Journal of Machine Learning Research 23 (2022) 1-43 Submitted 4/21; Revised 7/22; Published 8/22

Using Active Queries to Infer Symmetric Node Functions of Graph Dynamical Systems*

Abhijin Adiga ABHIJIN@VIRGINIA.EDU
Network Systems Science and Advanced Computing Division, Biocomplexity Institute and Initiative, University of Virginia, Charlottesville, VA 22904.

Chris J. Kuhlman CJK8GX@VIRGINIA.EDU
Network Systems Science and Advanced Computing Division, Biocomplexity Institute and Initiative, University of Virginia, Charlottesville, VA 22904.

Madhav V. Marathe MARATHE@VIRGINIA.EDU
Network Systems Science and Advanced Computing Division, Biocomplexity Institute and Initiative and Department of Computer Science, University of Virginia, Charlottesville, VA 22904.

S. S. Ravi SSRAVI0@GMAIL.COM
Network Systems Science and Advanced Computing Division, Biocomplexity Institute and Initiative, University of Virginia, Charlottesville, VA 22904 and Department of Computer Science, University at Albany – State University of New York, Albany, NY 12222.

Daniel J. Rosenkrantz DROSENKRANTZ@GMAIL.COM
Network Systems Science and Advanced Computing Division, Biocomplexity Institute and Initiative, University of Virginia, Charlottesville, VA 22904 and Department of Computer Science, University at Albany – State University of New York, Albany, NY 12222.

Richard E. Stearns THESTEARNS2@GMAIL.COM
Network Systems Science and Advanced Computing Division, Biocomplexity Institute and Initiative, University of Virginia, Charlottesville, VA 22904 and Department of Computer Science, University at Albany – State University of New York, Albany, NY 12222.

Editor: Karsten Borgwardt

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 (where all the queries must be submitted together) and adaptive (where the set of queries submitted at a stage may rely on the answers to previous queries). We establish bounds on the number of queries under both batch and adaptive query modes using vertex coloring and probabilistic methods. Our results show that a small number of appropriately chosen queries are provably sufficient to correctly learn all the local functions. 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. Also, we present a

*. A preliminary version of this paper appeared in the Proceedings of AAAI 2018 (Adiga et al., 2018b).

query compaction algorithm that identifies and removes redundant queries from a given query set. Our algorithms were evaluated through experiments on over 20 well-known networks.

Keywords: Graph dynamical systems, threshold and symmetric functions, active queries, lower and upper bounds, complexity

1. Introduction

1.1 Discrete dynamical systems and their significance

Discrete dynamical systems are used in a host of settings to understand population-level social contagion dynamics in terms of individual (human) agent behavior. Examples include the spread of health behaviors (Valente, 2010) such as needle sharing in drug use (Latkin, 1998; Valente, 2012) and overdose prevention (Sherman et al., 2009); segregation (Schelling, 1971); financial contagions (Gai and Kapadia, 2010); starting to use online communication tools (Karsai et al., 2014); and coordination (Rosenthal et al., 2015). The frameworks in these papers and our paper are network representations of populations, where nodes and edges represent entities (such as humans) and pairwise interactions, respectively. Our focus is on a particular class of discrete dynamical systems, called **Synchronous Dynamical Systems** (SyDSs). A formal definition of this model is given in Section 2.

Each of the works cited above 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 have already contracted it. This number for node v_i is called its **threshold** τ_i. We are interested in **complex contagions** (Centola and Macy, 2007) that are characteristic of social contagions. In this case, some agents may need multiple reinforcing interactions to adopt a contagion; that is, some nodes have thresholds of 2 or more. These models are used to study many problems, such as those defined above, and other social phenomena (e.g., Granovetter (1978); Schelling (1978); Centola and Macy (2007); Valente (1995); Easley and Kleinberg (2010)). Furthermore, 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. Thus, threshold models are used widely in the social sciences for understanding social contagions.

1.2 Motivation for local function inference

Our work is focused on inferring threshold (or more generally, symmetric) functions of nodes in dynamical systems. It is known that small changes in the thresholds of agents can make a large difference in population dynamics. An example is provided in (Granovetter, 1978), where a change in one agent's threshold, by a value of 1, changes population-level collective action from non-existent to full. Several papers have used mined data to infer thresholds for applications ranging from protests, to Twitter messaging, to joining social media (e.g., González-Bailón et al. (2011); Romero et al. (2011b); Ugander et al. (2012); Easley and Kleinberg (2010)). Importantly, in all of these cases, heterogeneous (i.e., non-uniform) thresholds among agents were inferred. Also, researchers have identified different types of influence based on agent attributes. For example, there are gender-based differences in contracting obesity and depression from peers (Christakis and Fowler, 2007; Stevens and

Prinstein, 2005). *Thus, agent thresholds must be determined based on an agent's (local) neighborhood, behavior (and states) of agents in this neighborhood, and individual behavior.* **Symmetric functions**, which generalize threshold functions, also serve as natural models in game theoretic settings (Papadimitriou and Roughgarden, 2003).

1.3 Active querying and its significance

In general, an inference algorithm may receive observed data about the dynamical system or submit queries to the system and obtain responses from the system. The former approach, which has been studied by many researchers (e.g., González-Bailón et al. (2011); Adiga et al. (2017); Romero et al. (2011a)), will be discussed briefly in Section 1.5. In this paper, we pursue the latter approach. We study inference by *active querying*, where the user has some *control* over what information is extracted from the dynamical system by querying it. Several behavioral studies have been conducted in a network setting in the context of public goods games, collective identity, and team-building exercises (Broere et al., 2019; Cedeno-Mieles et al., 2020; Charness et al., 2014). In all these works, human subjects (agents) are allowed or required to coordinate with one another (thus forming a network) to achieve their goals. The principal who conducts these games designs situations by providing incentives for agents to act in a certain manner. The individual and collective behaviors resulting from these experiments are then analyzed. Such settings have been effectively modeled as network dynamical systems (Santos et al., 2008; Cedeno-Mieles et al., 2020). Active querying of a dynamical system is akin to such controlled experimentation in the real-world experiments to understand agent behavior in a network setting. Since every experiment has a cost associated with it, a natural objective is to minimize the number of queries used to infer a system.

Here, we focus on a model where each query is a system configuration $\mathcal{C}$, with each **configuration** being an ordered tuple of states of the nodes of a dynamical system at a particular time t. The response from the system is the **successor** $\mathcal{C}'$ of $\mathcal{C}$, that is, the configuration of the system at time $t+1$. The goal is to infer the node functions of the system from the responses. Our query model is motivated by the following two lines of research.

1. First, learning Boolean functions is an important area of learning theory (e.g., Abasi et al. (2014); Angluin and Slonim (1994); Sloan et al. (2013); Angluin et al. (1992, 1993)). In this setting, each query (usually referred to as a **membership query**) specifies an input α to an unknown function f and an oracle returns the value $f(\alpha)$. Our query model is an extension of membership queries to a network setting. In our case, since multiple Boolean functions (one for each node of the dynamical system) must be inferred, each query (i.e., a configuration) provides inputs to all the functions, and the output is another configuration which provides the values of all the functions for the specified inputs. Our work is similar in form to the teacher model (e.g., Goldman and Kearns (1995); Dasgupta et al. (2019)) in the context of concept learning, where a teacher selects instances so that a student can learn the target concept from a concept class. In all these cases (including ours), the objective is to find a minimum set of queries that is sufficient to learn the object.

2. Second, our query model can also be thought of as an idealized version of *hypothetical* queries (i.e., alternate scenarios) that are useful in inferring local behaviors of individuals in the context of social contagions such as opinions or health issues (e.g., Sloan et al. (2013); Centola (2010, 2011)). For concreteness, we consider opinion propagation over a network, where the values 0 and 1 represent respectively positive and negative opinions on a certain topic. To understand when an individual (corresponding to a node in a social network) will change her opinion, a principal (e.g., a social scientist or a health-care professional) may pose a query of the form "If the opinions of your neighbors are represented by this Boolean vector $\mathbf{v}$, what will be your opinion?". Such queries do *not* set the opinions of the nodes in the social network to true (or actual) values. Instead, the principal tries to determine the situations that cause changes in an individual's behavior by analyzing the responses of the individual to a set of queries where the opinions of the neighbors of the individual are hypothetical values. While this can be done by issuing such queries to each node separately (Centola (2010, 2011)), the underlying network provides a way to combine these individual queries into a single configuration so that the number of queries can be further reduced. Thus, our query model exploits the network structure in constructing a small set of queries to learn the local functions.

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 et al. (2017)) or the popular and well-studied area of *active learning* (Settles, 2009; Dasgupta and Langford, 2009). We will discuss this in more detail in Section 1.5. To the best of our knowledge, this is the first work which approaches the inference problem for dynamical systems from a combinatorial and algorithmic perspective. In doing so, we relate it to well-studied graph theoretic approaches such as node coloring and probabilistic methods. The formulation also enables us to quantify rigorously the complexity of inferring such systems.

1.4 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. 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; this strategy is similar to the one used in games such as "Twenty Questions"[1]. The optimization goal is to minimize the number of queries. We present both theoretical and experimental results as summarized below. In the following, we use G to denote the underlying network of the SyDS with n nodes and Δ to denote the maximum node degree of G.

(a) Generating concise query sets for symmetric node functions under the batch mode. We develop an algorithm for generating query sets under the batch mode to correctly identify all the symmetric local functions of the dynamical system. Such a set is called a *complete query set*. The main idea used in this algorithm is **distance-two vertex coloring** of the network G. We show that the size of a complete query set is at most $\chi(G^2) + 1 \leq \min\{\Delta^2 + 2, n + 1\}$, where $\chi(G^2)$ is the minimum number of colors

1. See the Wikipedia entry on this game.

required for distance-two coloring of G (Theorem 10). We also establish a lower bound of $\Delta + 2$ on the size of any complete query set (Proposition 3).

(b) Complexity of generating optimal query sets. The query set generated by the method in (a) satisfies a *monotonicity* property (defined in Section 4.1). We show that the problem of finding an optimal *monotone* complete query set is **NP**-complete (Theorem 17). This provides an indication of the difficulty of efficiently generating optimal query sets.

(c) Sharper bounds based on probabilistic methods. We show that a query set of size $O(\Delta^{1.5} \log n)$ which is complete with probability at least $1 - \frac{1}{n}$ can be obtained using a sampling technique (Theorem 12). Using more sophisticated techniques based on Lovász Local Lemma (Mitzenmacher and Upfal, 2005), we show that there exists a complete query set of size $O(\Delta(\log \Delta)^{2.5})$ (Theorem 13). We note that this bound is asymptotically better than the $\Delta^2 + 2$ bound based on distance-two coloring mentioned above.

