Learning the Behavior of a Dynamical System via a “20 Questions” Approach

Abstract

Developing techniques to infer the behavior of networked social systems has attracted a lot of attention in the literature. Using a 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 which suggest that, in general, the problem of generating query sets of minimum size is computationally intractable. We present efficient heuristics that produce query sets under both batch and adaptive query modes. Our results show that a small number of appropriately chosen queries are provably sufficient to learn all the node functions. We also present experimental results to demonstrate the performance of our heuristics on over 20 well known networks.

Introduction

Background and Motivation. 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 overdose prevention (Sherman et al. 2009); viruses like Ebola (Siettos et al. 2015); 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 particular 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 agents need multiple reinforcing interactions to adopt a contagion; i.e., for cases where $t_i \geq 1$. (Watts 2002) argues that threshold models are used in a host of settings where incomplete information exists or when there is insufficient time to make more deliberate decisions.

In particular, we note that small changes in the thresholds of nodes can make large differences in population dynamics. An example is provided in (Granovetter 1978), where a change in one node's threshold by a value of 1, in an arbitrarily 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 state) of nodes in this neighborhood.* Symmetric functions, which generalize threshold functions, also serve as natural models in game theoretic settings (Papadimitriou and Roughgarden 2003).

Some works have studied threshold inference in a **passive** setting (e.g., (Adiga et al. 2017)) where observations are *given* and the problem is to infer thresholds from these 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 the system properties based on the observed 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 question”[1].

Our work is similar in spirit—but quite different in prob-

[1]See Wikipedia entry on this game.

lem domain and results—to some of the recent works on inference (e.g. (Kleinberg, Mullainathan, and Ugander 2017)). To the best of our knowledge, this is the first work which approaches the problem of inference of dynamical systems from a combinatorial and algorithmic perspective. In doing so, we relate it to well-studied graph theoretic problems such as coloring. The formulation also enables us to quantify rigorously the complexity of inferring such systems.

Summary of Results. Our focus is on the following problem: given the underlying graph of a dynamical system, construct queries to identify all the local functions. The optimization goal is to minimize the number of queries. We present both theoretical and experimental results as summarized below.

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 that reduces the size of a query set (or makes it compact) by eliminating redundant queries and establish its performance guarantee.
4. We evaluate the proposed 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.

All proofs for propositions, lemmas, and theorems appear in the supplement.

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). (Kearns and Vazirani 1994) study the problem of learning normal forms and Boolean functions. Works such as (González-Bailón et al. 2011; Romero, Meeder, and Kleinberg 2011) infer thresholds from social media data. Learning the source nodes of infection for contagion spreading 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. 2017) provides several problems aimed at inferring thresholds in threshold-based 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, there are several works that include allowance for errors in inferring system properties. These include (Valiant 1984; Juba 2016; He et al. 2016; Zhang, Mathew, and Juba 2017; Kleinberg, Mullainathan, and Ugander 2017).

There are several challenges in determining individual node thresholds in realistic settings: (i) data are collected at discrete time intervals (not continuously), (ii) there may be time delay effects in agents observing their neighborhoods, and (iii) inherent stochasticity (Valente 1996; Berry and Cameron 2017). Practical guidelines and issues for threshold measurement are discussed in (Berry and Cameron 2017). Here, we investigate problems of inferring local functions using rigorous formulations, supplementing them with experimental results from heuristics.

Synchronous Dynamical Systems (SyDSs)

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*.

Classes of Local Functions

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

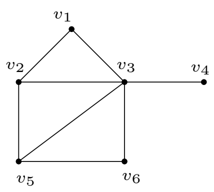
(i) **Threshold functions:** The local function f_v associated with node v of a SyDS $\mathcal{S}$ is a t_v-**threshold** function for some integer $t_v \geq 0$ if the following condition holds: the value of f_v is 1 if the number of 1's in the input to f_v is *at least* t_v; otherwise, the value of the function is 0. Let d_v denote the degree of node v, and let t_v denote the threshold 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.)

(ii) **Symmetric functions:** A local function f_v at node v is **symmetric** if the value of the function depends only on the number of 1's in the input. Thus, a symmetric function f_v

with k inputs can be specified using a table with $k+1$ rows, with row i specifying the value of the function when the number of 1's in the input to the function is exactly i, $0 \leq i \leq k$. 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 graph of a 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; that is, $\mathcal{C}$ is a **fixed point** for this system. □



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. A **fixed point** is a configuration $\mathcal{C}$ for which the successor is $\mathcal{C}$ itself.

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 $N[v_i]$.

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. We consider two query modes. In the **batch** query mode, a user must submit all the queries at the same time 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 by 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.* ■

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 and prove its performance guarantee.

Generating Query Sets Under the 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**.*

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

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. Use the algorithm of Theorem 1 to obtain a k-coloring of G^2 where $k \leq \min\{\Delta^2 + 1, n\}$.
3. Let $C_1, C_2, \ldots, C_k$ 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.

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 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 known result (West 2001).

Theorem 1 *Let $H(V_H, E_H)$ be a graph with maximum node degree Δ_H. Then, 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 2 *Let $\mathcal{S}$ be a symmetric SyDS whose graph $G(V, E)$ has n nodes and maximum node degree Δ. Algorithm* ALG-MONOTONE-SEQ *(Figure 2) produces a monotone complete query sequence M with at most $\min\{\Delta^2 + 2, n + 1\}$ queries.* ■

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. 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*. This result is stated below.

Theorem 3 *Let $\mathcal{S}$ be a symmetric SyDS with graph $G(V, E)$ where $|V| = n$ and maximum node degree = Δ. A query set Q of size $O(\Delta^{1.5} \log(n))$ which is complete with probability at least $\left(1 - \frac{1}{n}\right)$ can be constructed for $\mathcal{S}$.* ■

Generating Query Sets Under the 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 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 $O(\log n)$ queries under the adaptive mode.

