Inferring Users' Choice Functions in Networked Social Systems Through Active Queries

Abhijin Adiga, Chris J. Kuhlman, Madhav V. Marathe, S. S. Ravi,
Daniel J. Rosenkrantz and Richard E. Stearns

Abstract

Using synchronous dynamical systems (SyDSs) as a formal model for networked social systems, we study the problem of inferring users' choices in such systems. We observe that SyDSs with deterministic and probabilistic threshold functions as local functions can capture users' choices in the context of contagion propagation in social networks. We use an active query mechanism where a user interacts with a system by submitting queries, and the responses to the queries are used to infer the local functions. We develop methods that provide provably efficient query sets for inferring both deterministic and probabilistic forms of threshold functions. We also present experimental results using real world social networks.

1 Introduction

1.1 Motivation

Inferring unknown parameters of systems is currently an active area of research (see, e.g., [1, 6, 7, 12, 17, 28, 30]). In this paper, we study the problem of inferring the behaviors of users in networked social systems. Using a discrete dynamical system [5, 22] as the formal model of a networked system, we observe that in several contexts, behaviors that represent users' choices can be captured by classes of threshold functions. Thus, in such contexts, the problem of learning users' behaviors can be formally cast as that of learning the local threshold functions of discrete dynamical systems.

One approach for the inference of user behavior is based on observing a system (e.g., [1, 12, 23]) while another approach uses active interaction with the system in the form of queries (e.g., [2, 6, 17]). We focus on the latter approach and consider two versions of threshold functions, namely Granovetter-style deterministic thresholds [14] and probabilistic thresholds [4], to represent users' behaviors. We also consider symmetric functions (defined in Section 2) which properly contain threshold functions. Threshold models have been used by many researchers in the context of contagion propagation in social networks (e.g., [1, 7, 12]). Granovetter-style threshold functions capture simple deterministic choices by the entities comprising the social network. Here, if a node v's threshold is t_v, then v chooses to change its state from 0 to 1 during a time step τ if at least t_v of v's neighbors are in state 1 at time step $\tau - 1$. This type of threshold behavior has been used to study several types of social behavior such as initiation or cessation of smoking, joining or leaving a protest movement, spreading of Twitter hashtags, group coordination, joining an online health forum, and rare outbreaks [8, 12, 18, 24, 25, 27, 29]. Probabilistic threshold functions represent a more general form of social choice by including an additional degree of freedom, namely, a probability value. In one example of this class of functions, each node v is associated with two parameters: a threshold value t_v and a probability value p_v. If the number of neighbors of v in state 1 is less than t_v, the next state of v is 0. However, when the number of neighbors of v in state 1 is at least t_v, the next state of v is 1 with probability p_v and 0 with probability $1 - p_v$. Note that when $p_v = 1$, the probabilistic model coincides with the deterministic threshold model. The probabilistic threshold model has been used to

capture the propagation of epidemics and other social phenomena in populations [3,4,10,20]. Additional application contexts where threshold models have been used will be presented in the related work section.

To discuss our contributions, we first need to present an informal description of a synchronous dynamical system (SyDS); a formal description appears in Section 2. A SyDS is specified by an underlying undirected graph $G(V, E)$, where $|V| = n$. Each node $v \in V$ has a state value from $\{0, 1\}$ which changes over time. Each node v also has a local function f_v. The inputs to f_v are the current state of v and those of its neighbors, and the output of f_v is the next state of v. The configuration $\mathcal{C}$ of a system at time τ is the n-tuple where the i^{th} component of $\mathcal{C}$ represents the state of node v_i at time τ, $1 \leq i \leq n$. Given a configuration $\mathcal{C}$ at time τ, the **successor** $\mathcal{C}$ is the configuration[1] of the system at time $\tau + 1$. In our formulations, we assume that the underlying network of the SyDS is given and that the local function at each node is a form of threshold function. A **query** q to the system specifies a configuration and the response to the query is the successor of q. We can now provide an informal description of the *inference problem*: given the underlying network of a threshold SyDS and the model for its local functions, query the system and determine the local function at each node from the responses produced by the system for the queries. An additional goal is to use as few queries as possible. We consider two query models, namely **batch** (where all the queries must be submitted in one group) and **adaptive** (where the responses received for some queries can be used in constructing new queries).

This work is part of an ongoing project to define and demonstrate next-generation social science principles. Among the goals of the project are to develop scalable, formal, and rigorous methods for reproducible and transparent social science. The results in this paper take concrete steps to realize these goals in that formal methods are provided and evaluated for inferring properties of networked social systems. By learning properties of such systems, a more rigorous and transparent understanding of these types of user choice models and user behaviors can be gained.

1.2 Summary of Contributions

Our contributions are summarized below. We use n to denote the number of nodes in the given SyDS and Δ to denote the maximum node degree of the underlying graph.

1. For SyDSs with symmetric local functions, we show that under the batch model, there exists a query set Q of size $O\big(\Delta(\log \Delta)^{2.5}\big)$ such that from the responses to the queries in Q, the symmetric function at each node can be correctly identified. We establish this upper bound through a probabilistic argument that uses a result of Füredi and Kahn [11] on hypergraphs. Our upper bound provides an asymptotic improvement over a result in [2] where it is shown that for SyDSs with symmetric local functions, there is a query set of size $O(\Delta^{1.5} \log n)$ which can be used to correctly identify all the local functions with probability at least $1 - \frac{1}{n}$.

2. For any integer $n \geq 1$, we show that there are SyDSs with $N \geq n$ nodes where each local function is a threshold function such that for such SyDSs, there are query sets of size $\Omega(2^N)$ from which one cannot correctly infer all the threshold values. (Note that for a deterministic SyDS with N nodes, the maximum number of possible queries, which is equal to the number of distinct configurations, is 2^N.) The significance of this result is explained in Section 4.

[1] When the system is deterministic, the successor of a configuration is unique. In probabilistic systems, the successor may not be unique.

3. We present a randomized algorithm that generates query sets to infer probabilistic threshold functions. We show that our algorithm provides good performance guarantees on the threshold as well as the probability values.

4. We present experimental results obtained by implementing our algorithm mentioned in Item 3 above on real world networks. We address four topics. First, we show how the algorithm makes progress toward estimating thresholds by converging to true thresholds. Second, we describe the value and penalty associated with repeating queries that are necessary for stochastic local functions. Third, we show how the maximum number of queries required to estimate thresholds tends toward the mean number of queries as the transition probability and number of query repetitions increase. Fourth, we compare the number of queries required for estimating thresholds for stochastic and deterministic local functions.

1.3 Related Work

The problem of learning the parameters of unknown systems has attracted a lot of attention in the literature. We first discuss the work where inference is done by observing a system. The problem of learning finite automata and normal forms of Boolean functions are studied in [23] and [16] respectively. For social networks, González-Bailón et al. [12] present techniques for learning thresholds of nodes in a Twitter network using data from retweets. Algorithms for learning thresholds of a dynamical system using information about the system's trajectories are presented in [1]. Also, several researchers (e.g., [13,26]) have addressed the problem of learning influence probabilities in social networks from observed data such as system logs and timed traces. In contrast to inference from passive observations, a number of researchers have recently studied the use of active interactions with a system in learning the system's parameters. For example, Bei et al. [6] discuss methods for a seller to learn utility functions of buyers in a market by announcing prices of items and obtaining a set of goods that each buyer is willing to procure. Urschel et al. [28] present algorithms for inferring parameters of certain probabilistic processes (called determinantal point processes) through an appropriate sampling procedure. Zhou et al. [30] show how one can use queries to learn Nash equilibria in two-player games where the probability distributions used by players in choosing strategies are unknown. The use of queries to learn users' choices from a finite set of ranked options is studied in [15,17]. The use of active queries to infer the local functions of a SyDS was introduced in [2]. That paper presented methods for generating query sets for SyDSs where the local functions are either deterministic threshold functions or symmetric functions. Probabilistic threshold functions were not considered in [2].