(d) Query set compaction. We formulate the problem of query set compaction, i.e., minimizing the size of a query set by deleting redundant queries. We show that the problem is, in general, **NP**-complete. We present an approximation algorithm for the problem and prove that it achieves the best performance guarantee to within constant factors, under the common assumption that $\mathbf{P} \neq \mathbf{NP}$ (Theorems 23 and 26).

(e) Inferring threshold functions under the adaptive query mode. As can be expected, adaptive query mode, when applicable, can produce significantly smaller query sets compared to the batch mode. Our results for adaptive querying are based on binary search. This enables us to find the threshold of a node whose local function has k inputs using $O(\log k)$ queries. Using an extension of this technique, we show that the thresholds of all the nodes in a scale-free (i.e., power law) graph with exponent $\gamma \geq 1$ can be found using $O\big([n \log n]^{\frac{2}{\gamma+1}}\big)$ queries (Theorem 22). In addition, we show that for a dynamical system where the underlying network is a clique and each node is associated with a threshold function, $n + 1$ queries are necessary even under the adaptive query mode (Theorem 5). For general graphs, we develop a greedy heuristic based on binary search and distance-two coloring to construct query sets.

(f) Extensive experimentation on synthetic and real-world networks. We evaluated the various query generation algorithms under the batch mode on more than 20 real-world and synthetic networks. For most real-world networks, our algorithm based on distance-two coloring generates query sets of minimum size. We present experimental results to show that the combination of randomized query generation followed by query set compaction produces small query sets under the batch mode for many well known social networks. We evaluated the greedy heuristic under the adaptive mode (mentioned in Item (e) above) for various settings of networks and threshold assignments. Our results (Section 7) show that for most cases, it significantly outperforms the batch mode algorithms. We also study the effect of network structure and threshold assignments on the size of query sets.

1.5 Related Work

The problem of inferring unknown components of systems has received attention in the literature. There are several works on the passive mode of inference. For instance, 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 et al., 2011b) infer thresholds from social media data. Abrahao et al. (2013), Gomez-Rodriguez et al. (2010) and Soundarajan and Hopcroft (2010) consider the problem of inferring the network structure given the contagion propagation model. Learning the source nodes of infection for contagion spreading is addressed in (Zhu et al., 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. Recently, there have been several works on learnability of dynamical system properties under the Probably Approximately Correct (PAC) learning framework. Lokhov (2016) uses a dynamic message-passing algorithm to reconstruct parameters of a spreading model given infection cascades. Narasimhan et al. (2015) and He et al. (2016) have studied the learnability of the influence function of popular stochastic propagation models, and Adiga et al. (2019) consider the problem of learning node thresholds. Inference problems for a class of local functions, motivated by user behavior in Facebook like networks, are studied in Adiga et al. (2020). Several works show the sensitivity of results to other dynamical systems features, such as network structure (Allen and Gale, 2000; Ganesh et al., 2005; Chakrabarti et al., 2008; Prakash et al., 2012).

Active querying is studied in (Kleinberg et al., 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 $n = |U|$. The independent cascade model under the active query model has also been studied (Adiga et al., 2018a).

We note that our work is different from the area of *active learning* (e.g., Dasgupta and Langford (2009); Settles (2009)). An active learning algorithm has access to a limited number of labeled instances and a large number of unlabeled instances initially. The algorithm submits appropriately chosen unlabeled instances as queries to an oracle which returns the labels of those instances. This process allows an algorithm to learn more effectively using fewer labeled instances since the algorithm is allowed to select the instances used in the learning process. The main difference in our setting is that there are no labels; instead, our inference algorithms use queries, with each query specifying a configuration $\mathcal{C}$ and requiring the oracle to return the successor configuration $\mathcal{C}'$. Like active learning, our algorithms also choose queries appropriately so that the local functions can be inferred with a small number of queries.

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.

1.6 Organization

The remainder of this paper is organized as follows. Section 2 presents the formal definition of a synchronous dynamical system and also explains our query model. Section 3 presents lower bounds on the sizes of query sets under both batch and adaptive modes. Sections 4 and 5 present upper bounds on the query set sizes under the batch and adaptive query models respectively. Section 6 presents results for the query set compaction problem where the goal is to remove redundant queries. Section 7 presents experimental results using both synthetic and real-world social networks. Conclusions and some directions for future work appear in Section 8.

2. Synchronous Dynamical Systems (SyDSs) and Query Model

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.

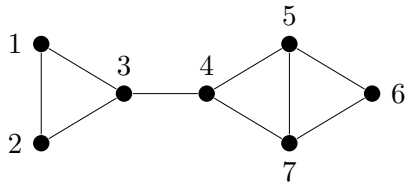
(i) **Threshold functions:** The local function f_v associated with node v of a SyDS $\mathcal{S}$ is a τ_v-**threshold** function for some integer $\tau_v \geq 0$ if the following condition holds: the value of f_v is 1 if the number of ones in the input to f_v is *at least* τ_v; otherwise, the value of the function is 0. Let d_v denote the degree of node v, and let τ_v denote the threshold of node v. The number of inputs to the function f_v is $d_v + 1$. Thus, we assume that $0 \leq \tau_v \leq d_v + 2$. (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 ones 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 ones in the input to the function is exactly i, $0 \leq i \leq k$. Note that each threshold function is also a symmetric function; however, symmetric functions are more general than threshold functions. For example, consider the following symmetric function f with five inputs: let the value of f be 1 when the number of ones in its input is (exactly) 1; otherwise, the value of f is 0. This is not a threshold

function since the value of the function is 1 when the number of ones in the input is 1 but changes to 0 when the number of ones in the input is larger than 1.

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(a). The thresholds are assigned in Figure 1(b). Also, we show an example trajectory (i.e., time sequence of configurations) of the system given that initially nodes 4, 6 and 7 are in state 1 and the rest are in state 0. Once the system reaches the configuration $\mathcal{C} = (1, 1, 1, 0, 0, 1, 0)$ at time step 3, it remains in that configuration forever; that is, $\mathcal{C}$ is a **fixed point** for this system. □



(a) Underlying network G

Node	1	2	3	4	5	6	7
Threshold	1	0	2	3	4	1	3

(b) Threshold assignments

	t_0	t_1	t_2	t_3
1	0	0	1	1
2	0	1	1	1
3	0	0	0	1
4	1	0	0	0
5	0	0	0	0
6	1	1	1	1
7	1	1	0	0

(c) Example trajectory

Figure 1: An example of a threshold 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 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. 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. As will be seen, for threshold SyDSs, the adaptive query mode can significantly decrease the number of queries. Figure 2 summarizes our active query framework.

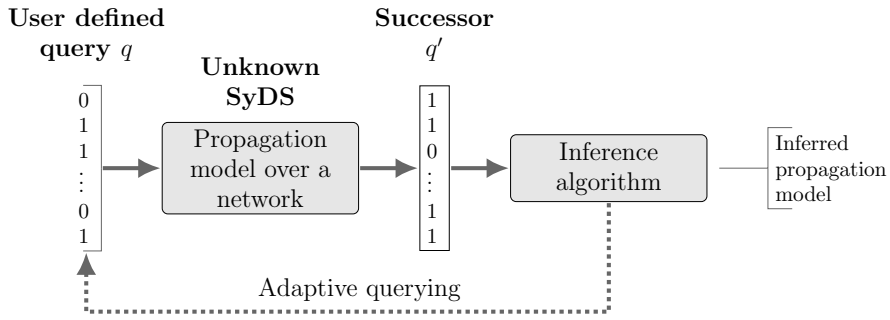


Figure 2: Active querying framework. The dashed line corresponds to active querying where the inference algorithm constructs the current query based on the response received from the system for the past queries. In batch querying, there is no such feedback.

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 ones in the input provided by q to the 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 2 *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. 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 3 *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: Under the batch mode, suppose a complete query set Q has *fewer than* Δ+2 queries. Since G has a node v of degree Δ, the number of ones 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 ones in the input to f_v is exactly k. This contradicts the assumption that Q is a complete query set. The proposition follows. ■

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 10 (Section 4) for all graphs.

Corollary 4 *For 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 5 *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, 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 is 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 5. ■

4. Querying Under the Batch Mode: Results for Symmetric Node Functions

In this section, we first present an algorithm for generating query sets under the batch mode for symmetric SyDSs and derive an upper bound on the optimal query set size. We then derive an asymptotically better bound using probabilistic methods. Finally, we present complexity results that suggest that in general, generating complete query sets of minimum size is computationally intractable.

4.1 Generating Query Sets Based on Node Coloring

4.1.1 Obtaining a Query Sequence From Node Coloring

Overview: 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 symmetric local functions and a variant of the node coloring problem for the underlying graph. We remind the reader that each query is a bit vector.

Definition 6 *(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 symmetric SyDS 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**.*

The following lemma points out a property of complete monotone query sequences.

Lemma 7 *Let $\mathcal{S}$ be a symmetric SyDS and let $M = \langle q_1, q_2, \ldots q_r \rangle$ be a complete monotone query sequence with 2 or more queries for $\mathcal{S}$. Then q_1 is the vector of all zeros and q_r is the vector with all ones.*

Proof: Suppose q_1 is not the vector with all zeros. Let v_i be a node such that the value assigned by q_1 to v_i is 1. Thus, score(q_1,v_i) ≥ 1. Since M is a monotone query sequence, score(q_j,v_i) ≥ 1 for $2 \leq j \leq r$. In other words, there is no query $q_j \in M$ for which score$(q_j, v_i) = 0$, contradicting the assumption that M covers v_i. Likewise, if q_r is not the vector with all ones, there is a node v_k such that the value assigned to v_k by q_r is 0. Thus, there is no query q_j in M such that score(q_j,v_k) = degree(v_k)+1. In other words, M does not cover v_k. ■

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 8
*(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 $\{1, 2, \ldots, k\}$ 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 follows: $\{u, v\} \in E'$ iff there is a path with at most 2 edges between u and v in G.*

We will also use the following result called Brooks' Theorem (West, 2001, Theorem 5.1.22).

Lemma 9 *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.* ■

Algorithm 1: Steps of Algorithm Alg-Monotone-Seq

Input: Graph $G(V, E)$ of a symmetric SyDS $\mathcal{S}$.
Output: A monotone complete query sequence M for $\mathcal{S}$.
1 Construct the graph $G^2(V, E')$.
2 Use the algorithm of Lemma 9 to 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 the above coloring step. (Color class C_j consists of all nodes assigned color j, $1 \leq j \leq k$.)
4 Create the query sequence $M = \langle q_0, q_1, \ldots, q_k \rangle$ with $k + 1$ queries as follows: query q_0 is a bit vector where every element is 0.
5 **for** $j = 1, 2, \ldots, k$ **do**
6 | 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.
7 **end**
8 Output the query sequence M.