The above idea can be applied to a more general class 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. The following result shows the usefulness of the adaptive query mode for (α, β)-simple graph.

Theorem 4 *For any threshold SyDS whose underlying graph G belongs to the class of (α, β)-simple graphs, $O(\log n)$ queries are sufficient in the adaptive mode to identify all the threshold values.* ■

The above result can be used to establish a bound on the number of queries under the adaptive mode for scale-free graphs as stated below.

Theorem 5 *For a threshold SyDS whose underlying graph $G(V, E)$ is scale-free with exponent $\gamma \geq 1$, the thresholds can be found using $O\left(n^{\frac{2}{\gamma+1}}\right)$ queries under the adaptive query mode.* ■

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 1 *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.*

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 2 for all graphs.

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

We now establish a lower bound under the adaptive query model to show 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 6 *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.* ■

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. In particular, we will show the **NP**-completeness of the following problem.

Short Monotone Complete Query Sequence (SMCQS)

Given: The underlying graph $G(V, E)$ of a SyDS $\mathcal{S}$ where each local function is symmetric and a positive integer k.

Question: Is there a monotone complete query sequence Q with at most k queries for $\mathcal{S}$?

Theorem 7 *Problem* SMCQS *is **NP**-complete.* ■

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. We refer to this as the **Query Set Compaction** problem and its formulation is as follows.

Query Set Compaction (QSC)

Given: The underlying graph $G(V, E)$ of a symmetric SyDS $\mathcal{S}$, a *complete* query set Q and an integer $k \leq |Q|$.

Question: Is there a subset $Q' \subseteq Q$ such that (i) $|Q'| \leq k$ and (ii) Q' is also a complete query set for $\mathcal{S}$?

The following result points out the intractability of QSC.

Theorem 8 *(a) The problem QSC is **NP**-complete even when the underlying graph has no edges. (b) Unless **P** = **NP**, QSC cannot be approximated to within the factor $o(\log n)$, where n is the number of nodes in the underlying graph of the SyDS.* ■

To complement the non-approximability result of the previous section, 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 well known **Minimum Set Cover** (MSC) problem (Garey and Johnson 1979). A (greedy) approximation algorithm for MSC which provides a performance guarantee of $O(\log n)$ for the MSC problem is well known (Vazirani 2001).

The steps of our approximation algorithm Approx-QSC for QSC are shown in Figure 3. It can be seen that the approximation algorithm runs in polynomial time. The performance guarantee provided by Approx-QSC is indicated in the following theorem.

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 is empty. 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 the element a_{ik} is added to the set 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

Theorem 9 *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.* ■

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.

We studied three approaches for inferring thresholds, two of which correspond to the batch mode and hence 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 random query approach based on Theorem 3. In both these cases, we applied the compaction algorithm (Figure 3) on the complete sets that were constructed. Next, we propose a greedy algorithm for inferring thresholds in the adaptive mode and evaluate 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.

Method 1: G^2 Coloring Based Approach

We studied the performance of ALG-MONOTONE-SEQ (Figure 2). The results of are in Table 2. For most real world networks considered in this paper, it gives the best possible performance, i.e., $n_c(G^2)$ is equal to $\Delta + 1$, the lower bound on the size of complete set (Proposition 1). For synthetic networks (random regular and Erdős-Rény 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 being much larger than average degree d_{avg}. This is

Table 1: Networks used in our experiments, their properties, and results of the different algorithms for inferring local functions or thresholds. The networks are grouped by type: social online, friendship, co-authorship (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}, $n_c(G^2) = 1000$.

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 $1 + \lambda^2_{\max}$ (Miao and Fan 2014). It is a well-known fact 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 $\lambda^2_{\max} + 1$ is much larger than $n_c(G^2) + 1$ in these cases.

Compaction. We note that the query set generated by this approach is already compact, i.e., no subset of queries can be complete. We provide an intuitive explanation for this in the supplement.

Method 2: Randomized Algorithm

In this approach, we use the method of Theorem 3 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 $1 \leq i \leq \Delta$. 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. However, as discussed below, with ℓ sufficiently large, we can obtain a complete set in few repetitions. 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	% Compaction	Network	Query set size	% Compaction
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).

Method 3: Adaptive Algorithm

While the previous two methods correspond to the batch mode, here we develop an adaptive algorithm to infer the thresholds. We give an outline of the approach. The algorithm description is in the supplementary material. 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 uncertainty reduces to either [score(q, v) +

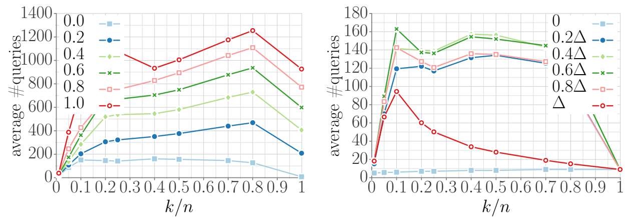


Figure 4: **Experiments with 1000 node random k-regular graphs.** (a) The threshold of a node is randomly assigned an integer in the interval $\left[(k+2)(1-\theta)/2, (k+2)(1+\theta)/2\right]$. The legend shows values of θ. (b) All nodes are assigned a fixed threshold relative to k. The legend shows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

$1, t_H(v) - 1]$ or $[t_L(v), \text{score}(q, v)]$ depending on the state observed 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 vertex, say $v_{\max}$ for which the threshold range is 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 in 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 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 the nodes can be assigned. 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 $\left[(d(v)+2)(1-\theta)/2, (d(v)+2)(1+\theta)/2\right]$. Note that for $\theta = 0$, the interval corresponds to the fixed threshold of $(d(v)+2)/2$ and 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 an order of $\log k$ to as high as n, the size of the graph. We note that for $k = n-1$, this is in accordance with Theorem 6. 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 that 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. The theoretical bounds developed in the previous sections provide bounds with respect to size of the graph and maximum degree. 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 will consider the synthetic 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 every node, the number of nodes within distance two (k^2+1 nodes) 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 neighborhoods of any two nodes is large which has the effect of reducing the variation in the scores of nodes. Therefore, particularly when the range of threshold values is limited, the thresholds can be inferred with far fewer queries than for sparser 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-tenth of the total query size, the total accumulated threshold range decreases to 5% of its original value for all the studied networks.

