guarantee problem in (Chistikov et al. 2020) considers the negation of the above question (i.e., is there a configuration $\mathcal{C}$ from which the SyDS does not reach a fixed point?). However, since the complexity class PSPACE is closed under complement (Papadimitriou 1993), there is no difference in the complexity of the two versions of the problem.

3 DAG-SyDSs: Some Phase Space Properties

In this section, we present some properties of the phase spaces of DAG-SyDSs. These properties are independent of the local functions at the nodes of the DAG-SyDS. Our first result points out that the phase spaces of DAG-SyDSs may contain exponentially long cycles.

Proposition 3.1 *For any $n > 1$, there is an n node DAG-SyDS whose phase space graph is a cycle of length 2^n.*

Proof: For a given $n > 1$ we construct the DAG-SyDS S_n to be a counter, as follows. The underlying graph contains n levels, one node per level. For each node, there is an incoming edge from the nodes on each of the lower levels. The transition function for each node is the function that retains the current value of the node if any of the lower order bits is 0, and changes the value of the node if all of the lower order bits are 1.

Suppose a given configuration of S_n is interpreted as encoding an integer k, $0 \leq k < 2^n$. Then the successor configuration encodes the integer $k + 1 \pmod{2^n}$. Thus, the phase space of S_n is a cycle of length 2^n. ■

For any SyDS, any infinitely long phase space path consists of a transient (possibly of length 0), followed by an infinite number of repetitions of a basic cycle. We now show that for any DAG-SyDS, the length of every phase space cycle is a power of 2. Moreover, the lengths of the longest transient and the longest phase space cycle are each bounded by an exponential function of the number of levels in the underlying graph of the SyDS. We begin with a lemma whose proof appears in (Rosenkrantz et al. 2020).

Lemma 3.1 *For a given DAG-SyDS, and a given initial configuration, suppose that the node values of all incoming edges to a given node v are stable at time t. Then node v is either alternating at time t, or stable at time $t + 1$.*

Lemma 3.2 *For a DAG-SyDS, every level 0 node is either alternating at time 0 or stable at time 1.*

Proof: A level 0 node has no incoming edges, so the result follows from Lemma 3.1. ■

We will also use the following result which is a slight restatement of Proposition 1 in (Chistikov et al. 2020).

Proposition 3.2 *For a DAG-SyDS, the length of a transient leading to a fixed point does not exceed the number of levels of the SyDS.*

We can now state our result on the lengths of cycles and transients in DAG-SyDSs.

Theorem 3.1 *For a DAG-SyDS, the length of every phase space cycle is a power of 2. Moreover, if the number of levels of a given acyclic SyDS is L, then no phase space cycle is longer than 2^L, and no transient is longer than $2^L - 1$.*

Proof (idea): We use induction on the number of levels. The details appear in (Rosenkrantz et al. 2020).

Theorem 3.1 provides upper bounds on the lengths of transients and cycles in DAG-SyDSs. We now present a result that provides matching lower bounds on cycle and transient lengths.

Theorem 3.2 *For every $L \geq 1$, there is a DAG-SyDS with L levels whose phase space contains a transient of length $2^L - 1$, leading to a cycle of length 2^L.*

Proof: See (Rosenkrantz et al. 2020).

4 Reachability Problem for DAG-SyDSs

Here, we establish the complexity of the Reachability problem for DAG-SyDSs with symmetric local functions. Our proof uses two major steps. In the first step (Section 4.1), we show that the PSPACE-hardness for DAG-SyDSs whose local functions are r-symmetric for a constant value of r. In the second step (Section 4.2), we show that the Reachability problem resulting from the first step can be reduced to the same problem for DAG-SyDSs where each local function is symmetric. We begin with the first step.

4.1 Reachability Problem for DAG-SyDSs with r-Symmetric Local Functions

This section establishes the following result.