Our algorithm Alg-Monotone-Seq for generating a monotone complete query sequence M for the given SyDS $\mathcal{S}$ is shown in Algorithm 1. 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 10 *Let $\mathcal{S}$ be a symmetric SyDS whose graph $G(V, E)$ has n nodes and maximum node degree Δ. Algorithm* Alg-Monotone-Seq *(Algorithm 1) 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 $\Delta(\Delta-1)+\Delta = \Delta^2$. Hence, by Lemma 9, G^2 can be colored using at most Δ^2+1 colors. Since G^2 has n nodes, n colors are always 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 zeros. 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) = ℓ. Query q_0 ensures that score(q,v) = 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) = ℓ. This completes the proof of Theorem 10. ■

4.1.2 Generating Coloring from a Monotone Complete Query Sequence

The above algorithm and Theorem 10 show that a monotone complete query sequence can be constructed from the coloring of the graph G^2. Theorem 11 shows that this relationship is not accidental; indeed, a valid coloring of G^2 can be generated from any monotone complete query sequence. This result is useful in proving that it is **NP**-hard to generate complete monotone query sequences of minimum length.

Theorem 11 *Let $G(V,E)$ be the underlying graph of a symmetric SyDS $\mathcal{S}$. 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, by Lemma 7, 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.

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 that 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.

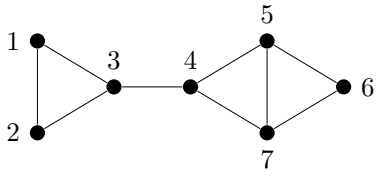
Case 1: The edge $\{v_i, v_j\}$ is in G.

Note that both v_i and v_j have color k. Let score(q_k,v_i) = 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(q_{k+1},v_i) $\geq \alpha + 2$. 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(q_k,v_x) = 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 edges in G, score(q_{k+1},v_x) $\geq \beta + 2$. 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 Theorem 11 follows. ■

Figure 3 gives an example of a batch query set generated using the distance-2 coloring method discussed above. The figure also includes a query set generated using an adaptive query algorithm discussed in Section 7.



(a) Underlying network G

Node	1	2	3	4	5	6	7
Threshold	1	0	2	3	4	1	3
Color	*a*	*b*	*c*	*d*	*a*	*b*	*c*

(b) Threshold assignments and G^2 coloring assignments

Color →	*a*	*b*	*c*	*d*
1	0	**1**	1	1	1
2	0	0	**1**	1	1
3	0	0	0	**1**	1
4	0	0	0	0	**1**
5	0	**1**	1	1	1
6	0	0	0	**1**	1
7	0	0	**1**	1	1

(c) Batch mode query set

	1		2		3		4		5	
1	0,4	0	0,2	0	0,1	0	0,1	**0**	1,1	0	1,1
2	0,4	1	0,2	0	0,1	0	0,1	0	0,0	0	0,0
3	0,5	**1**	0,2	1	2,2	1	2,2	0	2,2	1	2,2
4	0,5	0	2,5	**1**	3,5	**1**	4,5	0	4,5	**1**	4,4
5	0,5	0	2,5	1	3,4	1	4,4	1	4,4	1	4,4
6	0,4	**1**	0,2	0	1,1	0	1,1	1	1,1	0	1,1
7	0,5	1	2,5	0	3,4	1	3,4	**1**	3,3	1	3,3

(d) Adaptive querying

Figure 3: Query sets for the threshold SyDS of in Figure 1. In (c), the shaded rows correspond to nodes which have been assigned the color indicated by the column heading. In (d), the queries were generated adaptively using Algorithm 3 described in Section 7. There are five iterations and therefore, five queries. For each iteration, the corresponding query as well as the uncertainty (range) in the thresholds of the vertices are shown.

4.2 Generating query sets based on probabilistic methods

In Section 4.1, we showed that for any symmetric SyDS whose underlying graph has n nodes and maximum node degree Δ, one can obtain a complete query set with at most $\min\{\Delta^2+2, n+1\}$ queries. For some graphs with maximum node degree Δ, this method may generate a query set with $\Omega(\Delta^2)$ queries. Here, using probabilistic methods, we provide better bounds. First, using a simple sampling technique, we will show that there exists a query set of size at most $O(\Delta^{1.5}\log n)$ that is complete with high probability. This is asymptotically better than the previously established bounds for graphs where $\Delta \geq (\log n)^2$. Moreover, such a query set can be generated efficiently. Next, using more sophisticated techniques from Füredi and Kahn (1986) based on the Lovász Local Lemma, we show an upper bound of $O(\Delta(\log\Delta)^{2.5})$ on the size of complete query sets.

Theorem 12 *Let $\mathcal{S}$ be a symmetric SyDS with graph $G(V,E)$ where $|V| = n$ and maximum node degree $= \Delta$. 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 $\mathcal{S}$.*

The proof of the above theorem appears in Appendix A. Here, we provide the general idea for the query set construction method used in proving the above theorem. Let $\mathbb{D}(p)$ be a probability distribution defined on the set of configurations of a SyDS such that the state of each vertex is set to 1 independently with probability p. For a query q, we use the notation $q \sim \mathbb{D}(p)$ to mean that query q is drawn from the distribution $\mathbb{D}(p)$. We will generate a query set $Q = \left\{q_{ij} \sim \mathbb{D}\left(\frac{i}{\Delta+1}\right) \mid 1 \leq i \leq \Delta,\ 1 \leq j \leq r\right\}$ for some number of repetitions r. This construction enables us to prove that for any vertex v and positive integer $k \leq \Delta+1$, with high probability, there is a query $q_{ik} \in Q$ for which $\text{score}(q_{ik}, v) = k$. The reader is referred to Appendix A for details.

Now, we show an upper bound of $O(\Delta(\log\Delta)^{2.5})$ on the size of complete query sets under the batch mode for SyDSs with symmetric local functions.

Theorem 13 *Let $G(V,E)$ be the underlying graph of a symmetric SyDS $\mathcal{S}$. Let Δ denote the maximum node degree in G. Under the batch mode, for any $\Delta \geq 2$, there is a complete query set Q for $\mathcal{S}$ with $|Q| = O\left(\Delta(\log\Delta)^{2.5}\right)$.*

The following lemma is a key ingredient for the proof of Theorem 13. Suppose $G(V,E)$ is a graph and $V' \subseteq V$. We use $N[v,V']$ to denote the closed neighborhood of v restricted to V'; that is, $N[v,V'] = N[v] \cap V'$. The following notation is used in our proof of the lemma. Given a positive real number x, we let $\langle x \rangle$ to denote the integer obtained by rounding x to the nearest integer; that is, $\langle x \rangle = \lfloor x \rfloor$ if the fractional part of x is *less than* 0.5; otherwise, $\langle x \rangle = \lceil x \rceil$.

Lemma 14 *Let $G(V,E)$ be the underlying graph of a symmetric SyDS. Let $V' \subseteq V$ such that $\forall v \in V$, $|N[v,V']| \leq \ell$. Then, there exists a set Q with at most $22\,\ell^{3/2}\log|V'| + 2$ queries such that (i) $\forall v \in V \setminus V'$ and $q \in Q$, $q(v) = 0$ and (ii) $\forall v \in V$ and every $i \in \{0,1,\ldots,|N[v,V']|\}$, there exists $q \in Q$ such that $\text{score}(v,q) = i$.*

Proof: The all zeros query $\mathbf{0}$ is such that $\forall v \in V$, $\text{score}(v,\mathbf{0}) = 0$. The configuration q with all vertices of V' in state 1 yields $\forall v \in V$, $\text{score}(v,q) = |N[v,V']|$. These two queries

account for the additive term of 2. Let $\mathbb{D}(p, V')$ be the distribution where each $q \sim \mathbb{D}(p, V')$ is constructed as follows: $\forall v \in V'$, $\Pr(q(v) = 1) = p$ and $\forall v \in V \setminus V'$, $q(v) = 0$. Let $Q = \left\{q_{ij} \sim \mathbb{D}\left(\frac{i}{\ell}, V'\right) \mid 1 \leq i < \ell,\ 1 \leq j \leq 22\,\ell\sqrt{\ell+1}\log|V'|\right\}$. Let $d'[v] = |N[v, V']|$. For any $b \in \{1, \ldots, d'[v]\}$ and $q \sim \mathbb{D}\big(z, V'\big)$, where $z = \left\langle \frac{b\ell}{d'[v]} \right\rangle$,

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

where the second inequality follows from Lemma 27 which is given in Appendix A.

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

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

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

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

We also use the following two lemmas of Füredi and Kahn (Füredi and Kahn, 1986) that are based on the Lovász Local Lemma (Mitzenmacher and Upfal, 2005).

Lemma 15 *Let $\mathcal{H}(V, E_{\mathcal{H}})$ be a hypergraph on a set of n elements V such that each hyperedge has at most b elements and each element belongs to at most b hyperedges, where $b \geq 500$. Then, V can be partitioned into $\alpha = \left\lceil \frac{b}{\log b} \right\rceil$ sets $X_1, X_2, \ldots, X_\alpha$ of V such that $|H \cap X_i| \leq \lceil 4.7\log b \rceil$ for all $H \in E_{\mathcal{H}}$.*

Lemma 16 *Let $G(V, E)$ be a graph with maximum node degree Δ. Let $V' \subseteq V$ such that $\forall v \in V$, $|N[v, V']| \leq \ell$. Then, V' can be partitioned into $k \leq (\ell-1)\Delta+1$ sets $V'_1, \ldots, V'_k$ such that $\forall v$, $|N[v, V'] \cap V'_j| \leq 1$ for every block V'_j.*

Proof of Theorem 13: We will prove that for any $\Delta \geq 2$, the query set size is at most $2500\Delta(\log\Delta)^{2.5}$. We first note that Lemma 15 has a technical requirement that b be at least 500. In order to apply that result, we require $\Delta \geq 500$. Hence, we first consider the case $\Delta \leq 500$ separately. From Theorem 10, we have $|Q| \leq \Delta^2 + 2$. For any constant $c \geq 42$, it can be verified that $|Q| \leq \Delta^2 + 2 \leq c\Delta(\log\Delta)^{2.5}$ for the entire range $2 \leq \Delta \leq 500$. Therefore, the bound holds trivially for this case. So, we will assume for the remainder of this proof that $\Delta > 500$.

