The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Learning the Behavior of a Dynamical System Via a "20 Questions" Approach

Abhijin Adiga,[1] **Chris J. Kuhlman,**[1] **Madhav V. Marathe,**[1]
S. S. Ravi,[1,2] **Daniel J. Rosenkrantz,**[2] **Richard E. Stearns**[2]
[1]Biocomplexity Institute of Virginia Tech, Blacksburg, VA 24061 [2]University at Albany – SUNY, Albany, NY 12222
{abhijin, ckuhlman, mmarathe, ssravi}@vt.edu, drosenkrantz@gmail.com, thestearns2@gmail.com

Abstract

Using a graphical discrete dynamical system to model a networked social system, the problem of inferring the behavior of the system can be formulated as the problem of learning the local functions of the dynamical system. We investigate the problem assuming an active form of interaction with the system through queries. We consider two classes of local functions (namely, symmetric and threshold functions) and two interaction modes, namely batch mode (where all the queries must be submitted together) and adaptive mode (where the set of queries submitted at a stage may rely on the answers received to previous queries). We develop complexity results and efficient heuristics that produce query sets under both query modes. We demonstrate the performance of our heuristics through experiments on over 20 well-known networks.

1 Introduction

Background and Motivation. Graphical discrete dynamical systems are used in a variety of settings to understand population-level contagion dynamics in terms of individual (human) agent behavior. Examples include the spread of health behaviors (Valente 2010) such as drug overdose (Sherman et al. 2009); obesity (Christakis and Fowler 2007); segregation (Schelling 1971); becoming a user of an online communications tool (Karsai et al. 2014); coordination (Rosenthal et al. 2015); and financial contagions (Gai and Kapadia 2010). The frameworks in these works and in ours here are network representations of populations, where nodes and edges represent entities such as humans and pairwise interactions, respectively. Each of the cited works can be viewed as capturing influence through **threshold models** (Granovetter 1978; Schelling 1978), where a node v_i contracts a contagion if at least a specified number of its neighbors has already contracted it. This number for v_i is called its **threshold** t_i. We are interested in **complex contagions** (Centola and Macy 2007) that are characteristic of social contagions, where $t_i \geq 1$; i.e., agents need multiple reinforcing interactions to adopt a contagion. (Watts 2002) argues that threshold models are useful in a host of settings where incomplete information exists or when there is insufficient time to make more deliberate decisions.

In particular, small changes in the thresholds of nodes can make a large difference in population dynamics. An example is provided in (Granovetter 1978), where a change in one node's threshold by a value of 1 in a large graph, changes population-level collective action from non-existent to full collective action. Several works have used mined data to infer thresholds for applications ranging from protests, to Twitter messaging, to joining social media (González-Bailón et al. 2011; Romero, Meeder, and Kleinberg 2011; Ugander et al. 2012); see also (Easley and Kleinberg 2010). Importantly, in all of these cases, *heterogeneous* (i.e., non-uniform) thresholds among agents have been inferred. *Thus, node thresholds must be determined based on a node's individual behavior, its (local) neighborhood structure, and behaviors (and states) of its neighbors.* Symmetric functions, which generalize threshold functions, arise in game theoretic settings (Papadimitriou and Roughgarden 2003).

Some works have studied threshold inference in a **passive** setting (e.g., (Adiga et al. 2017a)) where observations are *given* and the problem is to infer thresholds from those observations. In this work, we study the case where an algorithm has *control* over what information it extracts from the system via **querying** the system for desired information. In particular, the algorithm gives a set of configurations (or queries) to the system and infers properties based on the system's outputs. We study two query modes, namely **batch** and **adaptive** modes, that differ in their degrees of control. Under the batch mode, all the queries must be submitted together. In the adaptive mode, queries can be submitted in several stages, and queries at a stage can depend on the answers to previous queries, a strategy similar to that used in games such as "Twenty questions"[1].

We relate inference problems to well-studied graph theoretic problems such as coloring, using combinatorial and algorithmic perspectives. The formulation also enables us to quantify rigorously the complexity of inferring such systems. Motivation for these problems comes in part from the DARPA Next Generation Social Science (NGS2) Program, where experimental data on social networks are used to infer properties of predictive models. Our work is similar in spirit—but quite different in problem domain and results—to some of the recent works on inference (e.g. (Kleinberg,

[1]See the Wikipedia entry on this game.

Mullainathan, and Ugander 2017)).

Summary of Results. Our focus is on the following problem: *Given the underlying graph of a graphical discrete dynamical system, construct queries to identify all the local functions.* The optimization goal is to minimize the number of queries. Our results are summarized below. Due to space limitations, detailed proofs can be found in (Adiga et al. 2017b; 2017c).