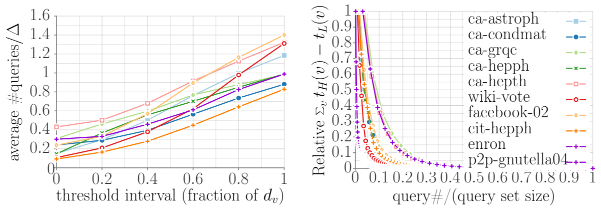


Figure 5: **Inferring thresholds for real-world networks.** (a) Adaptive heuristic for varying threshold ranges. (b) Progress made by the adaptive algorithm (Method 3) in each query.

Future Work

In this paper, our focus was on learning the threshold and symmetric functions of dynamical systems. One direction for future work is to investigate other classes of functions. Another direction is to explore the use of queries to infer other components of a dynamical system such as the network topology. A third direction is to extend the results for other classes of dynamical systems.

References

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

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

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*. Cambridge, MA: MIT Press.

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

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, Santa Fe, 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.

Siettos, C.; Anastassopoulou, C.; Russo, L.; Grigoras, C.; and Mylonakis, E. 2015. Modeling the 2014 ebola virus epidemic—agent-based simulations, temporal analysis and future predictions for liberia and sierra leone. *PLOS Currents Outbreaks* 1–22.

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. 1996. Social network thresholds in the diffusion of innovations. *Social Networks* 18:69–89.

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.

Learning the Behavior of a Dynamical System via a “20 Questions” Approach (Supplementary Material)

This supplement provides proofs for many results mentioned in the main paper. (As some of the proofs involve long mathematical formulas, this supplement has been formatted using the single column mode.)

Statement and Proof of Theorem 2

Statement of Theorem 2: Let $\mathcal{S}$ be a symmetric SyDS whose graph $G(V, E)$ has n nodes and maximum node degree Δ. Algorithm ALG-MONOTONE-SEQ (Figure 2) produces a monotone complete query sequence M with at most $\min\{\Delta^2 + 2, n + 1\}$ queries.

Proof: We first show that Step 2 of the algorithm can indeed color G^2 using at most $\min\{\Delta^2+1, n\}$ colors. Since the maximum node degree in G is Δ, each node v of G has at most Δ neighbors and at most $\Delta(\Delta - 1)$ nodes at a distance of 2 from v. Thus, the maximum node degree in G^2 is at most $\Delta(\Delta-1)+\Delta$ = Δ^2. Hence, by Theorem 1, G^2 can be colored using at most Δ^2+1 colors. Since G^2 has n nodes, n colors are sufficient. Thus, G^2 can be colored with at most $k = \min\{\Delta^2+1, n\}$ colors. Hence, the number of queries in $M = k + 1$ is at most $\min\{\Delta^2 + 2, n + 1\}$.

We now argue that the query sequence $M = \langle q_0, q_1, \ldots, q_k\rangle$ is monotone. Query q_0 is the bit vector with all 0’s. For any $j \geq 1$, query q_j sets all the nodes in color classes C_1 through C_j to 1 and the remaining nodes to 0. Thus, each node that is set to 1 in query q_j remains 1 in all the subsequent queries q_{j+1}, . . ., q_k. In other words, the sequence is monotone.

Thus, we are left with the proof that M is complete; that is, for each node v with degree α in G and each value ℓ, $0 \leq \ell \leq \alpha+1$, there is a query q in M such that score(q,v_i) = ℓ. Query q_0 ensures that score(q,v_i) = 0. For the other values of ℓ, consider the closed neighborhood $N[v]$ of v in G. Note that $|N[v]| = \alpha + 1$. For each pair of nodes v_x and v_y in $N[v]$, there is a path consisting of at most two edges in G. Thus, the nodes in $N[v]$ form a clique in G^2. In other words, each node in $N[v]$ must be in a different color class of G^2. Let C_{j_1}, C_{j_2}, . . ., $C_{j_{\alpha+1}}$ denote the color classes of G^2 in which the nodes in $N[v]$ appear, and assume without loss of generality that $j_1 < j_2 < \ldots < j_{\alpha+1}$. It is easy to see that for $1 \leq \ell \leq \alpha + 1$, query q_{j_ℓ} ensures that score(q_{j_ℓ},v_i) = ℓ. This completes the proof of Theorem 2. ■

Generating Coloring from a Monotone Complete Query Sequence

Theorem 2 shows that a monotone complete query sequence can be constructed from the coloring of the graph G^2. We now point out that this relationship is not accidental; indeed, from every monotone complete query sequence a valid coloring of G^2 can be generated.

Theorem 10 *Let $\mathcal{S}$ be a SyDS where each local function is symmetric. Let $G(V, E)$ be the underlying graph of $\mathcal{S}$. Suppose there is a monotone complete query sequence M with ℓ queries for $\mathcal{S}$. Then, G^2 can be colored using $\ell - 1$ colors.*

Proof: Let $M = \langle q_1, q_2, \ldots, q_\ell\rangle$ denote the given monotone complete query sequence for $\mathcal{S}$. Let C_i denote the set of nodes of G which have the value 0 in q_i and the value 1 in q_{i+1}, $1 \leq i \leq \ell - 1$. Assign color i to all the nodes in C_i, $1 \leq i \leq \ell - 1$. We now prove that this scheme assigns a color to each node and that this is a valid coloring of G^2.