For any graph G, let $\mathcal{H}$ be the hypergraph where each hyperedge H_v corresponds to the closed neighborhood of vertex v. Since the maximum degree is Δ, for all v, $|H_v| \leq \Delta + 1$

and v belongs to at most $\Delta+1$ hyperedges. By Lemma 15, for any $\Delta \geq 500$, the vertices of G can be partitioned into $\alpha \leq \left\lceil \frac{\Delta+1}{\log(\Delta+1)} \right\rceil$ subsets $X_1, X_2, \ldots, X_\alpha$, such that every vertex is adjacent to at most $\ell = \lceil 4.7 \log(\Delta+1) \rceil$ vertices in any X_i. For $1 \leq i \leq \alpha$, let $n_{\leq i}(v)$ denote the number of neighbors of v in $\bigcup_{j \leq i} X_j$. The query set Q is structured in the following manner. It is partitioned into α subsets $Q_1, Q_2, \ldots, Q_\alpha$ such that Q_i determines $f_v(b)$, $b = n_{\leq i-1}(v)+1, \ldots, n_{\leq i}(v)$ for each v, where $n_{\leq 0}(v) = 0$ by definition. In the remaining part, we will show that this can be achieved with $|Q_i| \leq 22\sqrt{\Delta}\log^2\Delta + 2$ for each i.

We will partition each X_i into subsets $X_{i1}, X_{i2}, \ldots, X_{ik}$ such that every vertex in V is adjacent to at most one vertex in X_{ij} for any j. Since $\forall v \in V$, $|N[v, X_i]| \leq \ell$, by setting $V' = X_i$ in Lemma 16, this can be achieved for $k \leq (\ell-1)\Delta+1$. Now, we construct an auxiliary graph $\widehat{G}$ with vertex set $\widehat{V} = V \cup \{x_{ij} \mid j = 1, \ldots, k\}$ where each x_{ij} corresponds to X_{ij}. The edge set $\widehat{E}$ contains edges from V to $\{x_{ij} \mid 1 \leq j \leq k\}$ where a vertex v is adjacent to x_{ij} if and only if it has a neighbor in X_{ij} in G. The vertex functions $f_v(\cdot)$ remain the same $\forall v \in V$. Applying Lemma 14 to $\widehat{G}$ with $V' = \{x_{ij} \mid 1 \leq j \leq k\}$, since $\forall v \in V$, $|N[v, X_i]| \leq \ell$, we note that there exist at most $22\ell^2\sqrt{\ell}\log k + 2 \leq 23\ell^2\sqrt{\ell}\log(\ell\Delta)$ queries $\widehat{q}$ such that $\forall v \in V$, $\widehat{q}(v) = 0$ and for each $b = 0, 1, ..., |N[v, X_i]|$, there exists a query $\widehat{q}$ such that v is adjacent to exactly b vertices in $\{x_{ij} \mid 1 \leq j \leq k\}$. Let this set of configurations be denoted by $\widehat{Q}_i$.

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

Thus, $|Q| \leq \alpha|Q_i| \leq \frac{2\Delta}{\log\Delta} 23\ell^2\sqrt{\ell}\log(\ell\Delta) < 2500\Delta(\log\Delta)^{2.5}$, where the last inequality is due to the fact that $\ell = 4.7\lceil\log(\Delta+1)\rceil$. Theorem 13 follows. ■

4.3 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 symmetric SyDS $\mathcal{S}$ and a positive integer k.

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

Theorem 17 *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) even for planar graphs (Lloyd and Ramanathan, 1992).

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 10 that when G^2 is r-colorable, there is a monotone complete query sequence with $k = r+1$ queries. Also, from Theorem 11, 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. ■

4.4 Generating Concise Query Sets: A Generalized Version

We now consider the complexity of a more general version of the problem of generating concise complete query sets under the batch model. Recall that for a symmetric SyDS, for each node v and each integer ℓ in {0, 1, ..., degree(v)+1}, a complete query set Q must have a query q such that score(q,v) = ℓ. In the version of the problem considered below, we will only require the query set to provide a small subset of score values. This problem can be formulated as follows.

Concise Query Set for Specified Scores (CQS-SS)

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

Question: Is there a query set Q with at most k queries for $\mathcal{S}$ such that for each node v and each score value $s \in L$, there is a query $q \in Q$ for which score(q,v) = s?

We now show the CQS-SS problem is **NP**-complete even when the set L of scores contains only one value and the underlying graph is highly restricted (e.g., graphs of maximum node degree 4, planar graphs). To prove this result, we use a reduction from the **One-In-Three 3SAT** (OIT-3SAT) problem which is defined as follows: given a set of Boolean variables $X = \{x_1, x_2, \ldots, x_n\}$ and a collection $C = \{C_1, C_2, \ldots, C_m\}$ of clauses where each clause C_j is a disjunction of *exactly* three variables from X, is there a truth assignment to the variables such that *exactly* one variable is set to true in each clause? In the definition of OIT-3SAT, note that none of the clauses contains negated literals. It is well known that OIT-3SAT is **NP**-complete (Garey and Johnson, 1979). The **NP**-completeness holds even under some restrictions as indicated in the following theorem.

Theorem 18 *OIT-3SAT is **NP**-complete even when either of the following conditions hold.*

(a) Each variable occurs in exactly three clauses (Schmidt, 2010).

*(b) Consider the following graph $H(V_H, E_H)$ constructed from an OIT-3SAT instance. V_H contains one node for each variable in X and one node for each clause in C. For a variable x_i and clause C_j, there is an edge between the corresponding nodes if x_i appears in C_j. OIT-3SAT is **NP**-hard even when graph H is planar (Hunt III et al., 1998; Mulzer and Rote, 2008).* ■

Theorem 19 *Problem* CQS-SS *is **NP**-complete even when set of scores has only one value. The problem remains **NP**-complete even when the underlying graph is constrained to belong to one of the following classes: (a) graphs with maximum node degree 4 or (b) planar graphs.*

Proof: It is easy to see that CQS-SS is in **NP**. To prove **NP**-hardness, we use a reduction from OIT-3SAT defined above. Consider an instance of OIT-3SAT consisting of the variable set X and the clause set C. We construct an instance of the CQS-SS problem as follows. We first describe the construction of the underlying graph $G(V, E)$.

1. For each variable x_i, V has a node v_i, $1 \leq i \leq n$. For each clause C_j, V has a node w_j, $1 \leq j \leq m$.
2. For each variable x_i and clause C_j, if x_i occurs in C_j, then E contains the edge $\{v_i, w_j\}$, $1 \leq i \leq n$ and $1 \leq j \leq m$.
3. For each node v_i, G has another node z_i and the edge $\{v_i, z_i\}$.
4. For each node w_j, G has 4 additional nodes, denoted by y_j^1, y_j^2, y_j^3 and y_j^4, and the following four edges: $\{w_j, y_j^1\}$, $\{y_j^1, y_j^2\}$, $\{y_j^1, y_j^4\}$, $\{y_j^2, y_j^3\}$.

An example of this graph construction is shown in Figure 4.

Variable set $X = \{x_1, x_2, x_3, x_4, x_5\}$
Clause set $C = \{C_1, C_2\}$, where $C_1 = \{x_1, x_2, x_3\}$ and $C_2 = \{x_3, x_4, x_5\}$.

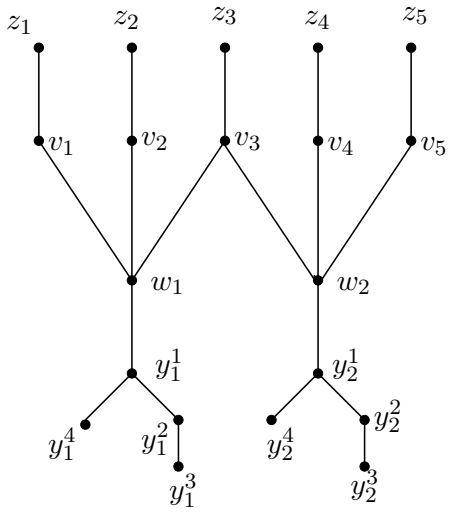


Figure 4: Example to illustrate the reduction from OIT-3SAT to CQS-SS.

The local function at each node is symmetric. The set L of scores contains just one value, namely 1. The number of queries k is set to 1. This completes the construction of the CQS-SS instance, and it can be seen that the construction can be done in polynomial time. We now prove that the resulting CQS-SS instance has a solution iff the OIT-3SAT instance has a solution.

Suppose the OIT-3SAT instance has a solution. We construct the query q which provides a score of 1 to each node of G as follows. Note that G has a total of $N = 2n + 5m$ nodes. For each node v of G, we will use the notation $q(v)$ to denote the value of v in q.

(a) Consider each variable x_i, $1 \leq i \leq n$. If it is set to true in the solution to OIT-3SAT, set $q(v_i)$ to 1 and $q(z_i)$ to 0; otherwise, set $q(v_i)$ to 0 and $q(z_i)$ to 1.

(b) For each clause C_j, $1 \leq j \leq m$, set $q(w_j)$ to 0.

(c) For each clause C_j, $1 \leq j \leq m$, the values in q for the nodes y_j^1 through y_j^4 are chosen as follows: $q(y_j^1) = q(y_j^2) = 0$ and $q(y_j^3) = q(y_j^4) = 1$.

It is easy to verify that for each $v \in V$, score$(q,v) = 1$. In other words, we have a solution to the CQS-SS instance.

Now suppose the CQS-SS instance has a solution. Let q be the corresponding query. We have the following claim.

Claim 1: (a) For $1 \leq j \leq m$, $q(w_j) = 0$. (b) For $1 \leq j \leq m$, $q(y_j^1) = 0$.

Proof of Claim 1:

Part (a): Suppose $q(w_j) = 1$ for some j, $1 \leq j \leq m$. Then since $\{w_j, y_j^1\}$ is an edge in E and q provides a score of 1 to each node, we must have $q(y_j^1) = 0$. Now, since y_j^1 is the only neighbor of y_j^4 and score$(q,y_j^4) = 1$, we must have $q(y_j^4) = 1$. Now, however, score$(q,y_j^1) > 2$ (since $q(w_j) = 1$ and $q(y_j^4) = 1$). This contradicts the validity of q and establishes Part (a) of Claim 1.

Part (b): Suppose $q(y_j^1) = 1$ for some j, $1 \leq j \leq m$. Then since $\{y_j^1, y_j^2\}$ is an edge in E and q provides a score of 1 to each node, we must have $q(y_j^2) = 0$. Now, since y_j^2 is the only neighbor of y_j^3 and score$(q,y_j^3) = 1$, we must have $q(y_j^3) = 1$. Now, however, score$(q,y_j^2) = 2$ (since $q(y_j^1) = 1$ and $q(y_j^3) = 1$). This contradicts the validity of q and establishes Part (b) of Claim 1.