Theorem 4.1 *The reachability problem is PSPACE-complete for DAG-SyDSs where each local function is r-symmetric for some $r \leq 6$.*

Proof: It is easy to see that the problem is in PSPACE. The proof of PSPACE-hardness is via a reduction from the Quantified Boolean Formulas (QBF) problem which is known to be PSPACE-complete (Garey and Johnson 1979). Let F denote the given quantified Boolean formula. Let f denote the Boolean expression that is quantified. Without loss of generality, we can assume that f is a 3SAT formula (Garey and Johnson 1979). Let $X = \{x_0, x_1, \ldots, x_{n-1}\}$ be the set of variables of f, and $\{c_0, c_1, \ldots, c_{m-1}\}$ be the set of clauses of f. Let F be $(Q_{n-1}x_{n-1}) \cdots (Q_1x_1)(Q_0x_0)f$, where each Q_i is either $\forall$ or $\exists$.

We give a reduction where the constructed DAG-SyDS is 6-symmetric (i.e., each local function is r-symmetric for some $r \leq 6$).

For each i, $0 \leq i < n$, we use the notation $\otimes_i$ to denote a binary Boolean operation as follows. If Q_i is $\forall$, then $\otimes_i$ is *and*; if Q_i is $\exists$, then $\otimes_i$ is *or*.

For the reduction, we construct a reachability problem instance whose SyDS $\mathcal{S}$ has an underlying graph with $2n + m + 2$ nodes, on $2n + 2$ levels.

SyDS $\mathcal{S}$ contains the following nodes. We let $Y = \{y_0, y_1, \ldots, y_n\}$ be a set of $n + 1$ nodes, $R = \{r_0, r_1, \ldots, r_{n-1}\}$ be a set of n nodes, $W = \{w_0, w_1, \ldots, w_{m-1}\}$ be a set of m nodes, and h be an extra node.

The initial configuration $\mathcal{C}$ for the constructed problem instance has the states of all nodes equal to 0.

The goal configuration $\mathcal{D}$ for the constructed problem instance has $h = 1$, $r_{n-1} = 1$, each $r_i = 0$ for $0 \leq i < n - 1$, each $y_i = 0$, and each w_j equal to 1 iff the corresponding clause c_j contains a positive literal.

Node set Y will function as a cyclical counter, with incoming edges and local transition functions similar to those for the n nodes in the construction presented in our proof of Proposition 3.1. For each i, $0 \leq i < n$, node y_i will correspond to the variable x_i in f.

Each node $w_i \in W$ will correspond to clause c_i.

Each node $r_i \in R$ will correspond to the quantified Boolean formula $(Q_ix_i) \cdots (Q_0x_0)f$. In particular, at some point in the operation of $\mathcal{S}$, node r_{n-1} will have the value of the quantified Boolean formula F.

Node h will serve as a control node. Once the value of node h equals 1, it remains 1 forever more, forces node r_{n-1} to retain its value, and forces all the other nodes in R to have value 0.

The underlying graph of $\mathcal{S}$ has directed edges (y_i, y_j) for each i and j such that $0 \leq i < j \leq n$. For each node w_j, there are incoming edges from those Y nodes corresponding to the variables occurring in clause c_j. Node h has incoming edges from the $n + 1$ nodes in Y. Node r_0 has an incoming edge from y_0, h, and from each node in W. For each node r_i, $1 \leq i < n$, there are incoming edges from r_{i-1}, h, and each y_j where $0 \leq j \leq i$.

The local transition function for each node in Y is the function that retains the current value of the node if any of the incoming variables equals 0, and changes the value of the node otherwise.

The local transition function for each node $w_i \in W$ is the same as clause c_i, but using the values of the incoming variables.

The local transition function for h is as follows. If $h = 1$, then 1. If $h = 0$ and the $n+1$ incoming edges from Y encode the integer $2^n + n - 1$, then 1. Otherwise, 0.

The local transition function for r_0 is as follows. If $h = 1$, then 0. Otherwise, if $y_0 = 1$, then the *and* of the incoming variables from W. Otherwise, operation $\otimes_0$ applied to r_0 and the *and* of the incoming variables from W.

The local transition function for r_i , $1 \le i < n - 1$, is as follows. If $h = 1$, then 0. Otherwise, if the $i + 1$ incoming edges from Y encode the integer $2^i + i$, then r_{i-1}. Otherwise, if the incoming edges from Y encode the integer i, then operation $\otimes_i$ applied to r_i and r_{i-1}. Otherwise, r_i.