First, we prove that each node is assigned a color. To see this, note that since M is a monotone complete query sequence, query q_1 has all its bits set to 0 and q_ℓ has all its bits set to 1. Therefore, for each node v, there is an index r such that the value assigned to v in q_r is 0 and that in q_{r+1} is 1. Thus, v appears in set C_r and receives color r. The monotonicity of M ensures that v remains 1 in queries q_{r+1} through q_ℓ. In other words, the color assigned to v does not change subsequently.

We now prove by contradiction that the above method produces a valid coloring of G^2. So, suppose that v_i and v_j are two nodes which receive the same color, say k, but G^2 has the edge $\{v_i, v_j\}$. By our coloring scheme, both v_i and v_j had the value 0 in q_k and the value 1 in q_{k+1}. There are two cases to consider.

Case 1: The edge $\{v_i, v_j\}$ is in G.

Note that both v_i and v_j have color k. Let score(v_i,q_k) = α. Since both v_i and v_j changed from 0 in q_k to 1 in q_{k+1} and $\{v_i, v_j\}$ is an edge in G, score(v_i,q_{k+1}) $\geq \alpha + 2$. Because M is monotone, none of the other queries in M provides a score of $\alpha + 1$ to v_i. This contradicts the assumption that M is complete.

Case 2: The edge $\{v_i, v_j\}$ is not in G but in G^2.

In this case, there is a node v_x such that the edges $\{v_i, v_x\}$ and $\{v_j, v_x\}$ are in G. Let score(v_x,q_k) = β. Since both v_i and v_j changed from 0 to 1 in q_{k+1} and both $\{v_i, v_x\}$ and $\{v_j, v_x\}$ are edge sin G, score(v_x,q_{k+1}) $\geq \beta + 2$. Because M is monotone, none of the other queries in M provides a score of $\beta + 1$ to v_x. Again, this contradicts the assumption that M is complete, and this completes the proof of Theorem 10. ■

Generating a Complete Query Set with High Probability

Here, we present a proof of Theorem 3. A statement of the theorem is given below.

Statement of Theorem 3: Let $\mathcal{S}$ be a symmetric SyDS with graph $G(V, E)$ where $|V| = n$ and maximum node degree = Δ. A query set Q of size $O(\Delta^{1.5} \log(n))$ which is complete with probability at least $\left(1 - \frac{1}{n}\right)$ can be constructed for $\mathcal{S}$.

In proving the above theorem, we will use the following notation. For any node v, let f_v denote the symmetric function at v and let $d(v)$ denote the degree of v. Note that each input to f_v is an integer that gives the number of 1's assigned to the closed neighborhood of v.

Proof of Theorem 3: Our method, discussed below, produces a query set with size at most $22\Delta\sqrt{\Delta+2}\log(n\Delta)$.

We first note that the all zeros and the all ones configurations can be used to query $f_v(0)$ and $f(d(v)+1)$, respectively for all $v \in V$. This contributes the additive term 2. For a real number x, let $\langle x \rangle$ denote $\lfloor x + .5 \rfloor$, the integer closest to x. Let $Q = \left\{q_{ij} \sim \mathbb{D}\left(\frac{i}{\Delta+1}\right) \mid 1 \leq i \leq \Delta,\ 1 \leq j \leq 22\sqrt{\Delta+2}\log(n\Delta)\right\}$ be the query set. For any $v \in V$, $b \in \{1, \ldots, d(v)\}$ and $q \sim \mathbb{D}(z)$, where $z = \left\langle \frac{b(\Delta+1)}{d(v)+1} \right\rangle$, we have $\Pr\left(f_v(b) \text{ is queried in } q\right) = \Pr\left(d_1[v,q] = b\right)$. Now, letting $p' = \Pr\left(d_1[v,q] = b\right)$, we have

$$\begin{aligned}
p' &\geq \binom{d(v)+1}{b}\left(\frac{z}{\Delta+1}\right)^b\left(1 - \frac{z}{\Delta+1}\right)^{d(v)+1-b} \\
&\geq \frac{1}{11\sqrt{d(v)+2}} \\
&\geq \frac{1}{11\sqrt{\Delta+2}},
\end{aligned}$$

where the second inequality follows from Lemma 1 (below). Now, $\Pr\left(f_v(b) \text{ is not queried by } Q\right) \leq \Pr\left(f_v(b) \text{ is not queried by } q_{zj},\ 1 \leq j \leq 22\sqrt{\Delta+2}\log(n\Delta)\right)$.

The latter quantity is $\leq \left(1 - \frac{1}{11\sqrt{\Delta+2}}\right)^{22\sqrt{\Delta+2}\log(n\Delta)} < \frac{1}{(n\Delta)^2}$. By union bound, $\Pr\left(Q \text{ is not a complete set}\right) = \Pr\left(\exists v, b \text{ such that } f_v(b) \text{ is not queried by } Q\right)$. Now, the latter quantity is $\leq \sum_{v \in V}\sum_{b=1}^{\Delta}\Pr\left(f_v(b) \text{ is not queried by } Q\right) < \frac{1}{n\Delta}$.

This completes the proof. ■

Lemma 1 *For any three positive integers $b \leq d \leq D$ and $z = \left\langle \frac{bD}{d} \right\rangle$, $\binom{d}{b}\left(\frac{z}{D}\right)^b\left(1 - \frac{z}{D}\right)^{d-b} \geq \frac{1}{11\sqrt{d+1}}$.*

We will first prove the following claims.