1. We develop algorithms for generating query sets under both batch and adaptive modes to identify local functions of dynamical systems. As can be expected, adaptive query mode can produce significantly smaller query sets compared to the batch mode. We also show that if the goal is to find a query set which can identify symmetric functions with high probability, the size of the query set can be further reduced.
2. We prove lower bounds on the number of queries needed under both batch and query modes. We also present complexity results that point out the difficulty of efficiently generating small query sets.
3. We present an approximation algorithm with a provable performance guarantee to make a query set compact by eliminating redundant queries.
4. We evaluate our algorithms on a large number of real-world and synthetic networks. For the batch mode, one of our approaches based on greedy graph coloring generated query sets of minimum size for most of the real-world networks. We also demonstrate the effectiveness of a simple approach based on sampling queries from a particular distribution followed by a compaction algorithm.
5. We develop a greedy adaptive heuristic based on binary search and evaluate it by generating query sets for various settings of networks and threshold assignments. Our results show that for most cases, it significantly outperforms the batch mode algorithms.

Related Work. There are several works on the passive mode of inference. Many researchers have studied the problem of learning automata (e.g., (Murphy 1996)). In (Kearns and Vazirani 1994), the problems of learning normal forms and Boolean functions are discussed. Works such as (González-Bailón et al. 2011; Romero, Meeder, and Kleinberg 2011) present methods to infer thresholds from social media data. Learning the source nodes of infection for contagions is addressed in (Zhu, Chen, and Ying 2017). Many of these problems are formally hard even for simple local functions. The work of (Adiga et al. 2017a) studies several problems aimed at inferring thresholds in discrete dynamical systems.

Active querying is studied in (Kleinberg, Mullainathan, and Ugander 2017) in the context of determining user choices from a finite set of ranked options—the choice set problem. The goal is to minimize the number of queries of arbitrary subsets S of size k, of a universal set U, to learn a user's choice from among the elements of each set S. With these results, the algorithm can then predict the user's choice for any subset $S \subseteq U$ of size k. They show that this can be accomplished with $O(n \log n)$ queries where $n = |U|$.

Although error-tolerant approaches for querying systems are beyond the scope of this work, several authors study inference problems wherein errors are allowed. These include (Valiant 1984; Juba 2016; He et al. 2016; Zhang, Mathew, and Juba 2017; Kleinberg, Mullainathan, and Ugander 2017).

While many of the above studies address social systems, other network-based dynamical systems, such as Boolean networks, gene regulatory networks, and polynomial dynamical systems are used to study inference problems. Boolean networks are inferred from data in (Berestovsky and Nakhleh 2013). Polynomial dynamical systems are inferred using biochemical network data in (Laubenbacher and Stigler 2009).

2 Synchronous Dynamical Systems (SyDSs)

2.1 Formal Definitions

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.

At any time t, the **configuration** $\mathcal{C}$ of a SyDS is the n-vector $(s_1^t, s_2^t, \ldots, s_n^t)$, where $s_i^t \in \mathbb{B}$ is the state of node v_i at time t ($1 \leq i \leq n$). In a SyDS, all nodes compute and update their next state *synchronously*.

2.2 Classes of Local Functions

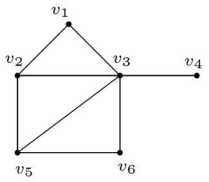
We consider two classes of local functions, namely **threshold** and **symmetric** functions. They are defined below.

(i) **Threshold functions:** The local function f_i associated with node v_i of a SyDS $\mathcal{S}$ is a t_i-**threshold** function for some integer $t_i \geq 0$ if the following condition holds: the value of f_i is 1 if the number of 1's in the input to f_i is *at least* t_i; otherwise, the value of the function is 0. Let d_i denote the degree of node v_i, and let t_i denote the threshold of node v_i. The number of inputs to the function f_i is $d_i + 1$. Thus, we assume that $0 \leq t_i \leq d_i + 2$. (The threshold values 0 and $d_i + 2$ allow us to realize local functions that always output 1 and 0 respectively.)

(ii) **Symmetric functions:** A local function f_i at node v_i is **symmetric** if the value of the function depends only on the number of 1's in the input. Thus, a symmetric function f_i with k inputs can be specified using a table with $k + 1$ rows, with row j specifying the value of the function when the number of 1's in the input to the function is exactly j, $0 \leq j \leq k$. Note that each threshold function is also a symmetric function.

We will use the term "symmetric SyDS" ("threshold SyDS") to refer to a SyDS whose local functions are all symmetric (threshold).

Example: Consider the threshold SyDS shown in Figure 1. Suppose the local transition functions at each of the nodes v_1, v_5 and v_6 is the 1-threshold function and the functions at v_2, v_3 and v_4 are 2-threshold functions. Assume that initially, v_3 is in state 1 and all other nodes are in state 0. During the first time step, the states of nodes v_1, v_5 and v_6 change to 1 since each of them has a neighbor (namely, v_3) in state 1. Also, the state of v_3 changes to 0 since its threshold is 2 and none of its neighbors is in state 1. The states of v_2 and v_4 don't change; they continue to be 0. During time step 2, v_2 and v_3 change to 1 but v_4 remains at 0. Once the system reaches the configuration $\mathcal{C} = (1, 1, 1, 0, 1, 1)$ at time step 2, it remains in that configuration forever.



Initial Config.:	(0, 0, 1, 0, 0, 0)
Config. at time 1:	(1, 0, 0, 0, 1, 1)
Config. at time 2:	(1, 1, 1, 0, 1, 1)

Note: Each configuration has the form $(s_1^t, s_2^t, s_3^t, s_4^t, s_5^t, s_6^t)$, where s_i^t is the state of node v_i at time t, $1 \leq i \leq 6$. The configuration at time 2 is a fixed point.