The local transition function for r_{n-1} is as follows. If $h = 0$ and the n incoming edges from Y encode the integer $2^{n-1} + n - 1$, then r_{n-2}. If $h = 0$ and the n incoming edges from Y encode the integer $n-1$, then operation $\otimes_{n-1}$ applied to r_{n-1} and r_{n-2}. Otherwise, r_{n-1}.

Note that SyDS $\mathcal{S}$ is 6-symmetric; the local transition functions for nodes r_i , $1 \le i \le n - 1$, contain 6 symmetry classes.

For $0 \le i < n$, we let F_i be a Boolean function of $n-i-1$ variables, as follows:

$$F_i(x_{n-1}, x_{n-2}, \ldots, x_{i+1}) = (Q_i x_i) \cdots (Q_1 x_1)(Q_0 x_0) f(X)$$

We now establish the correctness of the reduction.

Let β be a tuple, possibly empty, of Boolean values. Let $l(\beta)$ denote the degree of β, i.e., the number of variables in β. Let $k(\beta)$ denote the integer encoded by β. (If β is empty, then $k(\beta) = 0$.) For β such that $l(\beta) < n$, let $j(\beta) = n - l(\beta) - 1$, and let $t(\beta) = (k(\beta) + 1)2^{n-l(\beta)} + n - l(\beta)$.

For any $t \ge 0$, let X^t be the assignment to the variables X of f where $X^t(x_i) = \mathcal{C}_t(y_i)$, $0 \le i \le n - 1$. From the proof in Theorem 3.1, the first n nodes of $\mathcal{S}$ undergo a phase space cycle whose length is 2^n. Thus, all 2^n possible assignments to X occur during this cycle. In particular, all 2^n assignments to X are in the set $\{X^t \,|\, 0 \le t < 2^n\}$. Specifically, for any assignment α to X, $X^{k(\alpha)} = \alpha$.

For any assignment α to X, let $W(\alpha)$ be the assignment of values to the nodes set W where w_i equals the value of clause c_i for assignment α, $0 \le i \le m - 1$. Then, for all $t \ge 0$, $\mathcal{C}_{t+1}[W] = W(X^t)$.

For any given tuple β of $n - i - 1$ Boolean values corresponding to values of the variables $x_{n-1}, x_{n-2}, \ldots, x_{i+1}$, note that $j(\beta) = i$. Moreover, the value of $F_{j(\beta)}(\beta) = F_i(x_{n-1}, x_{n-2}, \ldots, x_{i+1})$ is determined by the 2^{i+1} assignments to X occurring at times $k(\beta)2^{n-l(\beta)}$ through $(k(\beta) + 1)2^{n-l(\beta)} - 1$.

Claim 1: For all j', $0 \le j' < n$, for all β such that $j(\beta) = j'$, $F_{j(\beta)}(\beta) = \mathcal{C}_{t(\beta)}(r_{j(\beta)})$.

We prove the claim by induction on j'.

Basis Step: Suppose that $j' = 0$. Consider any β such that $j(\beta) = 0$. Then $l(\beta) = n - 1$, $t(\beta) = 2k(\beta) + 3$, and the variables in β are $(x_{n-1}, x_{n-2}, \ldots, x_1)$. By definition,

$$F_0(x_{n-1}, \ldots, x_1) = (Q_0 x_0)\, f(x_{n-1}, \ldots, x_1, x_0).$$

Thus,

$$F_0(x_{n-1}, \ldots, x_1) = \\ f(x_{n-1}, \ldots, x_1, 0) \otimes_0 f(x_{n-1}, \ldots, x_1, 1).$$