2 Preliminaries

2.1 Synchronous Dynamical Systems

We follow the presentation in [1] to discuss the basic definitions associated with discrete dynamical systems. Let $\mathbb{B}$ denote the Boolean domain $\{0,1\}$. A **Synchronous Dynamical System** (SyDS) $\mathcal{S}$ over $\mathbb{B}$ is specified as a pair $\mathcal{S} = (G, \mathcal{F})$, where (a) $G(V, E)$, an undirected graph with $|V| = n$, represents the underlying graph of the SyDS, with node set V and edge set E, and (b) $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ is a collection of functions in the system, with f_i denoting the **local function** associated with node v_i, $1 \leq i \leq n$. Each node of G has a state value from $\mathbb{B}$. Each function f_i specifies the local interaction between node v_i and its neighbors in G. The inputs to function f_i are the state of v_i and those of the 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 computed efficiently.

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 $(1 \leq i \leq n)$. Given a configuration $\mathcal{C}$, the state of a node v in $\mathcal{C}$ is denoted by $\mathcal{C}(v)$. 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., [22]). If a given SyDS can transition in one step from a configuration $\mathcal{C}'$ to a configuration $\mathcal{C}$, then $\mathcal{C}$ is a **successor** of $\mathcal{C}'$.

Given a graph $G(V, E)$ and a node $v_i \in V$, the **closed neighborhood** of v_i, denoted by $N[v_i]$, is defined by $N[v_i] = \{v_i\} \cup \{v_j : \{v_i, v_j\} \in E\}$. Thus, the inputs to the local function f_i at v_i are the states of the nodes in $N[v_i]$. For any node v and configuration $\mathcal{C}$, the **score** of v in $\mathcal{C}$, denoted by score$(v, \mathcal{C})$, is the number of nodes in $N[v]$ whose state value in $\mathcal{C}$ is 1. Thus, the score of v in configuration $\mathcal{C}$ gives the number of 1's in the input to the local interaction function f_v when $\mathcal{C}$ is the current configuration of the SyDS.

2.2 Local Interaction Functions

We consider three classes of local interaction functions (or simply local functions).

(a) Threshold Functions. The local function f_v associated with node v of a SyDS $\mathcal{S}$ is a t_v-**threshold** function for some integer $t_v \geq 0$ if the following condition holds: the value of f_v is 1 if v's score in the current configuration is *at least* t_v; otherwise, the value of the function is 0. Letting d_v denote the degree of node v, the number of inputs to the function f_v is $d_v + 1$. Thus, we assume that $0 \leq t_v \leq d_v + 2$. (The threshold values 0 and $d_v + 2$ allow us to realize local functions that always output 1 and 0 respectively.)

(b) Symmetric Functions. The local function f_v associated with node v of a SyDS $\mathcal{S}$ is **symmetric** if the value of f_v depends only on the number of 1's in the input to f_v [9]. It is easy to see that each threshold function is also a symmetric function.

(c) Probabilistic Threshold Functions. Here, each node v is associated with two parameters: a threshold t_v and a probability p_v. The local function f_v is defined as follows. Let σ_v denote v's score in the current configuration. If $\sigma_v < t_v$, the output of f_v is 0; otherwise (i.e., $\sigma_v \geq t_v$), the output of f_v is 1 with probability p_v and 0 with probability $1 - p_v$. The nodes are assumed to make choices *independently* of each other. Probabilistic threshold functions generalize the class of threshold functions since when $p_v = 1$, a probabilistic threshold function coincides with the corresponding threshold function.

When each local function is a symmetric function, the corresponding SyDS is *deterministic*; thus, each configuration has exactly one successor. However, when one or more local functions are probabilistic threshold functions, the transition from one configuration to another is stochastic; thus, a configuration may have two or more successors. An example of a SyDS with deterministic and probabilistic threshold functions is given in the appendix.

2.3 Query Model

Our focus is on determining the local functions of a SyDS by actively interacting with the system. This interaction is in the form of queries to the system. Each query q specifies a system configuration. When the system is deterministic, the output for query q is the successor of q. When the system is stochastic, the system may return any successor of q. As in [2], we consider two modes of interaction, namely the **batch** mode (where all the queries must be submitted together) and **adaptive** mode (where a new query may be constructed using the responses to the previous queries).

Given a query q, we use $\text{score}(v, q)$ to denote the score of v with respect to q, that is, the number of 1's in the input to the local function f_v. We say that a query set Q is **complete** if for every node v and every integer $j \in [0 \,..\, d_v + 1]$, there is a query $q \in Q$ such that $\text{score}(v, q) = j$. It can be seen that for SyDSs where each local function is a deterministic threshold function, all the threshold values can be correctly identified from the responses to the queries in any complete query set. We use $\mathbf{0}$ ($\mathbf{1}$) to denote the query in which the state of each node is 0 (1).

2.4 Other Graph Theoretic Concepts

Given an undirected graph $G(V, E)$ and a pair of nodes u and v, the **distance** between u and v, denoted by $\delta_G(u, v)$, is the length of a shortest path (in terms of edges) between u and v in G. We assume that for any node u, $\delta_G(u, u) = 0$. The **square** of G, denoted by $G^2(V, E')$, is defined as follows: for any pair of nodes u and v, G^2 has the edge $\{u, v\}$ iff $\delta_G(u, v) \leq 2$. The **closed distance-2 neighborhood** of a node v in G, denoted by $N[v, G^2]$, is defined by $N[v, G^2] = \{u \,:\, \delta_G(u, v) \leq 2\}$.

A **hypergraph** $H(V_H, E_H)$ consists of a node set V_H and a hyperedge set E_H. Each hyperedge $h \in E_H$ is a nonempty subset of V_H. Any graph can be regarded as a hypergraph in which each hyperedge has exactly two elements. We will use hypergraphs in proving the results in Section 3.

3 Complete Query Sets for Symmetric Functions

In this section, we show an upper bound of $O(\Delta(\log \Delta)^{2.5})$ on the size of complete query sets under the batch mode for SyDSs with symmetric local functions. As mentioned in Section 1.2, this is an asymptotic improvement over the upper bound established in [2].

We begin with a number theoretic definition. Given a positive real number x, we use $\langle x \rangle$ to denote the integer obtained by rounding x to the nearest integer; that is, $\langle x \rangle = \lfloor x \rfloor$ if the fractional part of x is *less than* 0.5; otherwise, $\langle x \rangle = \lceil x \rceil$. We use the following technical lemma whose proof appears in the appendix.

Lemma 3.1. *Let b, d and D be positive integers such that $b \leq d \leq D$ and let $z = \left\langle \frac{bD}{d} \right\rangle$. Then, $\binom{d}{b} \left(\frac{z}{D}\right)^b \left(1 - \frac{z}{D}\right)^{d-b} \geq \frac{1}{11\sqrt{d+1}}$.* ■

Our next lemma provides an important intermediate result regarding query sets. This lemma is a key ingredient in our proof of the main result of this section (Theorem 3.5). To state this lemma, we need to introduce some notation. Suppose $G(V, E)$ is a graph and $V' \subseteq V$. We use $N[v, V']$ to denote the closed neighborhood of v restricted to V'; that is, $N[v, V'] = N[v] \cap V'$.