Figure 1: An Example of a SyDS.

Additional Terminology: 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 $\mathcal{C}'$ and $\mathcal{C}'$ is a **predecessor** of $\mathcal{C}$. Since our local functions are deterministic, each configuration has a unique successor; however, a configuration may have zero or more predecessors.

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 $N[v_i]$.

2.3 Query Model

The general problem addressed in this paper is that of correctly identifying the local functions of a SyDS by querying the system. We assume that the underlying network is known. Each query specifies a configuration $\mathcal{C}$ and the response from the system is the *successor* $\mathcal{C}'$ of $\mathcal{C}$. Since the state of each node is either 0 or 1, each query q and the response to q are bit vectors of length n. We consider two query modes. In the **batch** query mode, a user must submit all the queries together as a single batch. In the **adaptive** query mode, a user may submit the queries in several batches; the queries chosen in a batch may rely on the responses received from the system for the previous batches of queries. As will be seen, for threshold SyDSs, the adaptive query mode can significantly decrease the number of queries. The following additional definitions regarding queries will be used throughout this paper.

Given a query q and a node v_i, the **score** of q with respect to v_i, denoted by score(q,v_i), is the number of nodes in the closed neighborhood $N[v_i]$ of v_i that are set to 1 in q. Thus, score(q,v_i) gives the number of 1's in the input provided by q to the local function f_i at v_i.

Definition 1 *Let $\mathcal{S}$ be a symmetric SyDS. For any node v_i, let d_i denote the degree of v_i. (a) A query set Q **covers a node** v_i if for each j, $0 \leq j \leq d_i + 1$, there is a query $q \in Q$ such that score(q,v_i) = j. (b) A query set Q **covers a set** B of nodes if Q covers every node $v_i \in B$. (c) A query set Q is **complete** if it covers the node set V.*

When a query set Q covers a node v, the local symmetric function f_v can be correctly inferred from the responses to the queries in Q. Thus, complete query sets have the following property.

Observation 1 *Let $\mathcal{S}$ be a symmetric SyDS. If Q is a complete query set for $\mathcal{S}$, then each local function of $\mathcal{S}$ can be determined given the successor of each query in Q.* ■

3 Theoretical Results

In this section, we first present an algorithm for generating query sets under the batch mode for symmetric SyDSs. We then show that for threshold SyDSs, the number of queries can be substantially reduced under the adaptive query mode. We also establish lower bounds on the number of queries needed under both modes. We present complexity results that suggest that in general, generating complete query sets of minimum size is computationally intractable. We also develop an efficient heuristic to reduce the size of query sets by eliminating redundant queries.

3.1 Generating Query Sets: Batch Mode

We begin by defining the notion of a **monotone query sequence**. The sequence of queries constructed can be submitted as a batch to learn all the local functions of a symmetric SyDS. Using the notion of "sequence" allows us to point out an interesting connection between the problem of identifying local symmetric functions and a variant of the node coloring problem for the underlying graph.

Definition 2 *(a) Given two queries q_1 and q_2, we use the notation $q_1 \leq q_2$ to mean that every bit which is 1 in q_1 is also 1 in q_2. (b) A query sequence $\langle q_1, q_2, \ldots, q_r \rangle$ is **monotone** if for each i, $1 \leq i \leq r - 1$, $q_i \leq q_{i+1}$. (c) Let $\mathcal{S}$ be a SyDS in which each local function is symmetric and let M be a monotone query sequence. If M is also a complete query set for $\mathcal{S}$ (i.e., each node v of $\mathcal{S}$ is covered by M), then M is a **complete monotone query sequence**.*

We now present an algorithm to show that if the underlying graph G has n nodes, then there is a monotone complete query sequence M for $\mathcal{S}$ with at most $\min\{\Delta^2 + 2, n + 1\}$ queries, where Δ is the maximum node degree of G. This sequence of queries can be submitted as a batch to learn all the symmetric local functions. To establish this result, we recall the following definitions.

Definition 3 *(a) Given an undirected graph $H(V_H, E_H)$ and an integer $k \geq 1$, a k-**coloring** of H assigns a color from the set $\{1, 2, \ldots, k\}$ to each node of H such that for each edge $\{u, v\} \in E_H$, the colors assigned to u and v*

Input: Graph $G(V, E)$ of a symmetric SyDS $\mathcal{S}$.

Output: A monotone complete query sequence M for $\mathcal{S}$.

Steps:

1. Construct the graph $G^2(V, E')$.
2. Obtain a k-coloring of G^2 where $k \leq \min\{\Delta^2 + 1, n\}$.
3. Let C_1, C_2, ..., C_k denote the color classes created in Step 2. (Color class C_j consists of all nodes assigned color j, $1 \leq j \leq k$.) Create the query sequence $M = \langle q_0, q_1, \ldots, q_k \rangle$ with $k + 1$ queries as follows.
   (a) Query q_0 is a bit vector where every element is 0.
   (b) **for** $j = 1$ **to** k **do**
   Create query q_j by choosing the value 1 for all the nodes in $C_1 \cup \ldots \cup C_j$ and 0 for the other nodes.
4. Output the query sequence M.