Let γ^0 = $(x_{n-1}, x_{n-2}, \ldots, x_1, 0)$ and γ^1 = $(x_{n-1}, x_{n-2}, \ldots, x_1, 1)$. Then, $F_0(\beta) = f(\gamma^0) \otimes_0 f(\gamma^1)$. Note that $\gamma^0 = X^{k(\gamma^0)}$ and $\gamma^1 = X^{k(\gamma^1)}$. Also, $k(\gamma^0) = 2k(\beta)$ and $k(\gamma^1) = 2k(\beta) + 1$. Thus, at time $2k(\beta) + 1$, $W = W[\gamma^0]$; and at time $2k(\beta)+2$, $W = W[\gamma^1]$. Since at time $2k(\beta)+1$, $h = 0$, and $y_0 = 0$, the local transition function for r_0 sets $\mathcal{C}_{2k(\beta)+2}(r_0)$ to be the *and* of the values of the incoming edges from W at time $2k(\beta) + 1$. Thus, $\mathcal{C}_{2k(\beta)+2}(r_0) = f(\gamma^0)$. Since at time $2k(\beta) + 2$, $h = 0$, and $y_0 = 0$, the local transition function for r_0 sets $\mathcal{C}_{2k(\beta)+3}(r_0)$ to be result of $\otimes_i$ applied to $\mathcal{C}_{2k(\beta)+2}(r_0)$ and the *and* of the values of the incoming edges from W at time $2k(\beta) + 2$. Thus, $\mathcal{C}_{2k(\beta)+3}(r_0) = f(\gamma^0) \otimes_i f(\gamma^1) = F_0(\beta)$. This proves the claim for $j' = 0$.

Inductive Step: Now assume that the claim holds for a given value of j', $0 \le j' < n - 1$. We want to prove that the claim holds for $j' + 1$. Consider any β such that $j(\beta) = j' + 1$. We need to show that $F_{j'+1}(\beta) = \mathcal{C}_{t(\beta)}(r_{j'+1})$. We first note that $l(\beta) = n - j' - 2$, $t(\beta) = (k(\beta) + 1)2^{j'+2} + j' + 2$, and the variables in β are $(x_{n-1}, x_{n-2}, \ldots, x_{j'+2})$. (If $j' = n - 2$, then β contains no variables.) By definition, $F_{j'+1}(\beta) = (Q_{j'+1} x_{j'+1}) \cdots (Q_1 x_1)(Q_0 x_0) f(X)$. Thus,

$$F_{j'+1}(x_{n-1}, x_{n-2}, \ldots, x_{j'+2}) = \\ F_{j'}(x_{n-1}, x_{n-2}, \ldots, x_{j'+2}, 0) \otimes_{j'+1} \\ F_{j'}(x_{n-1}, x_{n-2}, \ldots, x_{j'+2}, 1).$$

Let γ^0 = $(x_{n-1}, x_{n-2}, \ldots, x_{j'+2}, 0)$ and γ^1 = $(x_{n-1}, x_{n-2}, \ldots, x_{j'+2}, 1)$. Then, $F_{j'+1}(\beta) = F_{j'}(\gamma^0) \otimes_{j'+1} F_{j'}(\gamma^1)$. Note that $t(\gamma^0) = (k(\gamma^0) + 1)2^{j'+1} + j' + 1$. Since $k(\gamma^0) = 2k(\beta)$, $t(\gamma^0) = (2k(\beta) + 1)2^{j'+1} + j' + 1 = k(\beta)2^{j'+2} + 2^{j'+1} + j' + 1$. Also, $k(\gamma^1) = 2k(\beta) + 1$, and $t(\gamma^1) = (k(\gamma^1) + 1)2^{j'+1} + j' + 1 = (2k(\beta) + 2)2^{j'+1} + j' + 1 = (k(\beta) + 1)2^{j'+2} + j' + 1$. By the inductive hypothesis, $F_{j'}(\gamma^0) = \mathcal{C}_{t(\gamma^0)}(r_{j(\gamma^0)}) = \mathcal{C}_{t(\gamma^0)}(r_{j'})$, and $F_{j'}(\gamma^1) = \mathcal{C}_{t(\gamma^1)}(r_{j(\gamma^1)}) = \mathcal{C}_{t(\gamma^1)}(r_{j'})$.