Claim 1 $\left(1 + \frac{1}{b}\right)^b$ *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,

$$\begin{aligned}
\frac{1 + b\left(\frac{b+1}{b}\right)}{b+1} &\geq \left(1^1\left(\frac{b+1}{b}\right)^b\right)^{\frac{1}{b+1}} \\
\frac{(b+1)+1}{b+1} &\geq \left(\frac{b+1}{b}\right)^{\frac{b}{b+1}}.
\end{aligned}$$

■

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

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 $b < \frac{d}{2}$, $h(b+1,d) \leq h(b,d)$ and for $b \geq \frac{d}{2}$, $h(b+1,d) > h(b,d)$, 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 $b < \frac{d}{2}$, $b' \geq b$ and when $b \geq \frac{d}{2}$, $b' < b$. Applying Claim 1, we have for $b < \frac{d}{2}$, $h(b+1,d) \leq h(b,d)$ and for $b \geq \frac{d}{2}$, $h(b+1,d) > h(b,d)$.

When b is even, $h\big(\frac{d}{2}, d\big) \geq \frac{1}{\sqrt{2d}}$. Now we will show that when b is odd, $h\big(\frac{d-1}{2}, d\big) \geq \frac{1}{\sqrt{2(d+1)}}$. Let $b = 2k+1$.

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

The inequality follows from Claim 1. Therefore, when d is odd, $h\big(\frac{d-1}{2}, d\big) > h\big(\frac{d-1}{2}, d+1\big) \geq h\big(\frac{d+1}{2}, d+1\big) \geq \frac{1}{\sqrt{2(d+1)}}$. ■

Claim 3 *For any positive* $x \leq \frac{1}{2}$, $1 - x \geq e^{-2x}$.

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

Proof of Lemma 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, $\big|z - \frac{bD}{d}\big| \leq \frac{1}{2}$.

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

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

The last but one inequality follows from Claim 3.

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

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

This completes the proof. ■

Lower Bounds on Query Set Sizes

Statement of Proposition 1: 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 least $\Delta + 2$ queries.

Proof: Under the batch mode, suppose a complete query set Q has *less than* $\Delta + 2$ queries. Since G has a node v of degree Δ, the number of 1's in the input to the symmetric function f_v at v varies from 0 to $\Delta + 1$, a total of $\Delta + 2$ values. Thus, there is at least one value k such that none of the queries in Q has a score of k with respect to v. Hence, the query set cannot correctly determine the value of the function f_v when the number of 1's in the input to f_v is exactly ks. This contradicts the assumption that Q is a complete query set. The proposition follows. ■

Statement of Theorem 6: 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.

Proof: Consider a threshold SyDS $\mathcal{S}$ whose underlying graph $G(V, E)$ is a clique on n nodes. Let $V = \{v_1, v_2, \ldots, v_n\}$. We will tentatively choose the threshold of node v_i to be i to answer queries under the adaptive model, $1 \leq i \leq n$. We will show that if the total number of queries is less than $n+1$, the answers to the queries cannot distinguish between this tentative assignment and a slightly different assignment of threshold values.

Suppose Q be a sequence of queries under the adaptive model, with $|Q| \leq n$. We generate the responses to the queries using the chosen tentative assignment of threshold values to nodes. The threshold of any node of $\mathcal{S}$ is in the range 0 through $n+1$ (where the threshold value $n+1$ indicates the function which is zero for every input). Since G is a clique, the the closed neighborhood of each node is the node set V. Thus, each query $q \in Q$ provides the same score to each node of G. There are $n+1$ scores in the range 0 through n. Since Q has at most n queries, there is at least one value k, $0 \leq k \leq n$ such that none of the queries in Q provides the score k. We have three cases depending on the value of k.

Case 1: $k = 0$. In this case, from the responses to the queries in Q, one cannot distinguish between the case where the threshold of node v_1 is 0 and the case where the threshold of v_1 is 1. (In both cases, the new state of v_1 is 1 in the response to each query in Q.)

Case 2: $k = n$. In this case, from the responses to the queries in Q, one cannot distinguish between the case where the threshold of node v_n is n and the case where the threshold of v_n is $n+1$. (In both cases, the new state of v_n is 0 in the response to each query in Q.)

Case 3: $1 \leq k \leq n-1$. In this case, from the responses to the queries in Q, one cannot distinguish between the case where the threshold of node v_k is k and the case where the threshold of v_k is $k+1$. (In both cases, the responses have the following property. For any query $q \in Q$ where score(q,v_k) $\leq k-1$, the new state of v_k is 0 in the response. For any query $q \in Q$ where score(q,v_k) $\leq k+1$, the new state of v_k is 1 in the response.)

Thus, under the adaptive model, a query set with n or fewer queries cannot correctly identify all the thresholds for the chosen SyDS. This completes the proof of Theorem 6. ■

Complexity of Generating Small Monotone Complete Query Sequences

Statement of Theorem 7: Problem SMCQS is **NP**-complete.

Proof: It is easy to see that SMCQS is in **NP**. To prove **NP**-hardness, we use a reduction from the **Distance-2 Coloring** (D2C) problem defined as follows: given an undirected graph $G(V, E)$ and an integer r, is G^2 r-colorable? It is known that D2C is **NP**-complete (McCormick 1983).

The reduction is straightforward. Given an instance of the D2C problem consisting of graph G and integer r, we obtain an instance of the SMCQS problem where the graph is G itself and the length k of the query sequence is $r+1$. It was shown in the proof of Theorem 2 that when G^2 is r-colorable, there is a monotone complete query sequence with $k = r+1$ queries. Also, it was shown in the proof of Theorem 10 that from any monotone complete query sequence of length $r+1$, one can obtain a valid coloring of G^2 with r colors. Thus, there is a solution to the SMCQS problem iff there is a solution to the D2C problem and this completes the proof. ■

Complexity of Query Set Compaction

Statement of Theorem 8: (a) The problem QSC is **NP**-complete even when the underlying graph has no edges. (b) Unless **P** = **NP**, QSC cannot be approximated to within the factor $o(\log n)$, where n is the number of nodes in the underlying graph of the SyDS.