We now continue with the main proof. We choose an assignment to the variables of X as follows. Consider each variable x_i, $1 \leq i \leq n$. If $q(v_i) = 1$, set x_i to true; otherwise, set x_i to false. To prove that this is a solution to the OIT-3SAT instance, we need to prove that each clause has exactly one variable set to true. Consider any clause C_j and the corresponding node w_j of G. We know that score$(q,w_j) = 1$. Since $q(w_j) = 0$ and $q(y_j^1) = 0$ (Claim 1), the q provides a score of 1 to w_j, there is exactly one node v_r such that $q(v_r) = 1$ and the variable x_r appears in clause C_j. Since we set x_r to true, exactly one of the variables in C_j is set to true. In other words, we have a solution to the OIT-3SAT instance. This completes the proof of **NP**-hardness of CQS-SS for general graphs.

To obtain the **NP**-hardness of CQS-SS for graphs with maximum node degree of 4, we carry out the above reduction from the version of OIT-3SAT in which each variable occurs in exactly three clauses (Part (a) of Theorem 18). To obtain the **NP**-hardness of CQS-SS for planar graphs, we use the a reduction from the version of OIT-3SAT mentioned in Part (b) of Theorem 18 and note that the additional vertices and edges used in the construction do not affect planarity. ■

5. Query Sets Under the Adaptive Mode: Results for Threshold Functions

We now consider the adaptive mode of queries. We show that 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. 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. Note that the threshold of v_1 can vary from 0 to $n+1$. We first try a query q for which score(q,v_1) $= \lceil n/2 \rceil$. Depending on the response to the query, the range of possible threshold values for v_1 changes to either 0 through $\lceil n/2 \rceil - 1$ or $\lceil n/2 \rceil + 1$ through $n+1$. When this is repeated, after $O(\log n)$ queries under the adaptive model, we can identify the threshold of v_1. After this, the following 3 additional queries are sufficient to identify the thresholds of the remaining $n-1$ nodes: a query with all zeros, a second query with all ones 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 using $O(\log n)$ queries under the adaptive mode. The following proposition summarizes the above discussion.

Proposition 20 *For a SyDS where the underlying graph is the star graph with n nodes and each local function is a threshold function, $O(\log n)$ queries are sufficient under the adaptive query mode to determine all the local functions.* ■

Since the center node of a star graph with n nodes has a degree of $n-1$, its threshold can be any one of the $n+1$ values in $\{0, 1, \ldots, n+1\}$. Thus, a simple information theoretic argument points out that $\Omega(\log n)$ is a lower bound on the number of adaptive queries needed to infer the threshold values of all the nodes of a star graph. Thus, the upper bound given by Proposition 20 is tight to within a constant factor.

As was shown in Section 3, in the batch mode, $n+1$ queries are sometimes necessary to identify all the thresholds. Thus, the adaptive query mode can reduce the number of queries significantly.

The above idea can be applied to a more general class of graphs. Let α and β be nonnegative integers. We say that a graph is (α, β)-**simple**, if α nodes have degree $> \beta$ and the remaining $n - \alpha$ nodes have a degree of at most β. For example, any star graph on n nodes is a $(1, 1)$-simple graph. Trivially, every graph is a $(0, n-1)$-simple graph. However, when α and β are required to be constants independent of n, a clique on n nodes is *not* an (α, β)-simple graph. Under the adaptive query model, we can obtain small query sets for (α, β)-simple graphs when α and β are small. In what follows, we show that the number of adaptive queries required for an (α, β)-simple graph is at most $\alpha(\lceil \log n \rceil + 1) + \beta^2 + 2$. When α and β are constants, this bound is $O(\log n)$.

Theorem 21 *For any threshold SyDS whose underlying graph G belongs to the class of (α, β)-simple graphs, $\alpha(\lceil \log n \rceil + 1) + \beta^2 + 2$ queries are sufficient in the adaptive mode to identify all the threshold values. In particular, if α and β are constants independent of n, the number of queries used in this approach is $O(\log n)$.*

Proof: We present an approach that uses at most $\alpha(\lceil \log n \rceil + 1) + \beta^2 + 2$ queries under the adaptive query mode. To discuss this approach, let V_0 denote the subset of nodes of G such that each node in V_0 has a degree larger than β. Thus, $|V_0| = \alpha$. Let V_1 denote the set of remaining $n - \alpha$ nodes (each of which has a degree of at most β). The steps of our method are as follows.

1. We first construct a set Q_1 of queries which are submitted in a single batch to identify the thresholds of all the nodes in V_1 as follows.
   (a) Let G_1 be the subgraph of G induced on V_1. Construct G_1^2, the square of G_1.
   (b) Construct a (node) coloring of G_1^2. Since the degree of each node in G_1 is at most β, the maximum node degree in G_1^2 is at most β^2. Hence, by Lemma 9, G_1^2 can be efficiently colored using at most $k = \beta^2 + 1$ colors. Let C_1, C_2, ..., C_k denote the color classes generated in this step.
   (c) Choose an arbitrary order, say v_{i_1}, v_{i_1}, ..., v_{i_α}, for the nodes in V_0. The first $\alpha + 1$ queries of Q_1, denoted by Q_1^j, $0 \leq j \leq \alpha$, are chosen as follows. In Q_1^0, all the nodes have the value 0. In Q_1^j, $1 \leq j \leq \alpha$, nodes v_{i_1} through v_{i_j} have the value 1 and all other nodes have the value 0.
   (d) The remaining k queries of Q_1, denoted by Q_1^j, $\alpha + 1 \leq j \leq \alpha + k$, are chosen as follows. For all these queries, the values for all the α nodes of V_0 are 1. The values for the nodes in V_1 are chosen as follows. In Q_1^j, all the nodes in color classes C_1 through C_j have the value 1 and all the remaining nodes of V_1 have the value 0, $1 \leq j \leq k$.
   (e) The total number of queries in Q_1 generated in Steps 1(c) and 1(d) is $\leq \alpha + 1 + \beta^2 + 1 = \alpha + \beta^2 + 2$.
2. Recall that $|V_0| = \alpha$. We use the adaptive mode for the nodes in V_0; that is, we use a separate binary search for each of the α nodes in V_0. For each node $v \in V_0$ with degree d_v, the batch queries in Q_1 include a query with scores 0 and $d_v + 1$. Therefore, the binary search needs to consider only scores 1 through d_v for v. This uses at most $\lceil \log (d_v + 1) \rceil$ queries for v; this number is at most $\lceil \log n \rceil$ since $d_v < n$. Thus, for all the α nodes in V_0, the number of adaptive queries is at most $\alpha \lceil \log n \rceil$. Q_0 denote the set of queries used in this step.

As argued above, the queries in Q_0 identify the thresholds of all the nodes in V_0. We now argue that the resulting query set Q_1 covers all the nodes of V_1. To see this, consider any node $v \in V_1$, and let d_v be the degree of V in G. Thus, we need to show that for every value $\gamma \in \{0, 1, \ldots, d_v + 1\}$, there is a query $q \in Q_1$ such that score(q,v) = γ. Let r_0 and r_1 denote the number of neighbors of v in V_0 and V_1 respectively. Thus, $d_v = r_0 + r_1$. For $0 \leq j \leq \alpha$, in query Q_1^j, exactly j nodes from V_0 are 1 and all the other nodes are 0. Thus, for each value γ, where $0 \leq \gamma \leq r_0$, there is a query $q \in Q_1$ such that score(q,v) = γ. Now consider any value γ in the range $r_0 + 1$ through $d_v = r_0 + r_1$. Let $N^1[v]$ denote the closed neighborhood of v in G_1. Note that $|N^1[v]| = r_1 + 1$. Because of the coloring of G_1^2, no two nodes in $N^1[v]$ may appear in the same color class. Let $C_{p_1} < C_{p_2} < \ldots < C_{p_{r_1+1}}$ denote the color classes in which nodes of $N^1[v]$ appear. It can be seen that for $1 \leq \ell \leq r_0 + r_1 + 1$,

query Q_1^ℓ is such that score(Q_1^ℓ,v) = ℓ. Thus, Q_1 covers all the nodes in V_1, and this completes the proof.

From the above discussion, $|Q_0| \leq \alpha\lceil \log n \rceil$ and $|Q_1| \leq \alpha + \beta^2 + 2$. Thus, the total number of queries used by the above method is at most $\alpha(\lceil \log n \rceil + 1) + \beta^2 + 2$. This completes the proof of Theorem 21. ■

The above result can be used to establish an upper bound on the number of queries under the adaptive mode for scale-free graphs. Following Easley and Kleinberg (2010), we say that a graph with n nodes and maximum degree Δ is **scale-free with exponent** γ if there are constants $c' > 0$ and $\gamma > 1$ such that for any integer d, $0 \leq d \leq \Delta$, the fraction of nodes with degree d in G is given by c'/d^γ. Our upper bound for the number of queries for scale-free graphs is stated below.

Theorem 22 *For a threshold SyDS whose underlying graph $G(V, E)$ is scale-free with exponent $\gamma > 1$, the thresholds can be found using $O\Big([n \log n]^{\frac{2}{\gamma+1}}\Big)$ queries under the adaptive query mode.*

Proof:

For a positive integer β (to be chosen), let $V_0 \subseteq V$ denote the set of nodes of G with degree $> \beta$ and let $V_1 \subseteq V$ denote the remaining nodes (each of which has degree at most β). By Theorem 21, the number of queries $g(\beta)$ required satisfies $g(\beta) \leq |V_0|(\lceil \log n \rceil + 1) + \beta^2 + 2$. Since $|V_0|$ is the number of nodes with degree $\geq \beta + 1$, we have

$$|V_0| \leq \sum_{x=\beta+1}^{\Delta} c' \frac{n}{x^\gamma} \leq c'' \int_\beta^\infty \frac{n}{x^\gamma} dx = c \frac{n}{\beta^{\gamma-1}},$$

for some constants c, c' and c''. Therefore, $g(\beta) = c\frac{n}{\beta^{\gamma-1}} \log n + \beta^2 + 2$. 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 $\beta^{\gamma+1} = \Theta(n \log n)$ or $\beta = \Theta\Big([n \log n]^{\frac{1}{\gamma+1}}\Big)$. For this value, $g(\beta) = O\big([n \log n]^{\frac{2}{\gamma+1}}\big)$, and this completes our proof of Theorem 22. ■

6. Query Set Compaction

Overview: Some methods for generating queries in the batch mode (e.g., the randomized query generation method discussed in Section 4.2) may a generate complete query set Q with *redundant* queries. Since our goal is to construct query sets of minimum size, it is of interest to consider the problem of eliminating such redundant queries while preserving the property of completeness (i.e., the query set can correctly identify all the local functions). We call this the **query set compaction** problem; a precise formulation appears below. We show that it is **NP**-hard to obtain a performance guarantee of $o(\log n)$ for this problem, where n is the number of nodes of a given SyDS. We also present an approximation algorithm that provides a performance guarantee of $O(\log n)$ for the problem.