At time $t(\gamma^0) = k(\beta)2^{j'+2} + 2^{j'+1} + j' + 1$, the $j' + 2$ incoming edges to $r_{j'+1}$ from Y encode the integer $2^{j'+1} + j' + 1$, so the local transition function for $r_{j'+1}$ evaluates to be the value of $r_{j'}$. Thus, $\mathcal{C}_{t(\gamma^0)+1}(r_{j'+1}) = \mathcal{C}_{t(\gamma^0)}(r_{j'}) = F_{j'}(\gamma^0)$. The local transition function for $r_{j'+1}$ has $r_{j'+1}$ retain its current value until $t(\gamma^1) = (k(\beta)+1)2^{j'+2} + j' + 1$. At time $t(\gamma^1) = (k(\beta) + 1)2^{j'+2} + j' + 1$, the $j' + 2$ incoming edges to $r_{j'+1}$ from Y encode the integer $j' + 1$, so the local transition function for $r_{j'+1}$ evaluates to be the result of applying operation $\otimes_{j'+1}$ to $r_{j'+1}$ and $r_{j'}$. Thus, $\mathcal{C}_{t(\gamma^1)+1}(r_{j'+1}) = \mathcal{C}_{t(\gamma^1)}(r_{j'+1}) \otimes_{j'+1} \mathcal{C}_{t(\gamma^1)}(r_{j'}) = F_{j'}(\gamma^0) \otimes_{j'+1} F_{j'}(\gamma^1) = F_{j'+1}(\beta)$. However, note that $t(\beta) = (k(\beta) + 1)2^{j'+2} + j' + 2 = t(\gamma^1) + 1$. Thus, $F_{j'+1}(\beta) = \mathcal{C}_{t(\beta)}(r_{j'+1})$, and Claim 1 follows.

Note that each node in Y and W cycles with a period that divides 2^{n+1}. Since $\mathcal{D}$ has all nodes in Y equal to 0, and this only occurs at a time that is a multiple of 2^{n+1}, $\mathcal{D}$ is reachable iff it is reachable at a time that is a multiple of

2^{n+1}.

Note that $\mathcal{C}(h) = 0$ and $\mathcal{D}(h) = 1$. Furthermore, node h is stable with value 1 at time $2^n + n$, and h is equal to 0 at all prior times. Thus $\mathcal{D}$ is reachable iff it is reachable at a time t such that $t \geq 2^n + n$. Considering node set Y, $\mathcal{D}$ is reachable iff it is reachable at a time that is a nonzero multiple of 2^{n+1}.

Note that every node of $\mathcal{S}$ has the same value for every time that is a nonzero multiple of 2^{n+1}. Thus, $\mathcal{D}$ is reachable iff it is reachable at time 2^{n+1}, i.e., iff $\mathcal{D} = \mathcal{C}_{2^{n+1}}$.

Recall that $\mathcal{D}(h) = \mathcal{C}_{2^{n+1}}(h)$ and $\mathcal{D}[Y] = \mathcal{C}_{2^{n+1}}[Y]$. Since $\mathcal{C}_{2^{n+1}-1}[Y]$ consists of all 1's, each $\mathcal{C}_{2^{n+1}}(w_j)$ equals 1 iff the corresponding clause c_j contains a positive literal. Thus, $\mathcal{D}[W] = \mathcal{C}_{2^{n+1}}[W]$. The nodes in $R - \{r_{n-1}\}$ are stable with value 0 at time $2^n + n + 1$, and so have the same value in $\mathcal{D}$ and $\mathcal{C}_{2^{n+1}}$. Thus, $\mathcal{D} = \mathcal{C}_{2^{n+1}}$ iff $\mathcal{D}(r_{n-1}) = \mathcal{C}_{2^{n+1}}(r_{n-1})$. Since $\mathcal{D}(r_{n-1}) = 1$, $\mathcal{D} = \mathcal{C}_{2^{n+1}}$ iff $\mathcal{C}_{2^{n+1}}(r_{n-1}) = 1$.

Note that node r_{n-1} is stable at time $2^n + n$, so $\mathcal{C}_{2^{n+1}}(r_{n-1}) = \mathcal{C}_{2^n+n}(r_{n-1})$. Thus $\mathcal{D}$ is reachable iff $\mathcal{C}_{2^n+n}(r_{n-1}) = 1$. Let β be the empty tuple. As has been shown above, $F_{j(\beta)}(\beta) = \mathcal{C}_{t(\beta)}(r_{j(\beta)})$. Note that $l(\beta) = 0$, $k(\beta) = 0$, $j(\beta) = n - 1$, and $t(\beta) = (k(\beta) + 1)2^{n-l(\beta)} + n - l(\beta) = 2^n + n$. Thus, $\mathcal{C}_{2^n+n}(r_{n-1}) = F_{n-1}$, which is the value of the given quantified Boolean formula F. Thus F is true iff $\mathcal{D}$ is reachable from $\mathcal{C}$, and this completes our proof of Theorem 4.1. ∎