Figure 2: Steps of the Algorithm ALG-MONOTONE-SEQ

*are different. (b) Given an undirected graph $G(V, E)$, the **square** of G, denoted by $G^2(V, E')$, is an undirected graph on the same vertex set V. The edge set E' is defined as: $\{u, v\} \in E'$ iff there is a path with at most 2 edges between u and v in G.*

We will also use the following result (West 2001): a graph $H(V_H, E_H)$ with maximum node degree Δ_H can be colored efficiently using at most $\Delta_H + 1$ colors.

Our algorithm ALG-MONOTONE-SEQ for generating a monotone complete query sequence M for the given SyDS $\mathcal{S}$ is shown in Figure 2. It is easy to see that the algorithm runs in polynomial time. The following theorem shows its correctness and estimates the number of queries generated.

Theorem 1 *Let $\mathcal{S}$ be a symmetric SyDS with graph $G(V, E)$ where $|V| = n$ and maximum node degree = Δ. Algorithm* ALG-MONOTONE-SEQ *produces a monotone complete query sequence M with $|M| \leq \min\{\Delta^2 + 2, n + 1\}$.*

Proof sketch: The sequence is produced by coloring G^2. Since the maximum node degree of G^2 is $\leq \Delta^2$, G^2 can be efficiently colored with $\min\{\Delta^2 + 1, n\}$ colors. The sequence produced has one query with all 0's and one query from each color class, for a total of at most $\min\{\Delta^2 + 2, n + 1\}$ queries. A proof that this is a monotone and complete query sequence appears in (Adiga et al. 2017b). ■

For some graphs with maximum node degree Δ, Algorithm ALG-MONOTONE-SEQ may generate a query sequence with $\Omega(\Delta^2)$ queries but it guarantees that the resulting query sequence is complete for a symmetric SyDS. We show in (Adiga et al. 2017c) that for graphs where $\Delta \geq (\log n)^2$, the number of queries can be reduced to $O(\Delta^{1.5} \log n)$, if we only need the query set to be complete with *high probability*.

3.2 Generating Query Sets: Adaptive Mode

For threshold SyDSs, the adaptive query mode can reduce the number of queries significantly. To illustrate this, consider a SyDS whose underlying graph is a **star graph** with n nodes; that is, there is one node v_1 with degree $n-1$ which is the root of the tree and each of the other nodes v_2 through v_n is a child of the root. As will be shown in the section on lower bounds, in the batch mode, $n+1$ queries are necessary even for the star graph to identify all the thresholds. However, under the adaptive mode, using the following method, $O(\log n)$ queries are sufficient.

The idea is simple: use *binary search* to identify the threshold of node v_1 whose degree is $n - 1$ using $O(\log n)$ queries. After this, the following 3 additional queries are sufficient to identify the thresholds of the remaining $n - 1$ nodes: a query with all 0's, a second query with all 1's and a third one in which v_1 has the value 1 and all the remaining nodes have the value 0. Thus, all the thresholds can be identified in $O(\log n)$ queries under the adaptive mode. Thus:

Proposition 1 *For a threshold-SyDS whose graph is a star with n nodes, all the threshold values can be found using $O(\log n)$ queries under the adaptive mode.* ■

The above idea can be applied to more general classes of graphs. Let a class of graphs with n nodes be called (α, β)**-simple**, if at most α nodes have degree $> \beta$ (the degree may be $\Omega(n)$) and all the remaining $n - \alpha$ nodes have a degree of at most β, with α and β being **constants** independent of n. Thus, each star graph belongs to the class of $(1, 1)$-simple graphs. It is shown in (Adiga et al. 2017b) that $O(\log n)$ queries are sufficient under the adaptive mode for any for (α, β)-simple graph with n nodes. As an extension of this result, it is also shown in (Adiga et al. 2017b) that if a graph with n nodes is **scale-free** (Easley and Kleinberg 2010) with exponent $\gamma \geq 1$, then $O(n^{\frac{2}{\gamma+1}})$ queries are sufficient under the adaptive mode.

3.3 Lower Bounds on Sizes of Query Sets

Here, we present lower bounds under batch and adaptive query modes. We begin with a result that provides a lower bound for any symmetric SyDS under the batch mode.

Proposition 2 *Let $\mathcal{S}$ be a symmetric SyDS where the underlying graph $G(V, E)$ has a maximum node degree Δ. Under the batch query model, every complete query set must contain at least $\Delta + 2$ queries.*

Proof sketch: Consider a node v_i with degree Δ. The symmetric function f_i at v_i has $\Delta + 2$ entries, since the number of 1's in the input to f_i varies from 0 to $\Delta + 1$. Thus, any query set Q with $|Q| < \Delta + 2$, cannot fully determine f_i. ■

As a simple consequence of the above proposition, the following result points out that there are SyDSs with n nodes for which every complete query set must have $n+1$ queries. This lower bound matches the upper bound of $n + 1$ given by Theorem 1 for all such graphs.