Lemma 3.2. *Let $G(V, E)$ be the underlying graph of a SyDS with symmetric local functions. Let V' be a non-empty subset of V such that $\forall v \in V$, $|N[v, V']| \leq \ell$. Then, there exists a set Q with at most $22\ell^{3/2} \log |V'| + 2$ queries such that (i) $\forall v \in V \setminus V'$ and $q \in Q$, $q(v) = 0$ and (ii) $\forall v \in V$ and every $i \in \{0, 1, \ldots, |N[v, V']|\}$, there exists $q \in Q$ such that $\text{score}(v, q) = i$.*

Proof. The all zeros query $\mathbf{0}$ is such that $\forall v \in V$, $\text{score}(v, \mathbf{0}) = 0$. The configuration q with all nodes of V' in state 1 and the other nodes in state 0 yields $\forall v \in V, \text{score}(v, q) = |N[v, V']|$. These two queries account for the additive term of 2 in the number of queries. Let $\mathbb{D}(p, V')$ be the distribution where each $q \sim \mathbb{D}(p, V')$ is constructed as follows: $\forall v \in V'$, $\Pr(q(v) = 1) = p$

and $\forall v \in V \setminus V', q(v) = 0$. Let $Q = \{q_{ij} \sim \mathbb{D}(\frac{i}{\ell}, V') \mid 1 \le i < \ell,\ 1 \le j \le 22\ell\sqrt{\ell+1}\log|V'|\}$. Let $d'[v] = |N[v, V']|$. For any $b \in \{1, \ldots, d'[v]\}$ and $q \sim \mathbb{D}(z, V')$, where $z = \left\langle \frac{b\ell}{d'[v]} \right\rangle$,

$$\Pr\big(\text{score}(v,q) = b\big) \ge \binom{d'[v]}{b}\left(\frac{z}{\ell}\right)^b\left(1-\frac{z}{\ell}\right)^{d'[v]-b} \ge \frac{1}{11\sqrt{d'[v]+1}} \ge \frac{1}{11\sqrt{\ell+1}}, \qquad (1)$$

where the second inequality follows from Lemma 3.1.

$$\begin{aligned}\Pr\big(\text{score}(v,q) \neq b \text{ for any } q \in Q\big) &\le \Pr\big(\text{score}(v,q_j) \neq b,\ 1 \le j \le 22\ell\sqrt{\ell+1}\log|V'|\big)\\ &\le \left(1-\frac{1}{11\sqrt{\ell+1}}\right)^{22\ell\sqrt{\ell+1}\log|V'|} < e^{-2\ell\log|V'|}.\end{aligned}$$

Since for every v, $|N[v, V']| \le \ell$, the number of distinct closed neighborhoods restricted to V' is at most $\sum_{i=1}^{\ell}\binom{|V'|}{i} \le \left(\frac{e|V'|}{\ell}\right)^{\ell}$. Note that if $\exists v \in V$ and $b \in \{1, \ldots, \ell\}$ such that $\text{score}(v, q) \neq b$ for any $q \in Q$, then there is a subset of V' of size $\le \ell$ for which in no query, b vertices are in state 1. Therefore, using union bound,

$$\Pr\big(\exists v \in V, b \in \{1, \ldots, \ell\} \text{ such that } \text{score}(v,q) \neq b,\ \forall q \in Q\big) \le \left(\frac{e|V'|}{\ell}\right)^{\ell} e^{-2\ell\log|V'|} < 1.$$

Hence, there exists a query set of size at most $|Q| = (\ell-1) \times 22\sqrt{\ell+1}\log|V'| + 2 < 22\ell^{3/2}\log|V'| + 2$ that satisfies the conditions in the statement of the lemma. □

We also use the following two lemmas of Füredi and Kahn [11] based on the Lovász Local Lemma [21].

Lemma 3.3. *Let $\mathcal{H}(V, E_{\mathcal{H}})$ be a hypergraph on a set of n elements V such that each hyperedge has at most b elements and each element belongs to at most b hyperedges, where $b \ge 500$. Then, V can be partitioned into $\alpha = \left\lceil \frac{b}{\log b} \right\rceil$ sets $X_1, X_2, \ldots, X_\alpha$ of V such that $|H \cap X_i| \le \lceil 4.7 \log b \rceil$ for all $H \in E_{\mathcal{H}}$.*

Lemma 3.4. *Let $V' \subseteq V$ such that $\forall v \in V$, $|N[v, V']| \le \ell$. Then, V' can be partitioned into $k \le (\ell-1)\Delta + 1$ sets $V'_1, \ldots, V'_k$ such that $\forall v$, $|N[v, V'] \cap V'_j| \le 1$ for $1 \le j \le k$.*

We can now prove the main result of this section.

Theorem 3.5. *Let $G(V, E)$ be the underlying graph of a SyDS $\mathcal{S}$ with symmetric local functions. Let Δ denote the maximum node degree in G. Under the batch mode, there is a complete query set Q for $\mathcal{S}$ with $|Q| = O\big(\Delta(\log\Delta)^{2.5}\big)$.*

Proof. It is shown in [2] that for any SyDS with symmetric local functions, there is a complete query set Q with $|Q| \le \Delta^2 + 1$. For $\Delta \le 500$, we can choose an appropriate constant c such that $\Delta^2 < c\Delta(\log\Delta)^{2.5}$; thus, the theorem holds when $\Delta \le 500$. Therefore, for the rest of the proof, we will assume that $\Delta > 500$.

For any graph G, let $\mathcal{H}$ be the hypergraph where each hyperedge H_v corresponds to the closed neighborhood of vertex v. Since the maximum degree is Δ, for all v, $|H_v| \le \Delta + 1$ and v belongs to at most $\Delta + 1$ hyperedges. By Lemma 3.3, for any $\Delta \ge 500$, the vertices of G can be partitioned into $\alpha \le \left\lceil \frac{\Delta+1}{\log(\Delta+1)} \right\rceil$ subsets $X_1, X_2, \ldots, X_\alpha$, such that every vertex is adjacent to at most $\ell = \lceil 4.7\log(\Delta+1) \rceil$ vertices in any X_i. For $1 \le i \le \alpha$, let $n_{\le i}(v)$ denote the number of neighbors of v in $\bigcup_{j \le i} X_j$. The query set Q is structured in the following

manner. It is partitioned into α subsets $Q_1, Q_2, \ldots, Q_\alpha$ such that Q_i determines $f_v(b)$, $b = n_{\leq i-1}(v)+1, \ldots, n_{\leq i}(v)$ for each v, where $n_{\leq 0}(v) = 0$ by definition. In the remaining part, we will show that this can be achieved with $|Q_i| \leq 22\sqrt{\Delta}\log^2\Delta + 2$ for each i.

We will partition each X_i into subsets $X_{i1}, X_{i2}, \ldots, X_{ik}$ such that every vertex in V is adjacent to at most one vertex in X_{ij} for any j. Since $\forall v \in V$, $|N[v, X_i]| \leq \ell$, by setting $V' = X_i$ in Lemma 3.4, this can be achieved for $k \leq (\ell-1)\Delta+1$. Now, we construct an auxiliary graph $\widehat{G}$ with vertex set $\widehat{V} = V \cup \{x_{ij} \mid j = 1, \ldots, k\}$ where each x_{ij} corresponds to X_{ij}. The edge set $\widehat{E}$ contains edges from V to $\{x_{ij} \mid 1 \leq j \leq k\}$ where a vertex v is adjacent to x_{ij} if and only if it has a neighbor in X_{ij} in G. The local functions $f_v(\cdot)$ remain the same $\forall v \in V$. Applying Lemma 3.2 to $\widehat{G}$ with $V' = \{x_{ij} \mid 1 \leq j \leq k\}$, since $\forall v \in V$, $|N[v, X_i]| \leq \ell$, we note that there exist at most $22\ell^2\sqrt{\ell}\log k + 2 \leq 23\ell^2\sqrt{\ell}\log(\ell\Delta)$ queries $\widehat{q}$ such that $\forall v \in V$, $\widehat{q}(v) = 0$ and for each $b = 0, 1, ..., |N[v, X_i]|$, there exists a query $\widehat{q}$ such that v is adjacent to exactly b vertices in $\{x_{ij} \mid 1 \leq j \leq k\}$. Let this set of configurations be denoted by $\widehat{Q}_i$.