4.2 Reducing the Number of Symmetry Classes

In this section, we show how the reachability problem for r-symmetric SyDSs (for any fixed r) can be reduced to the same problem for symmetric SyDSs. In particular, this reduction ensures that when we start with a DAG-SyDS, the reduction also produces a DAG-SyDS. Our approach uses the idea of embedding, which is defined below. In this definition, we use the notation $\mathcal{C} \rightarrow \mathcal{D}$ to indicate that $\mathcal{D}$ is the successor of $\mathcal{C}$.

Definition 4.1 *Given two SyDSs $\mathcal{S} = (G(V, E), \mathcal{F})$ and $\mathcal{S}' = (G'(V', E'), \mathcal{F}')$, let h be an onto function from V' to V. Let $\hat{h}$ be the function that maps each configuration $\mathcal{C}$ of $\mathcal{S}$ into the configuration $\mathcal{C}'$ of $\mathcal{S}'$ such that for each node $v' \in V'$, $\mathcal{C}'(v') = \mathcal{C}(h(v'))$. We say that h is an **embedding** of $\mathcal{S}$ into $\mathcal{S}'$ if for each configuration $\mathcal{C}$ of $\mathcal{S}$, letting $\mathcal{D}$ be the configuration such that $\mathcal{C} \rightarrow \mathcal{D}$ in $\mathcal{S}$, $\hat{h}(\mathcal{C}) \rightarrow \hat{h}(\mathcal{D})$ in $\mathcal{S}'$.*

The next lemma shows a property of an embedding.

Lemma 4.1 *Given SyDSs $\mathcal{S}$ and $\mathcal{S}'$, and an embedding h from $\mathcal{S}$ into $\mathcal{S}'$, the reachability problem for $\mathcal{S}$ is polynomially reducible to the reachability problem for $\mathcal{S}'$.*

Proof: In Definition 4.1, since h is an onto function, $\hat{h}$ is injective. For every configuration $\mathcal{C}$ of $\mathcal{S}$ and every $t \geq 0$, by induction on t, $\hat{h}(\mathcal{C}_t) = (\hat{h}(\mathcal{C}))_t$. Thus, $\mathcal{C}$ reaches a given configuration $\mathcal{D}$ iff $\hat{h}(\mathcal{C})$ reaches the configuration $\hat{h}(\mathcal{D})$. ∎

We can now state a result which shows that a suitable embedding can be constructed efficiently.

Theorem 4.2 *For any fixed value of r, there is a polynomial time algorithm that given a directed r-symmetric SyDS $\mathcal{S} = (G(V, E), \mathcal{F})$, constructs a directed symmetric SyDS $\mathcal{S}' = (G'(V', E'), \mathcal{F}')$ and an embedding h of $\mathcal{S}$ into $\mathcal{S}'$. Moreover, if G is acyclic, then so is G'.*

Proof: See (Rosenkrantz et al. 2020).

The following is the main result of Section 4.2.

Theorem 4.3 *For any fixed value of r, the reachability problem for directed r-symmetric SyDSs is polynomial-time reducible to the reachability problem for directed symmetric SyDSs, and the reachability problem for DAG r-symmetric SyDSs is polynomial-time reducible to the reachability problem for DAG symmetric SyDSs.*

Proof: From Lemma 4.1 and Theorem 4.2. ∎

Theorems 4.1 and 4.3 together imply the following main result of Section 4.

Theorem 4.4 *The Reachability problem for DAG-SyDS with symmetric local functions is* PSPACE*-complete.* ∎

5 Convergence Guarantee for DAG-SyDSs

As mentioned earlier, (Chistikov et al. 2020) showed that the convergence guarantee problem is PSPACE-complete for SyDSs on general directed graphs. Here, we show that this problem remains computationally intractable even for DAG SyDSs.

Theorem 5.1 *The convergence guarantee problem for DAG SyDSs is co-NP-complete, even when restricted to dags with three levels and 3-symmetric functions.*

Proof (idea): The reduction is from 3SAT (Garey and Johnson 1979). The details are given in (Rosenkrantz et al. 2020).