Corollary 1 *For a symmetric SyDS whose graph is a clique on n nodes, every complete query set under the batch mode must have at least $n + 1$ queries.* ■

The following result shown in (Adiga et al. 2017b) points out that there are threshold SyDSs for which a large number of queries are needed even under the adaptive query mode. However, this result does not rule out the possibility of smaller query sets for special graph classes.

Theorem 2 *For every $n \geq 1$, there is a threshold SyDS whose underlying graph is a clique on n nodes such that at least $n+1$ queries are necessary under the adaptive query mode to correctly identify all the threshold values.* ■

3.4 Complexity of Generating Small Monotone Complete Query Sequences

Here, we present a result that provides an indication of the difficulty of efficiently generating small query sets. Our complexity result is for the following problem which we call the **Short Monotone Complete Query Sequence** (SMCQS) problem: given the underlying graph $G(V, E)$ of a symmetric SyDS $\mathcal{S}$ and a positive integer k, is there a monotone complete query sequence Q with at most k queries for $\mathcal{S}$? We use a reduction from the Distance-2 coloring problem for graphs (McCormick 1983) to prove the following result; for details, see (Adiga et al. 2017b).

Theorem 3 *Problem* SMCQS *is **NP**-complete.* ■

3.5 Results for Query Set Compaction

Under the batch mode, after generating a complete set of queries, it is useful to reduce the size of the set by eliminating redundant queries. This **Query Set Compaction** (QSC) problem can be formulated as follows: given the underlying graph $G(V, E)$ of a symmetric SyDS $\mathcal{S}$, a *complete* query set Q and an integer $k \leq |Q|$, is there a subset $Q' \subseteq Q$ such that $|Q'| \leq k$ and Q' is also a complete query set for $\mathcal{S}$?

The following result shows the complexity of QSC. The proof in (Adiga et al. 2017c) uses a reduction from the **Minimum Set Cover** (MSC) problem (Garey and Johnson 1979).

Theorem 4 *Unless **P = NP**, QSC cannot be approximated in polynomial time to within the factor $o(\log n)$, where n is the number of nodes in the graph of the symmetric SyDS, even when the underlying graph has no edges.* ■

To complement the above hardness result, we present an efficient approximation algorithm with a performance guarantee of $O(\log n)$ for the QSC problem. The idea is to use a reduction from the QSC problem to the MSC problem and use a well known (greedy) approximation algorithm for MSC (Vazirani 2001) which provides a performance guarantee of $O(\log n)$.

The steps of our approximation algorithm Approx-QSC for QSC are shown in Figure 3. The performance of Approx-QSC is shown in the following theorem, whose proof appears in (Adiga et al. 2017c).

Theorem 5 *Algorithm Approx-QSC provides a performance guarantee of $O(\log n)$ where n is the number of nodes in the underlying graph of the SyDS.* ■

4 Experimental Results

We performed extensive experiments on more than 20 diverse real-world and synthetic networks. They are listed in Table 1 along with some of their properties. We present representative results for selected networks, with other networks exhibiting the same behavior unless stated otherwise.

Input: The underlying graph $G(V, E)$ of a symmetric SyDS $\mathcal{S}$ and a complete query set Q.

Output: A subset $Q' \subseteq Q$ such that Q' is also a complete query set and $|Q'|$ is as small as possible.

Steps:

1. To construct the base set X of the MSC instance, consider each node v_i; let d_i denote the degree of v_i. Create a set A_i of $d_i + 1$ elements, given by $A_i = \{a_{ik} : 0 \leq k \leq d_i\}$, for v_i. The set X is given by $X = \cup_{i=1}^{n} A_i$.
2. From each query $q_j \in Q$, construct a subset Y_j of X as follows. Initially, $Y_j = \emptyset$. For each $v_i \in V$, $1 \leq i \leq n$, if q_j sets k of the inputs to v_i to 1, then add the element a_{ik} to Y_j.
3. Use the greedy algorithm (Vazirani 2001) to get an approximate solution Y' to the resulting MSC instance.
4. Construct the query set Q' by choosing the query corresponding to each subset in Y' and output Q'.

Figure 3: Details regarding Algorithm Approx-QSC

We studied three approaches for inferring thresholds, two of which correspond to the batch mode and hence are applicable to symmetric SyDSs as well, and one being an adaptive approach. The first batch mode approach is based on coloring G^2 and the other is a probabilistic query approach mentioned in Section 3.1. In both cases, we applied the compaction algorithm (Figure 3) on the resulting complete query sets. Next, we developed a greedy algorithm for inferring thresholds under the adaptive mode and evaluated its performance.

Our theoretical results indicate that both network structure and the threshold assignments influence the number of queries required to infer the system. The experiments conducted were designed to further explore these aspects.