6.1 Problem Definition

A precise formulation of the query set compaction problem 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 for $\mathcal{S}$ 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 $\mathcal{S}$?

Our hardness results for QSC rely 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 $\mathbf{P} = \mathbf{NP}$, it cannot be approximated to within the factor $(1-\epsilon)\ln(n)$, for any ϵ, $0 < \epsilon < 1$, where n is the size of the base set (Vazirani, 2001).

6.2 Complexity Results for QSC

Theorem 23 *(a) The problem QSC is **NP**-complete even when the underlying graph has no edges. (b) Unless $\mathbf{P} = \mathbf{NP}$, QSC cannot be approximated to within the factor $o(\log n)$ in polynomial time, where n is the number of nodes in the underlying graph of the SyDS.*

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 ones 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_{m+1},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; thus, 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(q_{m+1}, v_{n+1}) is 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 a $2\rho = o(\log n)$ 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

$$\text{OPT(QSC)} \;\leq\; \text{OPT(MSC)} + 1. \tag{2}$$

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 $\rho\,\mathrm{OPT(QSC)}$ queries. From this query set, it was shown in the **NP**-hardness proof that a solution to the MSC instance with $\rho\,\mathrm{OPT(QSC)} - 1$ subsets can be constructed. Letting APPROX(MSC) denote the resulting number of subsets, we have

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

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. This completes the proof of Theorem 23. ■

6.3 An Approximation Algorithm for QSC

To complement the non-approximability result of the previous section, we will present an efficient approximation algorithm with a performance guarantee of $O(\log n)$ for the QSC problem. The basic idea is to use a reduction from the QSC problem to the MSC problem and then to use a known (greedy) approximation algorithm (Vazirani, 2001) for the MSC problem.

Algorithm 2: Steps of Algorithm Approx-QSC

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 a good approximation for a complete query set of minimum size.

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 $\mathrm{score}(q_j, v_i) = k$, 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'.

The steps of our approximation algorithm Approx-QSC for QSC are shown in Algorithm 2. We assume that an instance of the QSC problem is specified by an underlying graph $G(V, E)$ and a complete query set Q. Let $V = \{v_1, v_2, \ldots, v_n\}$ and $Q = \{q_1, q_2, \ldots, q_m\}$. Note that each query $q_j \in Q$ is an n-bit vector, with bit i specifying the value of node v_i, $1 \leq i \leq n$. Let d_i denote the degree of node v_i in G, $1 \leq i \leq n$. Steps 1 and 2 of the algorithm construct an instance of the MSC problem consisting of the base set X and a collection $Y = \{Y_1, Y_2, \ldots, Y_m\}$ of subsets of X. Each subset Y_j in Y is constructed from query $q_j \in Q$, $1 \leq j \leq m$. Step 3 applies a well known greedy algorithm (which, in each

iteration adds a subset Y_j from Y such that Y_j contains the maximum number of elements of X which have not yet been covered) to construct an approximate solution Y' to the MSC problem. The final step chooses a query corresponding to each subset in Y' and outputs the resulting query subset Q'. It can be seen that the approximation algorithm runs in polynomial time. To establish the performance guarantee provided by Approx-QSC, we need the following lemma.

Lemma 24 *The reduction from QSC to MSC used in Steps 1 and 2 of Approx-QSC (Algorithm 2) 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 $(1 \leq i \leq n)$ and any integer k, $0 \leq k \leq d_i$, there is a query $q \in Q'$ such that $\text{score}(q, v_i) = k$. 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 $\text{score}(q_{j_z}, v_i) = k$.

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 $\text{score}(q_{j_z}, v_i) = k$. 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 25 *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 establish the performance guarantee provided by Approx-QSC.

Theorem 26 *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.*

Proof: 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. We observe that the size of the base set X is $2|E| + n$. To see this, note that for each node $v_i \in V$ with degree d_i, the the number of elements added to X in Step 1 of the algorithm in Algorithm 2 is $d_i + 1$. Therefore, $|X| = \sum_{i=1}^{n}(d_i + 1) = (\sum_{i=1}^{n} d_i) + n = 2|E| + n$ since the sum of all the node degrees is $2|E|$. Since $|E| < n^2$, $|X| < 3n^2$. The greedy algorithm for MSC provides a performance guarantee of $O(\log |X|)$, which is $O(\log n)$ since $|X| < 3n^2$. Thus, the approximation algorithm for MSC produces a solution with at most $O(\log n)\,\text{OPT(MSC)}$ sets. By Lemma 24, 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 $O(\log n)\,\text{OPT(MSC)}$ which is equal to $O(\log n)\,\text{OPT(QSC)}$ by Observation 25. Thus, Approx-QSC has a performance guarantee of $O(\log n)$. ■

7. Experimental Results

7.1 Overview

The theoretical results of the previous sections establish that finding a smallest set of queries to infer the local functions of threshold or symmetric SyDSs is non-trivial. In this section, through extensive experimentation[2] on more than 20 real-world and synthetic networks, we seek to answer the following questions.

1. We study the performance of the proposed algorithms for batch mode. How do the the resulting query sets compare with the lower bounds and thus the optimal query set sizes for these networks? We studied an approach based on coloring G^2 for inferring threshold and symmetric SyDS.

2. While we were able to develop algorithms for batch mode with provable performance guarantees, coming up with a technique for adaptive mode for general graphs turns out to be hard. We propose a greedy heuristic inspired by distance-two coloring to infer thresholds of a SyDS (namely, Algorithm 3 presented in Section 7.4). How does this heuristic perform relative to the developed lower bounds? We experimentally analyze this question.

3. The bounds established above are in terms of maximum degree or the size of the graph. However, many real-world networks are scale-free, having a low average degree, but a very high maximum degree. How do our algorithms perform on these networks? Does network density (number of edges) have any role in determining the query set size?

4. We note that the query set size depends not only on the network structure, but also on threshold assignments. How does the uncertainty in the range of values the threshold can take influence the query set size? Our experiments also study this question.

Networks and threshold assignment. The diverse real-world and synthetic networks considered in this work are listed in Table 1 along with some of their properties. As indicated in that table, all the mined networks used in our experiments are from the SNAP library (Leskovec and Krevl, 2014). We present representative results for selected networks, with other networks exhibiting the same behavior unless stated otherwise. We assigned thresholds 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 $\big[(d(v)+2)(1-\theta)/2, (d(v)+2)(1+\theta)/2\big]$. 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.

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. Details regarding the computing environment used for our experiments are presented in Appendix B which also includes plots showing wall clock times for generating queries.

2. The code is available in `https://github.com/NSSAC/active_queries_threshold_gds_published_code.git`.

Table 1: Networks used in our experiments, their properties, and results of the different algorithms for inferring local functions or thresholds. The first 13 networks listed are real world networks from the SNAP library (Leskovec and Krevl, 2014). They are grouped by type: social online, friendship, co-authorship (collaboration). The last four are 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.** $d_{\mathbf{avg}}$	**max. deg.** Δ	**Spec. rad.** $\lambda_{\max}$	**Meth. 1** $n_c(G^2)+1$	**Meth. 2** $t(v)=\frac{d(v)+2}{2}$
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. 5(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}, $n_c(G^2) = 1000$.

7.2 Method 1: Approach Based on Coloring G^2

We studied the performance of Alg-Monotone-Seq (Algorithm 1). For most real world networks considered in this paper, this algorithm gives the best possible performance of $\Delta + 2$ (Proposition 3). This is due to the fact that in all these cases, $n_c(G^2)$, the number of colors used to color G^2 is equal to $\Delta + 1$, the lower bound on the number of required colors. Compare the values in Table 1 for columns "max. deg. Δ" and "Results: Query set size, Method 1." For synthetic networks (random regular and Erdős-Rény graphs) though, $n_c(G^2)$ is significantly higher than $\Delta + 2$, yet much lower than $\Delta^2 + 2$, the upper bound (Theorem 10). 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 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), where $\lambda_{\max}$ is the largest eigenvalue of the adjacency matrix of G. Again, see Table 1 and "$\lambda_{\max}$." 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 $\lambda_{\max}^2 + 1$ 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 proper subset of queries can be complete. To see this, suppose this set is not compact. Then, 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 state 1. Suppose query q_k is not required for the set to be complete. Then, using the arguments in the proof of Theorem 11, it can be seen 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 is a contradiction.

7.3 Experiments on Query Compaction

Here, we present some experimental results obtained by first generating query sets using the probabilistic method discussed in Section 4.2 and then applying the query set compaction heuristic presented in Section 6.3. To generate the complete query sets, we used the method of Theorem 12. Recall that 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 $1 \leq i \leq \Delta$. The query set generation process may have to be repeated a number of times to obtain a complete set. 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 whether each of them is a complete set. For $\ell = 2$, out of the 50 sets constructed, none of them was 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

Table 2: Results of randomized query generation with compaction.

Network	Query set size	% Compaction
FB	407	81
p2p-gnutella04	159	84
Enron	2306	83
Wikipedia	1420	86
ca-astroph	899	82
ca-condmat	393	85
ca-grqc	153	81
ca-hepph	1201	75
ca-hepth	140	78
cit-hepph	1240	85
Rand. reg. A	$\approx 5\Delta$	40

compaction ratio depends on the size of complete set which was given as input. We note that compaction of query sets generated by this method yields around 80% reduction in the size of the query set in most cases (Table 2). On the average, the combination of randomized algorithm and compaction gives query sets of size around 1.5 to 2 times that of the monotone query sequence method discussed in Section 4.1. However, the method of random query set generation followed by compaction is comparatively much easier to implement.

7.4 Method 2: An Adaptive Algorithm

While the previous methods are for the batch mode, here we develop an adaptive algorithm to infer the thresholds. We give an outline of the approach. The steps of the algorithm are shown in Algorithm 3. In this algorithm, we use the notation $N[v, G^2]$ to denote the set of nodes of v that are at distance of at most 2 in G (including v). We note that $N[v, G^2]$ can be computed by a two-step breadth-first search of G starting at v.

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 $\text{score}(q, v)$ falls in the range $[t_L(v), t_H(v)-1]$, the 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 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. Initially, all the nodes are in state 0. We first choose a 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 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

Algorithm 3: Greedy heuristic to infer the thresholds.

Data: Network $G(V, E)$; true thresholds t_v for every node v. (These thresholds are not available to the inference algorithm; they are used only to compute the successor for a query.)