6 Results for Monotone DAG-SyDSs

The class of monotone Boolean functions were defined in Section 2. A monotone DAG-SyDS is a DAG-SyDS in which every local function is monotone. In this section, we show that the reachability problem for monotone DAG-SyDSs can be solved efficiently. This result is based on the following theorem.

Theorem 6.1 *For a DAG monotone SyDS, every phase space cycle is a fixed point, and the length of any transient does not exceed the number of levels.*

Proof: See (Rosenkrantz et al. 2020).

The following is a direct corollary of Theorem 6.1.

Corollary 6.1 *The Reachability problem is efficiently solvable for monotone DAG-SyDSs.*

7 Directions for Future Work

We conclude by mentioning two directions for future research. For example, it is of interest to consider DAG-SyDSs where local functions are from other classes (e.g., weighted threshold functions (Crama and Hammer 2011)). Similarly, one may also consider restrictions on the graph structure (e.g., dags where nodes have bounded in-degrees) and investigate the complexity of reachability and other problems.

Acknowledgments

We thank the referees for carefully reading the paper and providing very helpful feedback. This work is partially supported by University of Virginia Strategic Investment Fund award number SIF160 and NSF Grants IIS-1633028 (BIG DATA), CMMI-1745207 (EAGER), OAC-1916805 (CINES), CCF-1918656 (Expeditions), CNS-2028004 (RAPID), OAC-2027541 (RAPID) and IIS-1908530.

Ethical Impact

The work reported in this paper addresses some complexity and algorithmic issues associated with formal models for contagion propagation in social networks. The results are in the form of theorems and algorithms that are useful in understanding the strengths and weaknesses of different formal models. Our work does not involve experiments on public or private data.

References

Adiga, A.; Kuhlman, C. J.; Marathe, M. V.; Mortveit, H. S.; Ravi, S. S.; and Vullikanti, A. 2019. Graphical Dynamical Systems and their Applications to Bio-Social Systems. *Springer International Journal of Advances in Engineering Sciences and Applied Mathematics* 11(2): 153–171.

Akutsu, T.; Hayashida, M.; Ching, W.; and Ng, M. K. 2007. Control of Boolean Networks: Hardness results and algorithms for tree structured networks. *Journal of Theoretical Biology* 244: 670–679.

Arnold, K. F.; Harrison, W. J.; Heppenstall, A. J.; and Gilthorpe, M. S. 2019. DAG-informed regression modelling, agent-based modelling and microsimulation modelling: A critical comparison of methods for causal inference. *International Journal of Epidemiology* 48(1): 243–253.

Auletta, V.; Ferraioli, D.; and Greco, G. 2018. Reasoning about Consensus when Opinions Diffuse through Majority Dynamics. In *Proc. IJCAI*, 49–55.

Barrett, C. L.; Hunt III, H. B.; Marathe, M. V.; Ravi, S. S.; Rosenkrantz, D. J.; and Stearns, R. E. 2006. Complexity of reachability problems for finite discrete dynamical systems. *Journal of Computer and System Sciences* 72(8): 1317–1345.

Blonski, M. 1999. Anonymous games with binary actions. *Games and Economic Behavior* 28(2): 171–180.

Botan, S.; Grandi, U.; and Perrussel, L. 2019. Multi-Issue Opinion Diffusion under Constraints. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS*, 828–836.

Bredereck, R.; and Elkind, E. 2017. Manipulating Opinion Diffusion in Social Networks. In *Proc. IJCAI*, 894–900.

Chistikov, D.; Lisowski, G.; Paterson, M.; and Turrini, P. 2020. Convergence of Opinion Diffusion is PSPACE-Complete. In *Proc. AAAI*, 7103–7110. AAAI Press.

Cliff, O. M.; Prokopenko, M.; and Fitch, R. 2020. Inferring Coupling of Distributed Dynamical Systems via Transfer Entropy. ArXiv Report: 1611.00549v1.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2009. *Introduction to Algorithms*. Cambridge, MA: MIT Press and McGraw-Hill, Second edition.