Our proof of this result relies on known results for the **Minimum Set Cover** (MSC) problem which is defined as follows: given a base set $X = \{x_1, x_2, \ldots, x_n\}$, a collection $Y = \{Y_1, Y_2, \ldots, Y_m\}$, where each Y_j is a subset of X, $1 \leq j \leq m$, and an integer $\alpha \leq m$, is there a subcollection $Y' \subseteq Y$ such that (i) $|Y'| \leq \alpha$ and (ii) the union of all the sets in Y' is equal to X? It is well known that MSC is **NP**-complete (Garey and Johnson 1979) and that unless **P** = **NP**, it cannot be approximated to within the factor $o(\log{(n)}$, where n is the size of the base set (Vazirani 2001).
Proof:

Part (a): It is easy to see that QSC is in **NP**. We prove **NP**-hardness through a reduction from MSC. Let the given instance of MSC consist of base set $X = \{x_1, x_2, \ldots, x_n\}$, collection $Y = \{Y_1, Y_2, \ldots, Y_m\}$ of nonempty subsets of X and integer $\alpha \leq m$. Without loss of generality, we may assume that each element of X appears in some subset in Y; otherwise, there is no solution to the MSC instance. We will construct the underlying graph $G(V, E)$ of a SyDS $\mathcal{S}$ and a complete query set Q for $\mathcal{S}$ as follows.

1. The node set V of G is given by $V = V_1 \cup V_2$, where $V_1 = \{v_1, v_2, \ldots, v_n\}$ is in one-to-one correspondence with the base set $X = \{x_1, x_2, \ldots, x_n\}$ of the MSC instance and $V_2 = \{v_{n+1}\}$ consists of just one node. (Thus, V has a total of $n + 1$ nodes.)
2. The edge set E of G is empty; that is, the degree of each node is 0. Thus, for each node $v \in V$, the number of 1's in the input to the local function f_v at v can only be either 0 or 1.
3. The query set Q consists of $m + 1$ queries (where $m = |Y|$) constructed as discussed below. Note that each query is an $(n + 1)$-bit vector, where the i^{th} bit specifies the value of node v_i, $1 \leq i \leq n + 1$.
   (a) For each $Y_j \in Y$, $1 \leq j \leq m$, Q contains a query q_j constructed as follows. Let $Y_j = \{x_{j_1}, x_{j_2}, \ldots, x_{j_r}\}$. Then, in query q_j, the bits corresponding to the nodes $v_{j_1}, v_{j_2}, \ldots, v_{j_r}$ are all 1 and the other bits are 0.
   (b) We add one more query q_{m+1} to Q; in query q_{m+1}, bits 1 through n are set to 0 and bit $n + 1$ is set to 1.
4. The upper bound on the size of the required subset Q' of queries is set to $\alpha + 1$.

This completes the construction of the QSC instance. It can be seen that the construction can be carried out in polynomial time. We now show that Q is a complete query set for $\mathcal{S}$.

Claim 1: The query set Q constructed above is a complete query set for $\mathcal{S}$.

Proof of Claim 1: We must show that for each node $v_i \in V$, Q contains two queries, say q_{i_0} and q_{i_1}, such that score(q_{i_0},v_i) = 0 and that score(q_{i_1},v_i) = 1. First, consider any node v_i, where $1 \leq i \leq n$. Query q_{m+1} sets the value of v_i to 0; thus score(q_{m+1},v_i) = 0. Suppose the element x_i (corresponding to node v_i) appears in subset Y_j. By our construction, query q_j sets the value of v_i to 1; thus, score(q_j,v_i) = 1. For node v_{n+1}, each query q_j created from Y_j sets the value of v_{n+1} to 0; that is, score(q_j,v_{n+1}) = 0; Also, query q_{m+1} sets the value of v_{n+1} to 1; thus, score(q_j,v_{n+1}) = 1. The claim follows. □

We now prove that there is a solution to the QSC instance if and only if there is a solution to the MSC instance.

Part 1: Suppose there is a solution Y' to the MSC instance consisting of sets Y_{j_1}, Y_{j_2}, ..., Y_{j_ℓ}, for some $\ell \leq \alpha$. Consider the query set $Q' = \{q_{j_1}, q_{j_2}, \ldots, q_{j_\ell}, q_{m+1}\}$, which includes the queries corresponding to the sets in Y' along with query q_{m+1}. Note that $Q' \subseteq Q$. Also, since $\ell \leq \alpha$, $|Q'| \leq \alpha + 1$. Thus, we only need to show that Q' is a complete query set. Consider any node v_i, where $1 \leq i \leq n$. Query q_{m+1} sets the value of v_i to 0; thus, score(q_{m+1},v_i) = 0. Further, Since Y' is a set cover, the element x_i (corresponding to node v_i) appears in some subset $Y_{j_z} \in Y'$. By our construction, in query q_{j_z}, the value of v_i is 1; thus, score(q_{j_z},v_i) = 1. For node v_{n+1}, each query $q \in Q' - \{q_{m+1}\}$ sets the value of v_{n+1} to 0; thus, score(q,v_{n+1}) = 0. Further, query q_{m+1} sets the value of v_{n+1} to 1; in other words, score(q_{m+1},v_{n+1}) = 1. Hence, Q' is a complete query set.