For each $\widehat{q} \in \widehat{Q}_i$, we construct a query $q \in Q_i$ as follows: $\forall v \in X_j$, $j < i$, we set $q(v) = 1$, and $\forall v \in X_j$, $j > i$, we set $q(v) = 0$. For $v \in X_i$, $q(v) = \widehat{q}(X_{ij})$, where X_{ij} is the set to which v belongs. Suppose in a configuration $\widehat{q}$, $v \in V$ has b neighbors in state 1. We will now show that $\text{score}(v, q) = n_{\leq i-1}(v) + b$. By definition of q, for any $u \in X_i$, $q(u) = 1$ if and only if $\widehat{q}(X_{ij}) = 1$. Since v is adjacent to at most one vertex in X_{ij} for any j, there are exactly b vertices of $N[v, X_i]$ with state 1 in q. Further, recalling that every vertex in set X_j for $j < i$ is in state 1 in q, $\text{score}(v, q) = n_{\leq i-1}(v) + b$. Finally, since for every $b = n_{\leq i-1}(v)+1, \ldots, n_{\leq i}(v)$, there exists a query $\widehat{q} \in \widehat{Q}$ with b neighbors of v in state 1, the proof follows.

Therefore, $|Q| \leq \alpha|Q_i| \leq \frac{2\Delta}{\log\Delta} 23\ell^2\sqrt{\ell}\log(\ell\Delta) < 2500\Delta(\log\Delta)^{2.5}$, since $\ell \leq 4.7\lceil\log(\Delta+1)\rceil$. This completes our proof of Theorem 3.5. □

It should be noted that Theorem 3.5 shows the existence of a small complete query set for SyDSs with symmetric functions. Developing an algorithm that can construct such a query set is left for future work.

4 Existence of Large Incomplete Query Sets

In this section, we show that even for SyDSs where each local function is a threshold function, one can construct exponentially large query sets which are incomplete, that is, they cannot be used to correctly infer all the threshold values. For space reasons, our proof of the following result appears in the appendix.

Theorem 4.1. *For any $n \geq 1$, there is a SyDS $\mathcal{S}$ with $N \geq n$ nodes such that (i) each local function of $\mathcal{S}$ is a deterministic threshold function and (ii) under the batch mode, there is an incomplete query set of size $\Omega(2^N)$ for $\mathcal{S}$.* ■

We now briefly discuss the significance of Theorem 4.1. For a SyDS with N nodes, the number of possible configurations, and hence the maximum number of distinct queries is, 2^N. Theorem 4.1 points out that one may not get a complete query set even if a constant fraction of all the possible configurations is included in the query set. This result provides an indication of the difficulty of finding small complete query sets even when all the local functions are threshold functions. In particular, the theorem suggests that naive random sampling schemes are unlikely to produce complete query sets even after generating a large number of samples. Thus, methods for choosing queries should be carefully designed so that the resulting query set is both complete and concise. In the next section, we present such

a method for probabilistic threshold functions which are more general than deterministic threshold functions.

5 Results for the Probabilistic Threshold Model

Algorithm 1: InferProbThreshold

Data: $G(V, E)$, ϵ, δ, $p_{\min}$, oracle $\mathcal{F}_{\mathrm{PT}}$
Result: Estimates $\hat{p}_v$ and $\hat{t}_v$ for each $v \in V$

/* Phase I: Infer the threshold value for each $v \in V$. */
$\forall v \in V,\ t^L(v) = 0,\ t^H(v) = d(v) + 2;\ V_{\mathrm{rem}} = V;$
while $V_{\mathrm{rem}} \neq \varnothing$ **do**
  $A = V_{\mathrm{rem}};\ \ q = \mathbf{0};$
  while $A \neq \varnothing$ **do**
    $v_{\max} = \arg\max_{v \in A}(t^H(v) - t^L(v));$
    $A = A \setminus N[v_{\max}, G^2];$ /* Remove closed distance-2 neighborhood of $v_{\max}$. */
    In q, set the states of nodes in $N[v_{\max}]$ such that
      $\mathrm{score}(v_{\max}, q) = \lceil (t^H(v_{\max}) + t^L(v_{\max}))/2 \rceil;$
  end
  Let $s_i = \mathcal{F}_{\mathrm{PT}}(q)$ for $i = 1, \ldots, r_t = \left\lceil \frac{1}{p_{\min}} \log(2n/\delta) \right\rceil;$
  $\forall v \in V,$ let $s(v) = \max_{1 \leq i \leq r_t}\{s_i(v)\};$
  for $v \in V_{\mathrm{rem}}$ **do**
    if $s(v) = 1$ *and* $t^H(v) > \mathrm{score}(v, q)$ **then**
      $t^H(v) = \max\big(t^L(v), \mathrm{score}(v, q)\big);$
    else if $s(v) = 0$ *and* $t^L(v) \leq \mathrm{score}(v, q)$ **then**
      $t^L(v) = \min\big(t^H(v), \mathrm{score}(v, q) + 1\big);$
  end
  $\forall v \in V_{\mathrm{rem}}$ such that $t^H(v) = t^L(v)$, remove v from V_{rem}.
end
$\hat{t}(v) = t^L(v);$ /* or $t^H(v)$*/
/* Phase II: Estimate the probability for each $v \in V$. */
$q = \mathbf{1};$
Let $s_i = \mathcal{F}_{\mathrm{PT}}(q)$ for $i = 1, \ldots, r_p = \left\lceil \frac{3}{\epsilon^2 p_{\min}} \log(2n/\delta) \right\rceil;$
$\forall v \in V,\ \hat{p}_v = \left[\sum_{i=1}^{r_p} s_i(v) \right] / r_p;$

In this section, we present our algorithm for inferring probabilistic threshold functions. The inputs to the algorithm are the underlying graph $G(V, E)$, values ϵ and δ that are used to specify the performance guarantees of the algorithm and a value $p_{\min}$ such that each probability value to be estimated is at least $p_{\min}$. The algorithm assumes the availability of an oracle $\mathcal{F}_{\mathrm{PT}}$ which returns a successor of a given query q. The outputs of the algorithm are the estimates for the threshold and probability value for each node.

The algorithm operates in two phases (see Algorithm 1 for details). In Phase I, it uses the adaptive query mode along with a binary search procedure to successively reduce the range of the threshold value for each node. Since the system is stochastic, queries are repeated an appropriate number of times to meet the required performance guarantees. In Phase II,

estimates for the probability values of nodes are obtained by simply repeating the query in which every entry is 1 and computing the number of times a node's state changes to 1 in the successor. We now establish the performance guarantees provided by the algorithm.

Theorem 5.1. *Let $G(V, E)$ be the underlying graph of a SyDS where each local function is a probabilistic threshold function. Let $\mathcal{F}_{\mathrm{PT}}$ be the oracle corresponding to the probabilistic threshold model defined over G with probability $p(v)$ and threshold $t(v)$ for each vertex v. Let $|\mathcal{Q}|$ be the number of distinct queries generated by Algorithm 1 for the particular case of $p(v) = 1,\ \forall v \in V$. Let $\epsilon, \delta, p_{\min} \in (0, 1)$ be given values. Algorithm 1 infers the probabilistic threshold model corresponding to $\mathcal{F}_{\mathrm{PT}}$ using at most $\frac{1}{p_{\min}} \log\left(\frac{2n}{\delta}\right)\left(|\mathcal{Q}| + \frac{3}{\epsilon^2}\right)$ queries with the following guarantees: With probability at least $1 - \delta$, (i) thresholds of all vertices are determined and (ii) for every vertex v, the probability $p(v)$ is estimated within a factor of $1 \pm \epsilon$, provided $p(v) \geq p_{\min}$.*

Proof. Since $\mathcal{Q}$ is a complete query set when $p(v) = 1$ (i.e., for the deterministic threshold case), we note that for every $\big(v, t(v)\big)$, there exists $q \in \mathcal{Q}$ such that $\mathrm{score}(v, q) = t(v)$. For r_t repetitions of this query, the probability that the estimated threshold is not equal to $t(v)$ is $(1 - p(v))^{r_t}$, the probability that the state of v is 0 for every repetition of query q. Let $\hat{t}_v$ be the inferred threshold. By union bound,

$$\begin{aligned}\Pr\big(\exists v \in V \text{ such that } \hat{t}_v \neq t(v)\big) &\leq \sum_{v \in V} \Pr\big(\hat{t}_v = t(v)\big) = \sum_{v \in V} (1 - p(v))^{r_t} \\ &\leq n(1 - p_{\min})^{r_t} \leq \frac{\delta}{2} \end{aligned} \tag{2}$$

for $r_t \geq \frac{1}{p_{\min}} \log(2n/\delta)$.