Crama, Y.; and Hammer, P. 2011. *Boolean Functions: Theory, Algorithms, and Applications*. New York, NY: Cambridge University Press.

Creager, E.; Madras, D.; Pitassi, T.; and Zemel, R. 2020. Causal modeling for fairness in dynamical systems. In *International Conference on Machine Learning*, 2185–2195. PMLR.

Daskalakis, C.; and Papadimitriou, C. H. 2007. Computing equilibria in anonymous games. In *48th Annual IEEE FOCS*, 83–93. IEEE.

Daskalakis, C.; and Papadimitriou, C. H. 2015. Approximate Nash equilibria in anonymous games. *Journal of Economic Theory* 156: 207–245.

Easley, D.; and Kleinberg, J. 2010. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press.

Esparza, J.; and Nielsen, M. 1994. Decidability Issues for Petri Nets. Tech. Report BRICS RS-94-8, Department of Computer Science, University of Aarhus, Denmark.

Floréen, P.; and Orponen, P. 1989. On the Computational Complexity of Analyzing Hopfield Nets. *Complex Systems* 3(6): 577–587.

Garey, M. R.; and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-completeness*. San Francisco: W. H. Freeman & Co.

Granovetter, M. 1978. Threshold models of collective behavior. *American Journal of Sociology* 83(6): 1420–1443.

Gupta, A.; Swarup, S.; Marathe, A.; Vullikanti, A. K.; Lakkaraju, K.; and Letchford, J. 2018. Designing Incentives to Maximize the Adoption of Rooftop Solar Technology. In André, E.; Koenig, S.; Dastani, M.; and Sukthankar, G., eds., *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, 1950–1952. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM.

Jackson, M. O. 2010. *Social and economic networks*. Princeton, NJ: Princeton University Press.

Kauffman, S.; Peterson, C.; Samuelsson, B.; and Troein, C. 2003. Random Boolean network models and the yeast transcriptional network. *Proc. National Academy of Sciences (PNAS)* 100(25): 14796–14799.

Kuhlman, C. J.; Kumar, A.; Marathe, M. V.; Ravi, S. S.; Rosenkrantz, D. J.; and Stearns, R. E. 2013. Analysis Problems for Special Classes of Bi-threshold Dynamical Systems. In *Proc. Workshop on Multi-Agent Interaction Networks (MAIN 2013), held in conjunction with the 12th AAMAS conference*, 26–33.

Materassi, D.; and Salapaka, M. V. 2013. Reconstruction of directed acyclic networks of dynamical systems. In *2013 American Control Conference*, 4687–4692. IEEE.

Mortveit, H.; and Reidys, C. 2007. *An Introduction to Sequential Dynamical Systems*. New York, NY: Springer Science & Business Media.

Ogihara, M.; and Uchizawa, K. 2017. Computational complexity studies of synchronous Boolean finite dynamical systems on directed graphs. *Information and Computation* 256: 226–236.

Ogihara, M.; and Uchizawa, K. 2020. Synchronous Boolean Finite Dynamical Systems on Directed Graphs over XOR Functions. In *Proc. 45th MFCS*, 76:1–76:13.

Orponen, P. 1993. On the Computational Power of Discrete Hopfield Nets. In *Proc. International Colloquium on Automata, Languages and Programming (ICALP)*, volume 700 of *Lecture Notes in Computer Science*, 215–226.

Orponen, P. 1994. Neural Networks and Complexity Theory. *Nord. J. Comput.* 1(1): 94–110.

Papadimitriou, C. H. 1993. *Computational Complexity*. Reading, MA: Addison Wesley Longman.

Rosenkrantz, D. J.; Marathe, M. V.; Ravi, S. S.; and Stearns, R. E. 2018. Testing Phase Space Properties of Synchronous Dynamical Systems with Nested Canalyzing Local Functions. In *Proc. 17th AAMAS*, 1585–1594.

Rosenkrantz, D. J.; Marathe, M. V.; Ravi, S. S.; and Stearns, R. E. 2020. Synchronous Dynamical Systems on Directed Acyclic Graphs (DAGs): Complexity and Algorithms. Technical Report 20-155, Biocomplexity Institute and Initiative, University of Virginia, Charlottesville, VA, USA.