Result: Infer the thresholds of all the nodes of G. (For data analysis purposes, the algorithm also returns the set Q of all the queries used in the inference procedure.)

for v *in* V **do**
  Let $t_L(v) = 0$ and $t_H(v) = d(v) + 2$;
end
Let $V_t = V$ denote the set of nodes for which threshold needs to be inferred;
Let $Q = \varnothing$ be the query set;
while $V_t \neq \varnothing$ **do**
  Let $i = 1$;
  Let q_i be the current query. Let $q_i[v] = 0$, $\forall v \in V$;
  Let $V_{\text{rem}} = V_t$;
  while $V_{\text{rem}} \neq \varnothing$ **do**
    Let $v_{\max} = \arg\max_{v \in V_{\text{rem}}} t_H(v) - t_L(v)$;
    Set exactly $\lfloor (t_H(v_{\max}) + t_L(v_{\max}))/2 \rfloor$ neighbors of $v_{\max}$ to state 1 in q_i;
    $V_{\text{rem}} \leftarrow V_{\text{rem}} \setminus \{N[v_{\max}, G^2]\}$;
  end
  Compute the successor s of q_i;
  //Update $t_L(v)$ and $t_H(v)$ for all $v \in V_t$
  for $v \in V_t$ **do**
    if *s[v]=0 and* $t_L(v) \leq score(q_i, v)$ **then**
      $t_L(v) \leftarrow \text{score}(q_i, v) + 1$;
    end
    else if *s[v]=1 and* $t_H(v) > score(q_i, v)$ **then**
      $t_H(v) \leftarrow \text{score}(q_i, v)$;
    end
  end
  Remove all nodes v from V_t such that $t_L(v) = t_H(v)$;
  $Q \leftarrow Q \cup \{q_i\}$;
end

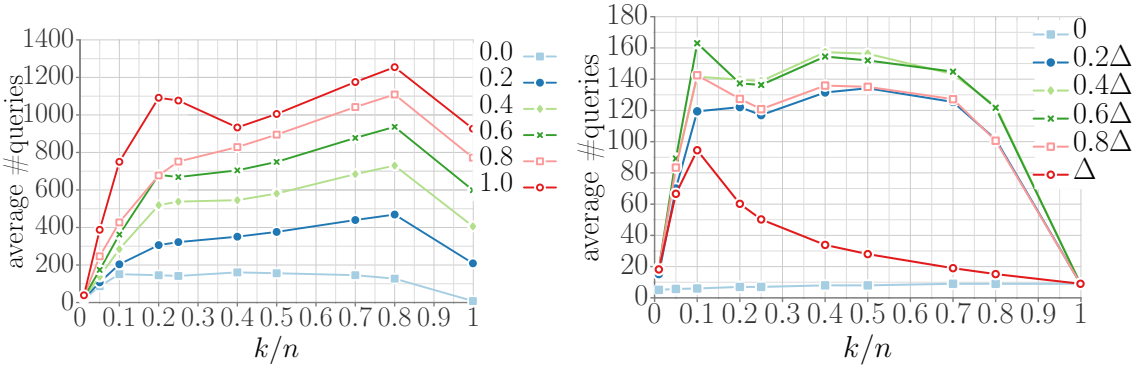


Figure 5: **Experiments with 1000 node random k-regular graphs.** (a) The threshold of a node is randomly assigned an integer in the interval $\lceil (k+2)(1-\theta)/2, (k+2)(1+\theta)/2 \rceil$. The legend shows values of θ. (b) All nodes are assigned a fixed threshold τ relative to k. The legend shows values of τ.

successor. We terminate this process when for all v, $t_L(v) = t_H(v)$. Results are summarized in the last column of Table 1. 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. Recall the threshold assignment procedure described in Section 7.1. The results are in Figures 5(a) and 6(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 5. 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 5(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 used is a 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 results 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 Δ. 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 networks. In Figure 5(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 (i.e., 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 used 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 fewer queries than for sparser networks.

Progress towards inferring thresholds. In Figure 6(b), we plot the accumulated threshold ranges for all vertices as the algorithm moves from one query to the next. We note that for most of the networks considered, 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. Thus, the uncertainty in the threshold range can be decreased significantly using a small number of adaptive queries.

In Appendix B, we have provided plots (Figure 7) for the wall clock time taken by Method 2 to generate queries.

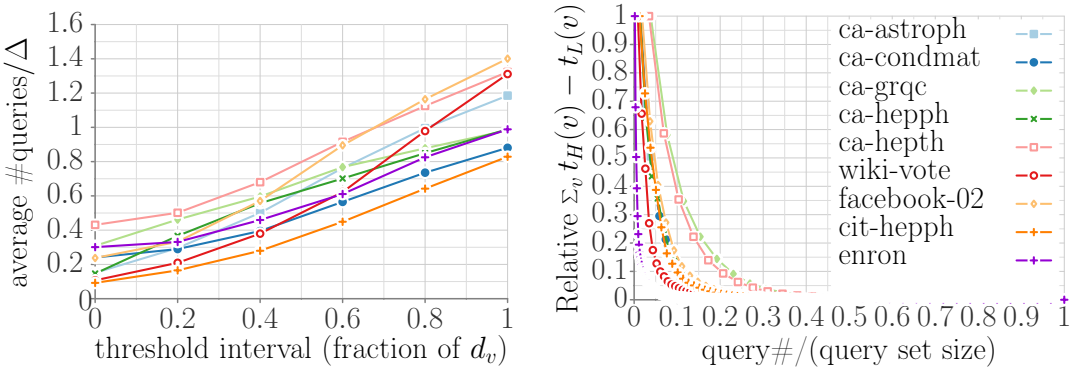


Figure 6: **Inferring thresholds for real-world networks.** (a) Adaptive heuristic for varying threshold ranges. (b) Progress made by the adaptive algorithm (Method 2) in each query.

8. Directions for Future Work

In this work, we studied the problem of inferring the behavior of a networked dynamical system through active querying. We now discuss several avenues for future work. We first mention an open problem that arises directly from our results. Theorem 13 shows the existence of a small complete query set for SyDSs with symmetric functions. Developing an efficient algorithm that can construct such a query set is an interesting research question. In our framework, the user has complete knowledge of the network, and queries and the responses from the system are assumed to specify the states of all the nodes in the system. In a typical real-world scenario, only partial knowledge of the system is available. Besides, the user might only require partial information from the system. For example, it may be sufficient for a user to know whether the threshold of every node is at most some chosen integer α. Methods that can accomplish threshold inference with a small number of queries

in such contexts are of practical interest. Another useful direction to explore is to develop inference methods when the system response consists of successors for $k \geq 2$ time steps (instead of a single time step). 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

We thank the referees for providing very helpful feedback. This work has been partially supported by DTRA (Contract HDTRA1-19-D-0007), University of Virginia Strategic Investment Fund award number SIF160, NSF grants IIS-1633028 (BIG DATA), CMMI-1745207 (EAGER), OAC-1916805 (CINES), CCF-1918656 (Expeditions), CNS-2028004 (RAPID), OAC-2027541 (RAPID) and IIS-1908530, USAID under the Cooperative Agreement NO. AID-OAA-L-15-00001 and USDA NIFA FACT 2019-67021-29933. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government.

Appendix A. Generating query sets based on probabilistic methods

We first recall the following definition. Given a positive real number x, we use $\langle x \rangle$ to denote the integer obtained by rounding x to the nearest integer; that is, $\langle x \rangle = \lfloor x \rfloor$ if the fractional part of x is *less than* 0.5; otherwise, $\langle x \rangle = \lceil x \rceil$.

Statement of Theorem 12. Let $\mathcal{S}$ be a symmetric SyDS with underlying graph $G(V, E)$, where $|V| = n$ and maximum node degree $= \Delta$. 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 $\mathcal{S}$.

Proof. In proving this 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 ones assigned to the closed neighborhood of v.

Our method, 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 the response for $f_v(0)$ and $f_v(d(v)+1)$, respectively for all $v \in V$. This contributes just an additive term 2. 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 $q \sim \mathbb{D}\left(\frac{z}{\Delta+1}\right)$, where $z = \left\langle \frac{b(\Delta+1)}{d(v)+1} \right\rangle$, we have

$$\begin{aligned}
\Pr\left(f_v(b) \text{ is queried in } q\right) &\geq \binom{d(v)+1}{b}\left(\frac{z}{\Delta+1}\right)^b\left(1-\frac{z}{\Delta+1}\right)^{d(v)+1-b} \\
&\geq \frac{1}{11\sqrt{d(v)+2}} \\
&\geq \frac{1}{11\sqrt{\Delta+2}},
\end{aligned}$$

where the inequality in the second of the three lines above follows from Lemma 27 (below). Now,

$$\Pr\big(f_v(b) \text{ is not queried by } Q\big) \quad \leq \quad \Pr\big(f_v(b) \text{ is not queried by } q_{zj}, \\ 1 \leq j \leq 22\sqrt{\Delta+2}\,\log(n\Delta)\big).$$

The quantity on the right hand side of the above inequality is at most

$$\left(1 - \tfrac{1}{11\sqrt{\Delta+2}}\right)^{22\sqrt{\Delta+2}\,\log(n\Delta)} < \tfrac{1}{(n\Delta)^2}\,.$$

Note that

$$\Pr\big(Q \text{ is not a complete set}\big) = \Pr\big(\exists v, b \text{ such that } f_v(b) \text{ is not queried by } Q\big).$$

By the union bound, the quantity on the right hand side of the above equation is at most

$$\textstyle\sum_{v\in V}\sum_{b=1}^{\Delta} \Pr\big(f_v(b) \text{ is not queried by } Q\big) < \tfrac{1}{n\Delta}\,.$$

This completes the proof of Theorem 12. ■

Lemma 27 *Let b, d and D be positive integers such that $b \leq d \leq D$ and let $z = \langle \frac{bD}{d} \rangle$. Then, $\binom{d}{b}\big(\frac{z}{D}\big)^b\big(1-\frac{z}{D}\big)^{d-b} \geq \frac{1}{11\sqrt{d+1}}$.*

We will first prove the following claims.

Claim 28 *$\big(1+\frac{1}{b}\big)^b$ is monotone increasing in b for positive integers.*

Proof: Consider the collection of $(b+1)$ numbers $\big(1, \frac{b+1}{b}, \ldots, \frac{b+1}{b}\big)$. Using the fact that their arithmetic mean is $\geq$ their geometric mean,

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

■

Claim 29 *$\binom{d}{b}\big(\frac{b}{d}\big)^b\big(1-\frac{b}{d}\big)^{d-b} \geq \frac{1}{\sqrt{2(d+1)}}$.*

Proof: Let $h(b,d) = \binom{d}{b}\big(\frac{b}{d}\big)^b\big(1-\frac{b}{d}\big)^{d-b}$. 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 $b = \lfloor \frac{d}{2} \rfloor$.

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

where, $b' = d - b - 1$. When $b < \frac{d}{2}$, $b' \geq b$ and when $b \geq \frac{d}{2}$, $b' < b$. Applying Claim 28, 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 $b = 2k+1$.