Let $x_v = \sum_{i=1}^{r_p} s_i(v)$. Recall that $\hat{p}_v = \frac{x_v}{r_p}$. Since x_v is a sum of r_p i.i.d. Bernoulli random variables with $\mathbb{E}[x_v] = r_p p(v)$, we can apply Chernoff bounds (parts (4a) and (4c) of Theorem A.1 in the appendix) to get

$$\Pr\big(|\hat{p}_v - p(v)| \geq \epsilon p(v)\big) = \Pr\big(|x_v - r_p p(v)| \geq \epsilon r_p p(v)\big) \leq 2e^{-\frac{\epsilon^2 r_p p(v)}{3}} \leq 2e^{-\frac{\epsilon^2 r_p p_{\min}}{3}}.$$

Again using the union bound,

$$\Pr\big(\exists v \in V \text{ such that } |\hat{p}_v - p(v)| \geq \epsilon p(v)\big) \leq 2ne^{-\frac{\epsilon^2 r_p p_{\min}}{3}} \leq \frac{\delta}{2} \tag{3}$$

for $r_p \geq \frac{3}{\epsilon^2 p_{\min}} \log(2n/\delta)$. Combining (2) and (3), we have the guarantees specified in the theorem. □

Table 1: Networks used in our experiments and their properties [19].

Network	Type	Num. Nodes	Avg. Deg.	Max. Deg.
ca-condmat	co-author	21,363	8.55	279
ca-grqc	co-author	4,158	6.46	81
ca-hepth	co-author	8,638	5.74	65

6 Experimental Results

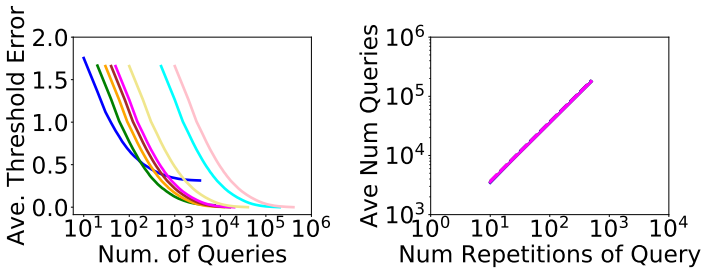
The purpose of the experiments is to evaluate Algorithm 1 using a set of social networks. Here, we focus on estimated threshold results, and leave probability estimation for an expanded version of the paper. The quality of the probability estimations is comparable to that of threshold estimates.

Table 2: Summary of the parameters and their values used in the analyses.

Parameter	**Description**
Networks.	Three networks in Table 1.
Threshold model.	The Probabilistic Threshold Functions (PTFs) of Section 2.2.
Threshold fraction, t_f.	The threshold fraction $0 \leq t_f \leq 1$ (taken as 0.5 and 1 here) determines the range over which true thresholds t_v are determined for nodes v, and the range over which thresholds are evaluated in estimating thresholds $\hat{t}_v$ according to Algorithm 1. We have $t_m = (d_v + 2)/2$ and $t_r = t_f(d_v + 2)/2$, such that the threshold range is given by the interval $[t_m - t_r, t_m + t_r]$.
Threshold instance, t_i.	For each network, there are five independent threshold assignments made to nodes for each of $t_f = 0.5$ and 1, called true thresholds. Values for instances are labeled 1 through 5.
Probabilities, $p \equiv p_h$.	For PTFs, when a node's threshold is met, with probability p, a node will transition from state 0 to 1; with $1 - p$, a node will not transition, even though its threshold is met. We examine values of $p = 0.25$, 0.5, 0.75, and 1. Values are uniform or homogeneous for all nodes of a network in each analysis.
Number r of repetitions of the same query.	The number of repetitions of the query, to obtain successor configurations, is to account for the stochasticity in the local function. The number of repetitions are 10, 20, 30, 40, 50, 100, 500, and 1000. This is an input parameter, versus r_t and r_p being computed in Algorithm 1.
Estimated solutions.	For each assignment of true thresholds and probabilities, ten estimated solutions are computed according to Algorithm 1. Thus, many results below are the average of 50 values (5 true instances $\times$ 10 estimated solutions per true instance).

6.1 Experimental Procedures and Parameters

Table 1 lists the networks studied. All are co-authorship networks. Experimental parameters are summarized in Table 2. For each network, we produce two sets of five true threshold assignments. Let t_f be a real value in the range $[0, 1]$, called the threshold fraction, let $t_m = (d_v + 2)/2$ be the mean threshold, and let $t_r = t_f(d_v + 2)/2$ be the threshold amplitude. Then when $t_f = 0.5$ (resp., $t_f = 1$), the threshold range for v, from which t_v is selected uniformly at random, is $[(d_v + 2)/4, 3(d_v + 2)/4]$ (resp., $[0, (d_v + 2)]$). In this way, the threshold ranges are always centered about t_m for each v. The threshold limits 0 and $d_v + 2$ are selected because the former threshold will always be satisfied and the latter never will. We generate five true threshold assignments (instances t_i), per network, for each of $t_f = 0.5$ and 1.0. Therefore, we have 10 different threshold assignments for each network.



(a) (b)

Figure 1: Evolution in progress toward solutions for estimated node thresholds as a function of number of queries for ca-condmat. (a) $t_f = 1$ and $p_h = 0.25$. Curves, in moving from left to right, correspond to $r = 10$, 20, 30, 40, 50, 100, 500, and 1000. As r increases, more queries are executed. (b) The average number of queries to produce small e_t does not decrease with increasing number of repetitions. Curves are coincident for all combinations of (t_f, p_h, r). The take-away is that while increasing r in the range 20 to 100 does improve (reduce) e_t, the effect is not marked. See text.

Node probabilities p are set uniformly for all nodes in a network. We examine uniform or homogeneous probabilities $p = 0.25$, 0.5, 0.75, and 1. We often write the probability as p_h to emphasize the probabilities are homogeneously assigned to all nodes of a network. The purpose of Algorithm 1 is to estimate the true thresholds and probabilities of the node functions. Henceforth, we use PTF as the abbreviation for the phrase "probabilistic threshold function".

Finally, for Algorithm 1, when $p_h < 1$, we submit a query repeatedly to assess stochasticity. The numbers of repetitions r examined are 10, 20, 30, 40, 50, 100, 500, and 1000. We note that these repetitions are counted in the number of queries submitted. For example, when $r = 30$, for each query formed, this counts as 30 queries submitted whose successor states are returned. For $p_h = 1$, only $r = 1$ repetition is required. For each set of true values (thresholds and probabilities), we compute ten estimated solutions to account for stochastic effects; typically, we average these results in Section 6.2 below.

6.2 Evaluation of True Threshold Assignments and Estimated Thresholds

Four different types of experimental results are presented for the three networks. Our findings are confined to the parameters of this study.

Effect of number of queries and number of repetitions on errors in estimated thresholds. Figure 1a provides average threshold error as a function of the number of queries for ca-condmat. Average threshold error is given by $e_t = (1/n)\ \Sigma_{v \in V} \left| [t_v - \hat{t}_v] \right|$. Each curve is data from one analysis, for a particular r value, and captures the dynamic nature of reductions in e_t as Algorithm 1 executes. Curves, in moving left to right, correspond to increasing r, from 10 to 1000. Final e_t values are approached asymptotically for each r.