Part 2: Let Q' be a solution to the QSC instance with $|Q'| \leq \alpha + 1$. We claim that $q_{m+1} \in Q'$. This is because q_{m+1} is the only query for which score(v_{n+1} to 1. Define $Q'' = Q' - \{q_{m+1}\}$. Let $|Q''| = \ell$ and note that $\ell \leq \alpha$. Further, let $Q'' = \{q_{j_1}, q_{j_2}, \ldots, q_{j_\ell}\}$. Consider the following subcollection Y' of Y given by $Y' = \{Y_{j_1}, Y_{j_2}, \ldots, Y_{j_\ell}\}$. We now show that Y' is a solution to the MSC instance. To see this consider any element $x_i \in X$, where $1 \leq i \leq n$. Query $q_{m+1} \in Q'$ sets node v_i to 0. Since Q' is a complete query set, some query $q_{j_z} \in Q''$ must set v_i to 1. By our construction, the subset Y_{j_z} contains x_i. Thus, Y' is a set cover. Since $|Y'| \leq \alpha$, Y' is a solution to the MSC instance, and this completes the **NP**-hardness proof.

We use the same reduction to prove the non-approximability result. Suppose $\mathcal{A}$ is an approximation algorithm that provides a performance guarantee of $\rho = o(\log n)$ for the QSC problem, where n is the number of nodes in the underlying graph of the SyDS. We will show that $\mathcal{A}$ can be used to construct an $2\rho = o(\log n)$ approximation algorithm for the MSC problem, contradicting the known non-approximability result for MSC. Towards this proof, consider any instance of the MSC optimization problem. Let OPT(MSC) denote the value of an optimal solution (i.e., the minimum size of a set cover) for the MSC instance and let OPT(QSC) denote the value of an optimal solution (i.e., the minimum size of a complete query subset) for the QSC instance constructed from the MSC instance. In the **NP**-hardness proof, we showed that

$$\text{OPT(QSC)} \;\leq\; \text{OPT(MSC)} + 1. \tag{1}$$

Suppose we run Algorithm $\mathcal{A}$ on the resulting QSC instance. Since $\mathcal{A}$ provides a ρ approximation, the solution produced by $\mathcal{A}$ has at most $\rho\,$OPT(QSC) queries. From this query set, it was shown in the **NP**-hardness proof that a solution to the MSC instance with $\rho\,$OPT(QSC) $- 1$ subsets can be constructed. Letting APPROX(MSC) denote the resulting number of subsets, we have

$$\begin{aligned}
\text{APPROX(MSC)} \quad &\leq \quad \rho\,\text{OPT(QSC)} - 1 \\
&\leq \quad \rho\;[\text{OPT(MSC)} + 1] - 1 \qquad \text{(using Equation (1))} \\
&\leq \quad 2\rho\,\text{OPT(MSC)} \qquad\qquad\quad \text{(since OPT(MSC)} \geq 1\text{)}.
\end{aligned}$$

Thus, if $\mathcal{A}$ provides a performance guarantee of $\rho = o(\log n)$ for the QSC problem, then there is a $2\rho = o(\log n)$ approximation algorithm for the MSC problem as well. This completes the proof of Theorem 8. ■

Statement of Theorem 9: Algorithm Approx-QSC provides a performance guarantee of $O(\log n)$ for the QSC problem, where n is the number of nodes in the underlying graph of the SyDS.

To establish the above theorem, we need the following lemma.

Lemma 2 *The reduction from QSC to MSC used in Steps 1 and 2 of Approx-QSC (Figure 3) produces an instance of MSC such that any solution with r subsets to the MSC instance is a solution with r queries to the QSC instance and vice versa.*

Proof: First, consider any solution $Y' = \{Y_{j_1}, Y_{j_2}, \ldots Y_{j_r}\}$ with r subsets to the MSC instance. Let $Q' = \{q_{j_1}, q_{j_2}, \ldots, q_{j_r}\}$ be the corresponding query set with r queries. We need to show that Q' is a complete query set; that is, for any node v_i $(1 \le i \le n)$ and any integer k, $0 \le k \le d_i$, there is a query $q \in Q'$ such that the number of 1's in the input to the local function f_i at v_i due to q is exactly k. To see this, note that Y' is a set cover. Thus, there is a set $Y_{j_z} \in Y'$ such that the element $a_{ik} \in X$ appears in Y_{j_z}. By our construction, a_{ik} was added to Y_{j_z} because query q_{j_z} provides exactly k 1's to the local function f_i at v_i.

To prove the converse, let $Q' = \{q_{j_1}, q_{j_2}, \ldots, q_{j_r}\}$ be a solution with r queries to the QSC instance. Consider the collection Y' of sets given by $Y' = \{Y_{j_1}, Y_{j_2}, \ldots Y_{j_r}\}$. We claim that Y' is a solution to the MSC instance. To see this, consider any element $a_{ik} \in X$. Since Q' is a complete query set, there is some query $q_{j_z} \in Q'$ such that the number of 1's in the input to the function f_i at node v_i due to q_{j_z} is exactly k. By our construction, set Y_{j_z} contains the element a_{ik}. In other words, Y' is a solution to the MSC instance with r sets. ■

The following is an immediate consequence of the above lemma.

Observation 2 *Let OPT(QSC) denote the size of an optimal query set for a given QSC instance and let OPT(MSC) denote the size of an optimal solution to the MSC instance obtained at the end of Step 2 of Approx-QSC. Then, OPT(QSC) = OPT(MSC).* ■

We can now prove Theorem 9.