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

The inequality follows from Claim 28. 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 30 *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 27. 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 30.

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

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

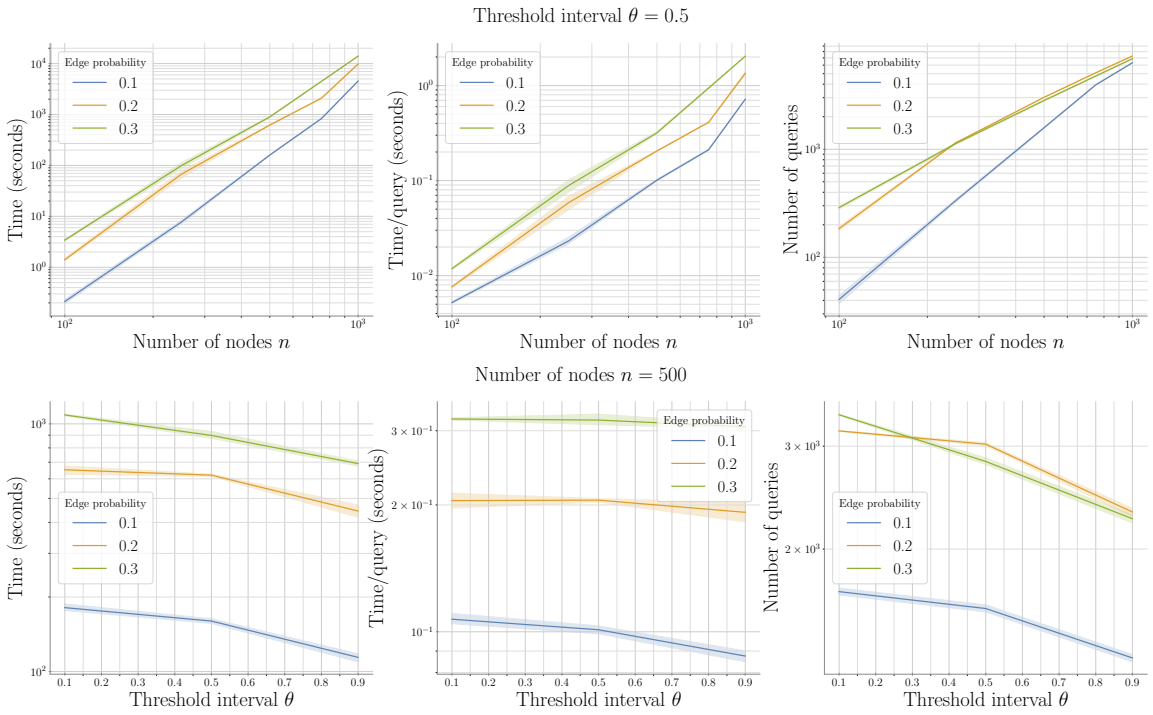


Figure 7: Plots of wall-clock time to generate queries using Method 2 for ER graphs. The first row corresponds to increasing number of nodes while the second row corresponds to increasing threshold interval θ. The first column corresponds to total time taken, the second column corresponds to average time per query, and third column corresponds to the number of queries for reference. For each combination of (n, p, θ), the values shown are averages over 10 instances.

This completes the proof of Lemma 27. ■

Appendix B. Implementation, computing environment and time

Implementations were done in Python 2.7 and 3.8. Compute nodes with 2 x Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz processors with 20 cores per CPU and 386GB memory were used. In Figure 7, we have provided plots for (i) total time taken and (ii) average time per query as a function of graph size (nodes and edges) and threshold interval θ for ER graphs. Also, for reference, we have provided the number of queries generated for each case. We observe that both total time as well as average time per query are super-linear in the number of nodes even though the number of queries generated is close to being linear in the number of nodes. This is because, the time taken depends on the total number of queries as well as the number of operations required to generate a query. Also, as θ increases, the time taken decreases for a fixed network. This can be mainly attributed to the decrease in the number of queries (last row).

References

A. Abasi, N. H. Bshouty, and H. Mazzawi. On exact learning monotone DNF from membership queries. *CoRR*, abs/1405.0792:1–16, 2014.

B. Abrahao, F. Chierichetti, R. Kleinberg, and A. Panconesi. Trace complexity of network inference. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 491–499. ACM, 2013.

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

A. Adiga, V. Cedeno-Mieles, C. J. Kuhlman, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. Inferring probabilistic contagion models over networks using active queries. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 377–386. ACM, 2018a.

A. Adiga, C. J. Kuhlman, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. Learning the behavior of a dynamical system via a "20 questions" approach. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, pages 4630–4637, 2018b.

A. Adiga, C. J. Kuhlman, M. V. Marathe, S. S. Ravi, and A. K. Vullikanti. PAC learnability of node functions in networked dynamical systems. In *Proc. International Conference on Machine Learning (ICML)*, pages 82–91, 2019.

A. Adiga, C. J. Kuhlman, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, R. E. Stearns, and A. K. Vullikanti. Bounds and complexity results for learning coalition-based interaction functions in networked social systems. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, New York, NY, USA, February 2020*, pages 3138–3145, 2020.

F. Allen and D. Gale. Financial contagion. *Journal of Political Economy*, 108:1–33, 2000.

D. Angluin and D. K. Slonim. Randomly fallible teachers: Learning monotone DNF with an incomplete membership oracle. *Machine Learning*, 14(1):7–26, 1994.

D. Angluin, M. Frazier, and L. Pitt. Learning conjunctions of horn clauses. *Machine Learning*, 9(2):147–164, 1992.

D. Angluin, L. Hellerstein, and M. Karpinski. Learning read-once formulas with queries. *J. ACM*, 40(1):185–210, 1993.

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

J. Broere, V. Buskens, H. Stoof, and A. Sánchez. An experimental study of network effects on coordination in asymmetric games. *Scientific reports*, 9(1):1–9, 2019.

V. Cedeno-Mieles, Z. Hu, Y. Ren, X. Deng, A. Adiga, C. Barrett, N. Contractor, S. Ekanayake, J. M. Epstein, B. J. Goode, et al. Networked experiments and modeling for producing collective identity in a group of human subjects using an iterative abduction framework. *Social Network Analysis and Mining*, 10(1):1–43, 2020.

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

D. Centola. An experimental study of homophily in the adoption of health behavior. *Science*, 334:1269–1272, December 2011.

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

D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos. Epidemic thresholds in real networks. *TISSEC*, 10:1–33, 2008.

G. Charness, R. Cobo-Reyes, and N. Jimenez. Identities, selection, and contributions in a public-goods game. *Games and Economic Behavior*, 87:322–338, 2014.

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

S. Dasgupta and J. Langford. A tutorial on active learning. Tutorial at ICML, 2009.

S. Dasgupta, D. Hsu, S. Poulis, and X. Zhu. Teaching a black-box learner. In *International Conference on Machine Learning*, pages 1547–1555. PMLR, 2019.

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

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

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

A. Ganesh, L. Massoulie, and D. Towsley. The effect of network topology on the spread of epidemics. In *Proc. INFOCOM*, pages 1455–1466, 2005.

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

S. A. Goldman and M. J. Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995.

M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1019–1028. ACM, 2010.

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

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

X. He, K. Xu, D. Kempe, and Y. Liu. Learning influence functions from incomplete observations. In *Advances in Neural Information Processing Systems*, pages 2073–2081, 2016.

H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, and R. E. Stearns. The complexity of planar counting problems. *SIAM J. Comput.*, 27(4):1142–1167, 1998.

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

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

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

C. A. Latkin. Outreach in natural settings: The use of peer leaders for HIV prevention among injecting drug users' networks (supplement). *Public Health Reports*, 113:151–159, 1998.

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

E. L. Lloyd and S. Ramanathan. On the complexity of distance-2 coloring. In *Computing and Information - ICCI'92, Fourth International Conference on Computing and Information, Toronto, Ontario, Canada, May 28-30, 1992, Proceedings*, pages 71–74, 1992.

A. Lokhov. Reconstructing parameters of spreading models from partial observations. In *Advances in Neural Information Processing Systems*, pages 3467–3475, 2016.

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

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

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

W. Mulzer and G. Rote. Minimum-weight triangulation is NP-hard. *J. ACM*, 55(2):11:1–11:29, 2008.

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

H. Narasimhan, D. C. Parkes, and Y. Singer. Learnability of influence in networks. In *Advances in Neural Information Processing Systems*, pages 3186–3194, 2015.

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

B. A. Prakash, D. Chakrabarti, N. Valler, M. Faloutsos, and C. Faloutsos. Threshold conditions for arbitrary cascade models on arbitrary networks. *Knowledge and Information Systems*, 33(3):549–575, 2012.

D. M. Romero, B. Meeder, and J. Kleinberg. Differences in the mechanics of information diffusion across topics: Idioms, political hashtags, and complex contagion on Twitter. In *Proceedings of the 20th International Conference on World Wide Web*, pages 695–704. ACM, 2011a.

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

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

F. C. Santos, M. D. Santos, and J. M. Pacheco. Social diversity promotes the emergence of cooperation in public goods games. *Nature*, 454(7201):213–216, 2008.

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

T. C. Schelling. *Micromotives and Macrobehavior*. Norton, New York, NY, 1978.

T. Schmidt. Computational complexity of SAT, XSAT and NAE-SAT for linear and mixed Horn CNF formulas. Ph.D. Thesis, Department of Mathematics, University of Koln, 2010.

B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

S. G. Sherman, D. S. Ganna, K. E. Tobin, C. A. Latkin, C. Welsh, and P. Bielenson. 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, 2009.

R. H. Sloan, B. Szorenyi, and G. Turan. Learning boolean functions with queries. In *Boolean Models and Methods in Mathematics, Computer Science and Engineering*, chapter 7, pages 221–256. Oxford University Press, New York, NY, 2013.

S. Soundarajan and J. E. Hopcroft. Recovering social networks from contagion information. In *Proceedings of the 7th Annual Conference on Theory and Models of Computation*, pages 419–430. Springer, 2010.

E. A. Stevens and M. J. Prinstein. Peer contagion of depressogenic attributional styles among adolescents: A longitudinal study. *Journal of Abnormal Child Psychology*, 33(1): 25–37, 2005.

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

T. W. Valente. *Network Models of the Diffusion of Innovations*. Hampton Press, 1995.

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

T. W. Valente. *Social Networks and Health: Models, Methods, and Applications*. Oxford University Press, New York, NY, 2010.

T. W. Valente. Network interventions. *Science*, 337:49–53, 2012.

V. V. Vazirani. *Approximation Algorithms*. Springer, New York, NY, 2001.

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

D. B. West. *Introduction to Graph Theory*. Prentice-Hall, Englewood Cliffs, NJ, 2001.

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