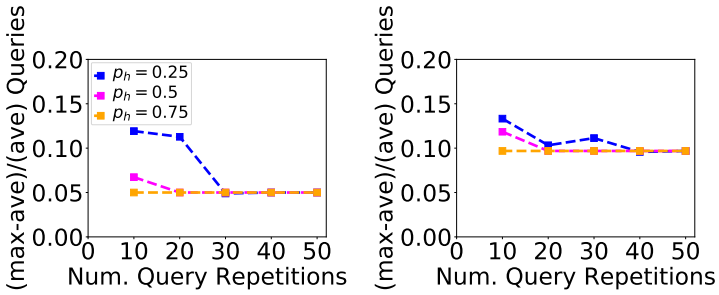
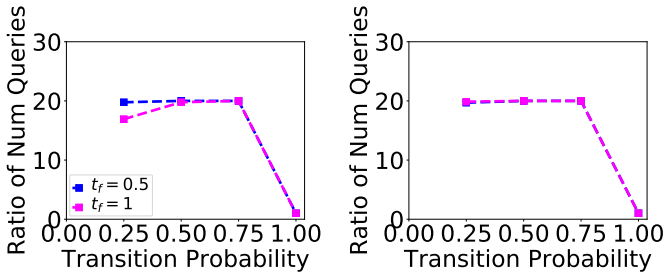


Figure 2: Reductions in the ranges of numbers of queries to compute estimated threshold solutions. Data are shown for (a) ca-grqc and $t_f = 0.5$, and (b) ca-hepth and $t_f = 1$. For these data, the differences between maximum and average are greatest for $p_h = 0.25$ and least $p_h = 0.75$ at $r = 10$. However, as r increases to 50, the maximum number of queries, over 50 instances, decreases toward the average number of queries (but not monotonically for ca-hepth).

Limitations in the effectiveness of query repetitions. Figure 1b shows final results from the data in Figure 1a, and other data, for ca-condmat, but now the average number of queries to obtain small threshold errors is shown as a function of r, for all combinations of t_f, p_h, and r of this study. The major result is that driving up the number r of repetitions does not bring major benefits: the total number of queries to produce small e_t does not decrease for large r. (One might hypothesize that increasing r will enable better estimates of $t(v)$ and $p(v)$ with a fewer overall number of queries.) While there is benefit in increasing r, because e_t may reduce from 0.01 to 0.0001 with increasing r in the range 20 to 100, it is not a major effect in practical terms. Rather, r serves as a multiplier for each unique query formed, which increases the cost of estimating thresholds (in terms of total numbers of queries). Note that each data point is the average of 50 values from Table 2.

Reduction in the variance of the numbers of queries to achieve solutions, and its limits. Figure 2 provides two plots, for ca-grqc and ca-hepth, to evaluate how the maximum number of queries reduces toward the average number of queries, as r increases. Note that there is a permanent difference between the maximum and average that does not vanish as r increases up to 1000: 5% for ca-grqc and 10% for ca-hepth. These data show that the biggest range in numbers of queries is for $p_h = 0.25$ and least for $p_h = 0.75$.



(a) (b)

Figure 3: Effect of stochastic transitions on the number of queries to estimate thresholds for (a) ca-grqc and (b) ca-condmat. Ratio of number of queries with transition probability $p_h < 1$, normalized by the number of queries when the transition probability $p_h = 1$. To be conservative, we assume that the $r = 20$ results give sufficiently accurate e_t values. These data indicate that when $p_h < 1$, so that the model is stochastic, a factor of roughly 20 on the number of queries is required to obtain good threshold estimates, compared to the number of queries required in the deterministic case ($p_h = 1$).

Comparisons in numbers of queries to estimate thresholds for stochastic PTFs and deterministic threshold functions. Comparisons are made between stochastic PTFs where $p_h < 1$ and the analogous deterministic model where $p_h = 1$. Results are shown in Figure 3 for ca-grqc and ca-condmat, for $t_f = 0.5$ and 1 (although the results for different t_f essentially overlay). The number of queries for $p_h = 1$ (see [2]) are used to normalize the results at all p_h values to obtain the multiplier in the number of queries required to estimate thresholds for stochastic models. From these plots, the multiplier is about 20. Note that if we required exact solutions, which correspond to $r \approx 100$, then this multiplier would increase to about 100; in this sense, these results are conservative.

7 Future Research Directions

We considered the problem of inferring users' choice functions in networked systems. Using the synchronous dynamical system formalism for a networked system, we modeled users' choice functions by threshold and probabilistic threshold functions. There are several directions for future work. One direction is to further improve the bounds on the size of complete query sets for symmetric functions. Another direction is to consider other classes of local functions to capture users' choices. A limitation of our work is the assumption that the entire system is observable; that is, queries and responses specify the states of all the nodes in the system. It is of interest to investigate techniques that can overcome this limitation.

Acknowledgments: We thank the referees of COMSOC 2018 for providing valuable suggestions. We also thank the computer systems administrators and managers at the Biocomplexity Institute for their help in this and many other works: Dominik Borkowski, William Miles Gentry, Jeremy Johnson, William Marmagas, Douglas McMaster, Kevin Shinpaugh and Robert Wills. This work has been partially supported by DARPA Cooperative Agreement D17AC00003 (NGS2), DTRA CNIMS (Contract HDTRA1-11-D-0016-0001), NSF DIBBS Grant ACI-1443054 and NSF BIG DATA Grant IIS-1633028. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

References

[1] A. Adiga, C. J. Kuhlman, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. Inferring local transition functions of discrete dynamical systems from observations of system behavior. *Theor. CS.*, 679:126–144, 2017.

[2] A. Adiga, C. J. Kuhlman, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. Learning the behavior of a dynamical system via a '20 questions' approach. Proc. AAAI (to appear), 2018.

[3] R. Axelrod. *The Complexity of Cooperation.* Princeton University Press, Princeton, NJ, 1994.

[4] C. Barrett, H. B. Hunt III, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. Modeling and analyzing social network dynamics using stochastic discrete graphical dynamical systems. *Theoretical Computer Science*, 412(30):3932–3946, 2011.

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

[6] X. Bei, W. Chen, J. Garg, M. Hoefer, and X. Sun. Learning market parameters using aggregate demand queries. In *Proc. AAAI*, pages 411–417, 2016.

[7] G. Berry and C. J. Cameron. A new method to reduce overestimation of thresholds with observational network data. arXiv:1702.02700v1 [cs.SI], Feb. 2017.

[8] D. Centola. The spread of behavior in an online social network experiment. *Science*, 329:1194–1197, 2010.

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

[10] J. Epstein and R. Axtell. *Growing Artificial Societies: Social Science from the Bottom Up.* Brookings and MIT Press, Cambridge, MA, 1996.

[11] Z. Füredi and J. Kahn. On the dimensions of ordered sets of bounded degree. *Order*, 3:15–20, 1986.

[12] S. González-Bailón, J. Borge-Holthoefer, A. Rivero, and Y. Moreno. The dynamics of protest recruitment through an online network. *Scientific Reports*, 1:7 pages, 2011.

[13] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan. Learning influence probabilities in social networks. In *Proc. ACM Intl. Conf. on Web Search and Data Mining (WSDM 2010)*, pages 241–250, 2010.

[14] M. Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, pages 1420–1443, 1978.

[15] E. Kazemi, L. Chen, S. Dasgupta, and A. Karbasi. Comparison based learning from weak oracles. Arxiv:1802.06942v1 [cs.LG], 2018.

[16] M. J. Kearns and V. V. Vazirani. *An Introduction to Computational Learning Theory.* MIT Press, Cambridge, MA, 1994.

[17] J. Kleinberg, S. Mullainathan, and J. Ugander. Comparison-based choices. arXiv:1705.05735v1 [cs.DS], May 2017.

[18] C. J. Kuhlman, V. S. Anil Kumar, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, S. Swarup, and G. Tuli. A Bithreshold Model of Complex Contagion and its Application to the Spread of Smoking Behavior. In *Proc. SNA-KDD Workshop*, pages 18.1–18.10, 2011.