4.1 Method 1: G^2 Coloring Based Approach

We studied the performance of ALG-MONOTONE-SEQ (Figure 2). The results are in Table 2. For most real world networks considered in this paper, it gives the best possible performance, i.e., the number of colors used to color G^2, denoted by $n_c(G^2)$, is equal to $\Delta + 1$; this along with the query of all 0's is a lower bound on the size of a complete set (Proposition 2). For synthetic networks (random regular and Erdős-Rényi graphs) though, $n_c(G^2)$ is significantly higher than $\Delta + 1$, yet much lower than $\Delta^2 + 1$. The reader should note that the observed performance is due to a combination of the structure of G^2 and the nature of the greedy coloring scheme. We observe that unlike the synthetic networks considered, most of the real-world networks are scale-free with maximum degree much larger than the average degree d_{avg}. This is a possible reason for the superior performance of this approach. We also compared the results to the spectral radius bound, that is, the number of colors needed to color G^2 is at most $1 + \lambda_{\max}^2$ (Miao and Fan 2014). It is well-known that $\sqrt{\Delta} \leq \lambda_{\max} \leq \Delta$, and for the real-world networks considered, $\lambda_{\max}$ is indeed much less than Δ. However, despite this fact, we observe that $\lambda_{\max}^2 + 1$ is much

Table 1: Networks used in our experiments, their properties, and results of the different algorithms for inferring symmetric or threshold functions. The networks are grouped by type: social online, friendship, collaboration (Leskovec and Krevl 2014) and synthetic networks. To conserve space, we have provided range of values for some network families.

Network (num. of instances)	**Properties**					**Results Query set size**	
	Type	n	**avg. deg.** d_{avg}	**max. deg.** Δ	**Spec. rad.** $\lambda_{\max}$	**Meth. 1** $n_c(G^2)+1$	**Meth. 3** $t(v)=\frac{d(v)+2}{2}$
FB	social media	43,953	8.30	223	39.7	225	53
p2p-gnutella04	hw connectivity	10,876	7.35	103	17.08	105	31
Enron	email	33,696	10.73	1383	118.4	1385	624
Epinions	online opinions	75,879	10.69	3044	246	3046	294
Slashdot0811	online	77,360	12.13	2539	250.3	2541	214
Slashdot0902	online	82,168	12.27	2552	252.6	2554	267
Wikipedia	online voting	7,115	28.32	1065	138.2	1067	114
ca-astroph	co-author	17,903	22.00	504	94.43	506	76
ca-condmat	co-author	21,363	8.55	279	37.89	281	67
ca-grqc	co-author	4,158	6.46	81	45.62	83	25
ca-hepph	co-author	11,204	21.00	491	244.9	619	72
ca-hepth	co-author	8,638	5.74	65	31.03	67	28
cit-hepph	co-author	34,401	24.46	846	76.58	848	78
Clique	synthetic	1000	999	999	999	1001	8
Rand. reg. A (10)*	synthetic	1000	10,800	10,800	10,800	34-36,1001	Fig. 4(a)(0.0)
Rand. reg. B (10)	synthetic	80,000	10,12	10,12	10,12	38	20 (avg)
Erdős-Rényi (10)	synthetic	80,000	10, 12	25-28,27-32	11.1,13.04-13.09	36-38, 46-47	–

* Degrees are 10, 50, 100, 200, 250, 400, 500, 700, 800. For $d_{\text{avg}} = 50$, 100, $n_c(G^2)+1 =$ 348-358, 988-996, and for greater d_{avg}, $n_c(G^2) = 1000$.

larger than $n_c(G^2)+1$ in these cases.

Compaction. For the reasons mentioned above, query sets generated by this approach are already compact. Thus, the compaction algorithm of Section 3.5 did not eliminate any queries. A formal explanation for this appears in (Adiga et al. 2017c).

4.2 Method 2: Randomized Algorithm

In this approach, we use the probabilistic method mentioned in Section 3.1 to construct a complete set. The query set contains the configurations of all zeros, of all ones and $\ell\Delta$ random queries where ℓ queries are sampled from distributions $\mathbb{D}(i/\Delta)$ for $1 \le i \le \Delta$. Compared to Method 1, this is a very simple approach not requiring construction of G^2 or graph coloring. However, it is not guaranteed that the constructed query set is complete and therefore the process may have to be repeated a number of times. Further, the resulting set, even though large, can be compressed using the compaction algorithm.

We constructed 50 such query sets for three values of ℓ (2, 5 and 10) and checked if each of them is a complete set. For $\ell = 2$, out of the 50 sets none of them were complete. However, for $\ell = 10$, from 5 to 50 query sets turned out to be complete sets depending on the network. We applied the compaction algorithm on the complete sets generated by the randomized algorithm. The results for $\ell = 10$ are in Table 2. The compaction ratio depends on the size of complete set which was given as input. On an average, the combination of randomized algorithm and compaction gives query sets of size around 1.5 to 2 times that of Method 1 (Table 1 (Meth. 1)). However, comparatively these are much easier to generate.

Table 2: Results of Method 2.

Network	Query set size	% comp.	Network	Query set size	% comp.
FB	407	81	ca-grqc	153	81
p2p	159	84	ca-hepph	1201	75
Enron	2306	83	ca-hepth	140	78
Wikipedia	1420	86	cit-hepph	1240	85
ca-astroph	899	82	Rand. reg. A	$\approx 5\Delta$	40
ca-condmat	393	85	–	–	–

Performance of compaction. We note that compaction of query sets generated by Method 2 consistently yields 80% reduction in the size of the query set (Table 2).