Proof of Theorem 9: Let OPT(QSC) denote the size of an optimal query set for the QSC instance and let OPT(MSC) denote the size of an optimal solution to the MSC instance. As mentioned earlier, the size of the base set X is $2|E| + n$. Since $|E| < n^2$, we note that $|X| < 3n^2$. The greedy algorithm for MSC provides a performance guarantee of $O(\log |X|$. Since $|X| < 3n^2$, this performance guarantee is $O(\log n)$. Thus, the approximation algorithm for MSC produces a solution with at most $O(\log n)$ OPT(MSC) sets. By Lemma 2, any solution to MSC with r subsets leads to a solution with r queries for QSC. Thus, the size of the resulting query set is at most $O(\log n)$ OPT(MSC) which is equal to $O(\log n)$ OPT(QSC) by Observation 2. Thus, Approx-QSC has a performance guarantee of $O(\log n)$. ■

Statement of Theorem 4: For any threshold SyDS whose underlying graph G belongs to the class of (α, β)-simple graphs, $O(\log n)$ queries are sufficient in the adaptive mode to identify all the threshold values.

Proof: We give an approach that uses $\alpha \lceil \log n \rceil + \beta^2 \lceil \log \beta \rceil$ queries under the adaptive model. Since α and β are constants independent of n, the number of queries is $O(\log n)$.

The idea is to first use a separate binary search for each of the α nodes of degree q; this uses at most $\alpha \lceil \log n \rceil$ queries. Let V' denote the subset of nodes of G such that each node in V' has a degree of at most β. Now, if we construct G^2, it can be seen that the subgraph G' of G^2 on V', it can be seen that the maximum node degree in G' is at most β^2. Therefore, G' can be colored using at most β^2 colors and a binary search can be done simultaneously for all the nodes in each color class. This will use at most $\beta^2 \lceil \log \beta \rceil$ queries, giving a total of $\alpha \lceil \log n \rceil + \beta^2 \lceil \log \beta \rceil$ queries. ■

Statement of Theorem 5: For a threshold SyDS whose underlying graph $G(V, E)$ is scale-free with exponent $\gamma \ge 1$, the thresholds can be found using $O\big(n^{\frac{2}{\gamma+1}}\big)$ queries under the adaptive query mode.

Proof: Let $V_1 \subseteq V$ correspond to the set of nodes with degree at most β and V_2 be the remaining set. By Theorem 4, the number of queries required is at most $g(\beta) := |V_2| \log n + \beta^2 \log \beta$. Since $|V_2|$ is the number of nodes with degree $\ge \beta$, it is at most

$$\sum_{x=\beta}^{\Delta} c' \frac{n}{x^\gamma} \le c'' \int_\beta^\infty \frac{n}{x^\gamma} dx = c \frac{n}{\beta^{\gamma-1}},$$

for some constants c, c' and c''. Therefore, $g(\beta) = c\frac{n}{\beta^{\gamma-1}} \log n + \beta^2 \log \beta$. For $\beta > 0$, $g(\beta)$ is a convex function. Equating its first derivative to 0 and rearranging, we note that $g(\beta)$ attains a minimum value for β satisfying $\beta^{\gamma+1} \log \beta = \Theta(n \log n)$ or $\beta = \Theta\big(n^{\frac{1}{\gamma+1}}\big)$. For this value, $g(\beta) = O\big(n^{\frac{2}{\gamma+1}}\big)$. ■

Why Query Sets Generated by Method I are Difficult to Compress: As mentioned in the paper, our experimental results show that query sets generated by coloring G^2 are already compact; that is, i.e., no subset of these query sets can be complete. This can be explained intuitively as follows. If this set is not compact, there exists at least one query which is not required. We recall that by construction, the query set can be arranged as a monotone increasing sequence $q_0, q_1, \ldots,$ such that in query q_i all nodes of color i are set to switched to state 1. Suppose query q_k is not required for the set to be complete. Then, using the arguments in the proof of Theorem 2, we can show that if all vertices of color k were assigned color $k+1$, the coloring would still be valid. This means that in the greedy strategy, all nodes colored $k+1$ could actually have been assigned color k. Since the greedy strategy always gives the minimum color available to nodes, this situation cannot occur.

Method 3: Description of Our Adaptive Algorithm

Algorithm 1: Greedy heuristic to infer the thresholds.

```
   Data: Network $G(V, E)$ and thresholds $t_v$ for every node $v$.
   Result: Complete query set $Q$
 1 for $v$ in $V$ do
 2     Let $t_L(v) = 0$ and $t_H(v) = d(v) + 2$;
 3 end
 4 Let $V_t = V$ denote the set of nodes for which threshold needs to be inferred;
 5 Let $Q = \varnothing$ be the query set;
 6 while $V_t \neq \varnothing$ do
 7     Let $i = 1$;
 8     Let $q_i$ be the current query. Let $q_i[v] = 0, \forall v \in V$;
 9     Let $V_{\text{rem}} = V_t$;
10     while $V_{rem} \neq \varnothing$ do
11         Let $v_{\max} = \arg\max_{v \in V_{\text{rem}}} t_H(v) - t_L(v)$;
12         Set exactly $\lfloor (t_H(v_{\max}) - t_L(v_{\max}))/2 \rfloor$ neighbors of $v_{\max}$ to state 1 in $q_i$;
13         $V_{\text{rem}} \leftarrow V_{\text{rem}} \setminus \{N[v_{\max}, G^2]\}$;
14     end
15     Compute the successor $s$ of $q_i$;
16     //Update $t_L(v)$ and $t_H(v)$ for all $v \in V_t$//
17     for $v \in V_t$ do
18         if s[v]=0 and $t_L(v) \leq$ score$(q_i, v)$ then
19             $t_L(v) \leftarrow$ score$(q_i, v) + 1$;
20         end
21         else if s[v]=1 and $t_H(v) >$ score$(q_i, v)$ then
22             $t_H(v) \leftarrow$ score$(q_i, v)$;
23         end
24     end
25     Remove all nodes $v$ from $V_t$ such that $t_L(v) = t_H(v)$;
26     $Q \leftarrow Q \cup \{q_i\}$;
27 end
```