[19] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[20] M. Macy and R. Willer. From factors to actors: Computational sociology and agent-based modeling. *Annual Reviews in Sociology*, 28:143–166, 2002.

[21] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, 2005.

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

[23] K. P. Murphy. Passively learning finite automata. Technical Report 96-04-017, Santa Fe Institute, Santa Fe, NM, 1996.

[24] D. Romero, B. Meeder, and J. Kleinberg. Differences in the mechanics of information diffusion across topics: Idioms, political hashtags, and complex contagion on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 695–704. ACM, 2011.

[25] S. B. Rosenthal, C. R. Twomey, A. T. Hartnett, H. S. Wu, and I. D. Couzin. Revealing the hidden networks of interaction in mobile animal groups allows prediction of complex behavioral contagion. *Proceedings of the National Academy of Sciences*, 112(15):4690–4695, 2015.

[26] K. Saito, R. Nakano, and M. Kimura. Prediction of information diffusion probabilities for Independent Cascade model. In *Proc. Knowledge-Based Intelligent Information and Engineering Sytems (KES 2008)*, pages 67–75, 2008.

[27] G. Tuli, M. V. Marathe, S. S. Ravi, and S. Swarup. Addiction dynamics may explain the slow decline of smoking prevalence. In *SBP*, volume 7227 of *Lecture Notes in Computer Science*, pages 114–122. Springer, 2012.

[28] J. Urschel, V. Brunel, A. Moitra, and P. Rigollet. Learning determinantal point processes with moments and cycles. In *Proc. 34th ICML*, pages 3511–3520, 2017.

[29] D. J. Watts. A simple model of global cascades on random networks. *Proceedings of the National Academy of Sciences*, 99:5766–5771, 2002.

[30] Y. Zhou, J. Li, and J. Zhu. Identify the Nash equilibrium in static games with random payoffs. In *Proc. 34th ICML*, pages 4160–4169, 2017.

Abhijin Adiga
Network Dynamics and Simulation Science Laboratory
Biocomplexity Institute of Virginia Tech
Blacksburg, VA 24061, USA
Email: abhijin@vt.edu

Chris J. Kuhlman
Network Dynamics and Simulation Science Laboratory
Biocomplexity Institute of Virginia Tech
Blacksburg, VA 24061, USA
Email: ckuhlman@vt.edu

Madhav V. Marathe
Network Dynamics and Simulation Science Laboratory
Biocomplexity Institute of Virginia Tech and
Computer Science Department, Virginia Tech
Blacksburg, VA 24061, USA
Email: mmarathe@vt.edu

S. S. Ravi
Network Dynamics and Simulation Science Laboratory
Biocomplexity Institute of Virginia Tech
Blacksburg, VA 24061, USA and
Department of Computer Science
University at Albany – State University of New York
Albany, NY 12222
Email: ssravi0@gmail.com

Daniel J. Rosenkrantz
Department of Computer Science
University at Albany – State University of New York
Albany, NY 12222
Email: drosenkrantz@gmail.com

Richard E. Stearns
Department of Computer Science
University at Albany – State University of New York
Albany, NY 12222
Email: thestearns2@gmail.com

A Appendix

A.1 An Example of a Synchronous Dynamical System (SyDS)

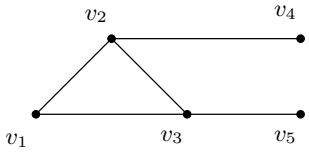


Figure 4: The underlying graph of a SyDS

Example: Consider the graph of a SyDS shown in Figure 4. Assume that initially, v_1, v_4 and v_5 are in state 1 and all other nodes are in state 0. Thus, the initial configuration $\mathcal{C}_0$ of the system is $(1, 0, 0, 1, 1)$.

We first consider the case when each local function is a threshold function (i.e., the system is deterministic). Suppose the local transition functions at each of the nodes v_1, v_2 and v_3 is the 2-threshold function and those at v_4 and v_5 are 1-threshold functions. During the first time step, the states of nodes v_2 and v_3 change to 1 since each of them has a score of 2 and their local functions are 2-threshold functions. The states of v_4 and v_5 remain at 1 (since each of them has a score of 1 and their local functions are 1-threshold functions). The state of v_1 changes to 0 since its threshold is 2 and its score is 1. Thus, the configuration $\mathcal{C}_1$ of the system at time 1 is $(0, 1, 1, 1, 1)$. In the next time step, it can be seen that v_1 changes to 1 while the other nodes remain at 1. Thus, the configuration $\mathcal{C}_2$ of the system at time 2 is $(1, 1, 1, 1, 1)$. The system remains in this configuration forever; that is, the configuration $(1, 1, 1, 1, 1)$ reached at time 2 is a **fixed point** for the system.

Now, suppose the local functions are probabilistic threshold functions. Let the local transition functions at each of the nodes v_1, v_2 and v_3 be the 2-threshold function with probability 0.75 and those at v_4 and v_5 be 1-threshold functions with probability 0.9. Further, let the initial configuration $\mathcal{C}_0$ be $(1, 0, 0, 1, 1)$ as before. At time 1, the state of v_1 changes to 0 (since its score is 1 but its threshold is 2). For each of the nodes v_2 through v_5, even though the score is at least as large as its threshold, there is a non-zero probability that their states will change to 0 in time step 1. Thus, the system may reach the configuration $(0, 0, 0, 0, 0)$ at time step 1 with probability $1/16$. This configuration is a fixed point for the stochastic system.

A.2 Chernoff bounds

Some of our results rely on the following theorem for independent binary random variables. For a proof of this theorem, the reader is referred to [21].

Theorem A.1 (Chernoff bounds)**.** *Suppose $X_1, \ldots, X_n$ are independent random binary variables, X denotes their sum, and $\mu = \mathbb{E}[X]$. Then*

$$\Pr[X \geq (1+\beta)\mu] \leq e^{-\beta^2\mu/3}, \quad 0 < \beta < 1, \tag{4a}$$

$$\Pr[X \geq (1+\beta)\mu] \leq e^{-\beta\mu/3}, \quad 1 < \beta, \tag{4b}$$

$$\Pr[X \leq (1-\beta)\mu] \leq e^{-\beta^2\mu/2}, \quad 0 < \beta < 1. \tag{4c}$$

A.3 Proof of Lemma 3.1

We first establish three claims which are useful in proving Lemma 3.1.

Claim A.2. $\left(1 + \frac{1}{b}\right)^b$ *is monotone increasing in b for positive integers.*

Proof: Consider the collection of $(b+1)$ numbers $\left(1, \frac{b+1}{b}, \ldots, \frac{b+1}{b}\right)$. Using the fact that

their arithmetic mean is $\geq$ their geometric mean, we have

$$\frac{1+b\left(\frac{b+1}{b}\right)}{b+1} \geq \left(1^1\left(\frac{b+1}{b}\right)^b\right)^{\frac{1}{b+1}}$$

Hence,

$$\frac{(b+1)+1}{b+1} \geq \left(\frac{b+1}{b}\right)^{\frac{b}{b+1}}.$$

■

Claim A.3. $\binom{d}{b}\left(\frac{b}{d}\right)^b\left(1-\frac{b}{d}\right)^{d-b} \geq \frac{1}{\sqrt{2(d+1)}}$.