4.3 Method 3: Adaptive Algorithm

While the two previous methods are for the batch mode, here we discuss an adaptive algorithm to infer the thresholds. We give an outline of the approach; a complete description appears in (Adiga et al. 2017b). For every node, let $t_L(v)$ and $t_H(v)$ be the minimum and maximum possible values of threshold that v can be assigned. These values quantify the uncertainty about the threshold. The threshold is said to have been inferred when $t_H(v) = t_L(v)$. In a query q, if score(q, v) falls in the range $[t_L(v), t_H(v) - 1]$, then, the

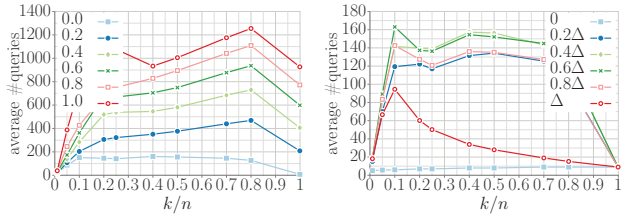


Figure 4: Experiments with 1000 node random k-regular graphs. (a) The threshold of a node is randomly assigned an integer in the interval $\lfloor (k+2)(1-\theta)/2, (k+2)(1+\theta)/2 \rceil$. The legend shows values of θ. (b) All nodes are assigned a fixed threshold relative to k. The legend shows values of t_i.

uncertainty reduces to either $[\text{score}(q,v)+1, t_H(v)-1]$ or $[t_L(v), \text{score}(q,v)]$ depending on the state in the successor configuration. In this heuristic, we use a greedy adaptive approach where the current query is constructed iteratively in the following way. To begin with, all nodes are in state 0. We first choose that node, say $v_{\max}$, for which the threshold range is a maximum. We set exactly $\lfloor (t_L(v)+t_H(v))/2 \rfloor$ of nodes in its closed neighborhood to state 1. This guarantees a reduction of the range by half. In the next iteration, we ignore all nodes in G within distance-2 of $v_{\max}$ and repeat this process. The query is fully constructed when there are no more vertices to consider. After each query, the range $[t_L(v), t_H(v)-1]$ for every v is updated based on its state in the successor. We terminate this process when for all v, $t_L(v) = t_H(v)$. The analysis of our experimental results follows.

Influence of threshold values and ranges. In general, the number of queries required is highly dependent on the possible threshold values of nodes. We conducted experiments in the following manner. Let $0 \leq \theta \leq 1$ be a real number. For a fixed value of θ, each node v was assigned a threshold value uniformly at random from the interval $\lfloor (d(v)+2)(1-\theta)/2, (d(v)+2)(1+\theta)/2 \rceil$. For $\theta = 0$, the interval corresponds to the fixed threshold of $(d(v)+2)/2$; for $\theta = 1$, any value from 0 to $d(v)+2$ is possible. The results are in Figure 4(a) and 5(a) for random k-regular and real-world networks respectively. For the random-regular graphs, the number of queries (averaged over 10 instances of graphs for each k) increases from $O(\log k)$ to as high as n, the size of the graph. We note that for $k = n-1$, this is in accordance with Theorem 2. For the real-world graphs, we see that increasing the range of threshold has the effect of gradually increasing the number of queries, but the number is less than 1.5Δ. In Figure 4(b), we investigate the influence of the threshold value on query set size. Again, we considered random k-regular graphs with varying k. Every node was assigned the same threshold. We see that the number of queries required is maximum when the threshold is around $\Delta/2$, and it decreases as the threshold approaches either 0 or Δ.

Influence of network structure. Theoretical bounds on query set sizes developed in Section 3 were in terms of Δ and n. Here, our objectives are two-fold. Firstly, we compare our adaptive approaches to the non-adaptive bounds, particularly the number of queries required relative to $\log \Delta$, Δ and Δ^2. Secondly, we investigate the effect of graph density and degree distribution on the performance of the heuristic.

We note that graph density plays an important role in the performance of the algorithm. First, we consider synthetic k-regular networks. In Figure 4(b), we see that for low values of k, the number of queries required is very small, but it increases rapidly (for higher values of thresholds). When the graph is sparse, for any node, the number of nodes within distance two (namely, k^2+1) is small. Therefore, for every query constructed by the heuristic, the uncertainty range of around n/k^2 nodes (the "$v_{\max}$" vertices) is halved. However, as k increases, this number decreases drastically. Hence we see that the number of queries required increases. However, as the graph density increases, the intersection of the neighborhoods of any two nodes is large, and this has the effect of reducing the variation in the scores of nodes. Therefore, particularly when the range of threshold values is limited, far fewer queries are required to infer thresholds in dense networks.

Progress towards inferring thresholds. In Figure 5(b), we plot the accumulated threshold ranges for all vertices as the algorithm moves from one query to the next. We note that within one-fifth of the total query set size, the total accumulated threshold range decreases well below 5% of its original value for the majority of studied networks.

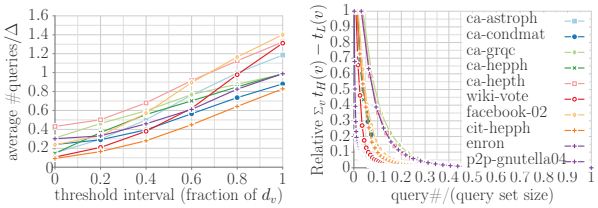


Figure 5: Inferring thresholds for real-world networks. (a) Adaptive heuristic for varying threshold ranges. (b) Progress made by the adaptive algorithm (Method 3) in each query.

5 Limitations and Future Work

One limitation of our work is that queries and the responses from the system are assumed to specify the states of all the nodes in the system. An important research direction is to extend the results to allow specifications of partial configurations in queries and responses. Further, our focus was on dynamical systems with threshold and symmetric functions. Thus, another research direction is to consider other classes of functions. Finally, it is also of interest to explore the use of queries to infer other components of a dynamical system (e.g., the network topology).

Acknowledgments: This work has been partially supported by DARPA Cooperative Agreement D17AC00003 (NGS2), DTRA CNIMS (Contract HDTRA1-11-D-0016-0001), DTRA Comprehensive National Incident Management System Contract HDTRA1-17-D-0023, 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

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

Adiga, A.; Kuhlman, C. J.; Marathe, M. V.; Ravi, S. S.; Rosenkrantz, D. J.; and Stearns, R. E. 2017b. Learning the behavior of dynamical systems – Part I: Deterministic query generation methods. NDSSL TR#2017-1035, Biocomplexity Institute of Virginia Tech, Blacksburg, VA.

Adiga, A.; Kuhlman, C. J.; Marathe, M. V.; Ravi, S. S.; Rosenkrantz, D. J.; and Stearns, R. E. 2017c. Learning the behavior of dynamical systems – Part II: Randomized query generation and query compaction. NDSSL TR#2017-1036, Biocomplexity Institute of Virginia Tech, Blacksburg, VA.

Berestovsky, N., and Nakhleh, L. 2013. An evaluation of methods for inferring boolean networks from time-series data. *PLoS One* 8:e66031–1–e66031–9.

Centola, D., and Macy, M. 2007. Complex contagions and the weakness of long ties. *American Journal of Sociology* 113(3):702–734.

Christakis, N. A., and Fowler, J. H. 2007. The spread of obesity in a large social network over 32 years. *New England Journal of Medicine* 357(4):370–379.

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

Gai, P., and Kapadia, S. 2010. Contagion in financial networks. *Proceedings of the Royal Society A* 466:2401–2423.

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.

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

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

He, X.; Xu, K.; Kempe, D.; and Liu, Y. 2016. Learning influence functions from incomplete observations. arXiv:1611.02305 [cs.SI].

Juba, B. 2016. Learning abductive reasoning using random examples. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 999–1007.

Karsai, M.; Iniguez, G.; Kaski, K.; and Kertesz, J. 2014. Complex contagion process in spreading of online innovation. *Journal of the Royal Society Interface* 11:20140694–1–20140694–8.

Kearns, M. J., and Vazirani, U. V. 1994. *An Introduction to Computational Learning Theory*. MIT Press.

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

Laubenbacher, R., and Stigler, B. 2009. Design of experiments and biochemical network inference. In *Algebraic and Geometric Methods in Statistics*, 1–13.

Leskovec, J., and Krevl, A. 2014. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data.

McCormick, S. T. 1983. Optimal approximation of sparse Hessians and its equivalence to a graph coloring problem. *Math. Programming* 26(2):153–171.

Miao, L., and Fan, Y. 2014. The distance coloring of graphs. *Acta Mathematica Sinica* 30(9):1579–1587.

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

Papadimitriou, C. H., and Roughgarden, T. 2003. Equilibria in symmetric games. Report, Stanford University.

Romero, D. M.; Meeder, B.; and Kleinberg, J. 2011. 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*, 695–704. ACM.

Rosenthal, S. B.; Twomey, C. R.; Hartnett, A. T.; Wu, H. S.; and Couzin, I. D. 2015. 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.

Schelling, T. C. 1971. Dynamic models of segregation. *Journal of Mathematical Sociology* 1:143–186.

Schelling, T. C. 1978. *Micromotives and Macrobehavior*.

Sherman, S. G.; Ganna, D. S.; Tobin, K. E.; Latkin, C. A.; Welsh, C.; and Bielenson, P. 2009. The life they save may be mine: Diffusion of overdose prevention information from a city sponsored programme. *International Journal of Drug Policy* 20:137–142.

Ugander, J.; Backstrom, L.; Marlow, C.; and Kleinberg, J. 2012. Structural diversity in social contagion. *Proceedings of the National Academy of Sciences* 109(16):5962–5966.

Valente, T. W. 2010. *Social Networks and Health: Models, Methods, and Applications*.

Valiant, L. G. 1984. A theory of the learnable. *Communications of the ACM* 18(11):1134–1142.

Vazirani, V. V. 2001. *Approximation Algorithms*.

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

West, D. B. 2001. *Introduction to Graph Theory*.

Zhang, M.; Mathew, T.; and Juba, B. A. 2017. An improved algorithm for learning to perform exception-tolerant abduction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 1257–1265.

Zhu, K.; Chen, Z.; and Ying, L. 2017. Catch'em all: Locating multiple diffusion sources in networks with partial observations. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 1676–1683.