Proof: Let $h(b,d) = \binom{d}{b}\left(\frac{b}{d}\right)^b\left(1-\frac{b}{d}\right)^{d-b}$. We will first show that for $b \leq \left\lfloor\frac{d}{2}\right\rfloor - 1$, $h(b+1,d) < h(b,d)$ and for $b \geq \left\lfloor\frac{d}{2}\right\rfloor$, $h(b+1,d) \geq h(b,d)$, and hence, $h(\cdot)$ attains a minimum value at $b = \left\lfloor\frac{d}{2}\right\rfloor$.

$$\begin{aligned}\frac{h(b+1,d)}{h(b,d)} &= \frac{\binom{d}{b+1}}{\binom{d}{b}}\frac{\left(\frac{b+1}{d}\right)^{b+1}}{\left(\frac{b}{d}\right)^b}\frac{\left(1-\frac{b+1}{d}\right)^{d-b-1}}{\left(1-\frac{b}{d}\right)^{d-b}} \\ &= \frac{d-b}{b+1}\left(\frac{b+1}{b}\right)^b\frac{b+1}{d}\left(\frac{d-b-1}{d-b}\right)^{d-b}\frac{d}{d-b-1} \\ &= \left(\frac{b+1}{b}\right)^b\left(\frac{d-b-1}{d-b}\right)^{d-b-1} = \left(\frac{b+1}{b}\right)^b\left(\frac{b'}{b'+1}\right)^{b'},\end{aligned}$$

where, $b' = d-b-1$. When $b \leq \left\lfloor\frac{d}{2}\right\rfloor - 1$, $b' > b$ and when $b \geq \left\lfloor\frac{d-1}{2}\right\rfloor + 1$, $b' \leq b$. The rest follows by applying Claim A.2. When b is even, it is well-known that $h\left(\frac{d}{2},d\right) \geq \frac{1}{\sqrt{2d}}$ (using a lower bound on the central binomial coefficient). Now we will show that when b is odd, $h\left(\frac{d-1}{2},d\right) \geq \frac{1}{\sqrt{2(d+1)}}$. Let $b = 2k+1$.

$$\begin{aligned}\frac{h\big(k,2k+1\big)}{h\Big(k,2k+2\Big)} &= \frac{\binom{2k+1}{k}}{\binom{2k+2}{k}}\frac{\left(\frac{k}{2k+1}\right)^k}{\left(\frac{k}{2k+2}\right)^k}\frac{\left(1-\frac{k}{2k+1}\right)^{k+1}}{\left(1-\frac{k}{2k+2}\right)^{k+2}} \\ &= \frac{k+2}{2k+2}\left(\frac{2k+2}{2k+1}\right)^k\left(\frac{k+1}{k+2}\right)^{k+1}\left(\frac{2k+2}{2k+1}\right)^{k+1}\frac{2k+2}{k+2} \\ &= \left(1+\frac{1}{2k+1}\right)^{2k+1}\left(1+\frac{1}{k+1}\right)^{-(k+1)} > 1\,.\end{aligned}$$

The inequality follows from Claim A.2. Therefore, when d is odd,

$$h\left(\tfrac{d-1}{2},d\right) \;>\; h\left(\tfrac{d-1}{2},d+1\right) \;\geq\; h\left(\tfrac{d+1}{2},d+1\right) \;\geq\; \frac{1}{\sqrt{2(d+1)}}.$$

This completes the proof of Claim A.3. ■

Claim A.4. *For any positive $x \leq \frac{1}{2}$, $1-x \geq e^{-2x}$.*

Proof: $e^{2x}(1-x) > (1+2x)(1-x) = 1 + x(1-2x) \geq 1$. ■

Statement of Lemma 3.1: Let b, d and D be positive integers such that $b \leq d \leq D$ and let $z = \langle \frac{bD}{d} \rangle$. Then, $\binom{d}{b}\left(\frac{z}{D}\right)^b\left(1-\frac{z}{D}\right)^{d-b} \geq \frac{1}{11\sqrt{d+1}}$.

Proof of Lemma 3.1: We have two cases to consider: (a) $z \leq \frac{bD}{d}$ and (b) $z > \frac{bD}{d}$. But first we note that by definition, $\left|z - \frac{bD}{d}\right| \leq \frac{1}{2}$.

Case (a): $\frac{bD}{d} - \frac{1}{2} \leq z \leq \frac{bD}{d}$.

$$\begin{aligned}
\binom{d}{b}\left(\frac{z}{D}\right)^b\left(1-\frac{z}{D}\right)^{d-b} &\geq \binom{d}{b}\left(\frac{z}{D}\right)^b\left(1-\frac{b}{d}\right)^{d-b} \\
&\geq \binom{d}{b}\left(\frac{b}{d}-\frac{1}{2D}\right)^b\left(1-\frac{b}{d}\right)^{d-b} \\
&\geq \binom{d}{b}\left(\frac{b}{d}\right)^b\left(1-\frac{b}{d}\right)^{d-b}\left(1-\frac{d}{2bD}\right)^b \\
&\geq \frac{1}{2\sqrt{d}}\left(1-\frac{d}{2bD}\right)^b \geq \frac{1}{e^2\sqrt{2(d+1)}} \geq \frac{1}{11\sqrt{d+1}}\,.
\end{aligned}$$

The last but one inequality above follows from Claim A.4 and the trivial bound $d/D \leq 1$.

Case (b): $\frac{bD}{d} \leq z \leq \frac{bD}{d} + \frac{1}{2}$.

$$\begin{aligned}
\binom{d}{b}\left(\frac{z}{D}\right)^b\left(1-\frac{z}{D}\right)^{d-b} &\geq \binom{d}{b}\left(\frac{b}{d}\right)^b\left(1-\frac{z}{D}\right)^{d-b} \\
&\geq \binom{d}{b}\left(\frac{b}{d}\right)^b\left(1-\frac{b}{d}-\frac{1}{2D}\right)^{d-b} \\
&\geq \binom{d}{b}\left(\frac{b}{d}\right)^b\left(1-\frac{b}{d}\right)^{d-b}\left(1-\frac{\frac{1}{2D}}{1-\frac{b}{d}}\right)^{d-b} \\
&\geq \frac{1}{\sqrt{2(d+1)}}\left(1-\frac{d}{2(d-b)D}\right)^{d-b} > \frac{1}{11\sqrt{d+1}}\,.
\end{aligned}$$

This completes the proof of Lemma 3.1. ■

A.4 Proof of Theorem 4.1

Statement of Theorem 4.1: For any $n \geq 1$, there is a SyDS $\mathcal{S}$ with $N \geq n$ nodes such that (i) each local function of $\mathcal{S}$ is a deterministic threshold function and (ii) under the batch mode, there is an incomplete query set of size $\Omega(2^N)$ for $\mathcal{S}$.

Proof: We show that the result holds even for SyDSs whose underlying graphs are complete binary trees. We construct such a SyDS $\mathcal{S}$ as follows. Given an integer $n \geq 1$, choose the smallest integer $N \geq n$ such that $N = 2^k - 1$ for some positive integer $k \geq 3$. Let the underlying graph $G(V, E)$ of $\mathcal{S}$ be the complete binary tree with k levels. To specify the queries, assume that the root is named v_1, the two children of the root are named v_2 and v_3 respectively, and the remaining $N - 3$ nodes are labeled arbitrarily using the labels v_4 through v_N. The local function at each node is a threshold function, where the threshold

value is strictly positive. However, different nodes may have different threshold values and the goal is to infer the threshold values of all the nodes.

For a query q, let $q(i)$ denote the value specified by q for node v_i, $1 \leq i \leq N$. (Note that $q(i) \in \{0, 1\}$, $1 \leq i \leq N$.) Consider the query set Q defined by

$$Q \ = \ \{q \ : \ q(i) = 0, \ i = 1, 2, 3\}.$$

Thus, Q contains each query q such that the first three entries of q are all 0. Hence $|Q| = 2^{N-3} = \Omega(2^N)$. The root node v_1 has a degree of 2 and has a positive threshold; that is, its threshold value can be any integer in the range 1 through $d(v_1) + 2 = 4$. If a query set is complete, then for each $j \in \{0, 1, 2, 3\}$, it must contain a query q such that score(v_1, q) $= j$. However, from the way Q is constructed, it can be seen that for every query $q \in Q$, score$(v_1, q) = 0$. Thus, from the responses to the queries in Q, one cannot correctly identify the threshold of the root node v_1. In other words, even though $|Q| = \Omega(2^N)$, Q is not a complete query set under the batch mode. ■