Localization and Identification in Networks Using Robust Identifying Codes

Moshe Laifenfeld
Department of Electrical and Computer Eng.
Boston University
Boston, MA 02215, USA
Email: moshel@bu.edu

Abstract—Many practical problems in networks such as fault detection and diagnosis, location detection, environmental monitoring, etc., require to identify specific nodes or links based on a set of observations, which size is a subject of optimization. In this paper we focus on a combinatorial approach to these problems, which is closely related to coding techniques, and specifically to identifying codes. The identifying code problem for a given graph involves finding a minimum set of vertices whose neighborhoods uniquely overlap at any given graph vertex. In this paper we show efficient reductions between the identifying code problem and well-known covering problems, resulting in a tight hardness of approximation result and provable good centralized and distributed approximations. We further provide empirical and theoretical results on identifying codes in random networks and efficient constructions of these codes in infinite grids.

I. INTRODUCTION

The need for efficient algorithms for identifying a set of nodes or links in a network has emerged as networks became larger and harder to manage. Applications such as fault detection and diagnosis in network and specifically in processor arrays [5], location detection in sensor networks [6,7], environmental monitoring in utility networks [10], and others, have been studied extensively, using different approaches that have resulted in a large variety of practical algorithms as well as theoretical results. In this paper we focus on a combinatorial approach, which is closely related to error-correcting codes, and specifically to identifying codes. Many of the results provided in this paper can be found in [1–4].

A. Identifying Codes

An identifying code is a subset of vertices in a graph with the property that the (incoming) neighborhood of any vertex has a unique intersection with the code. For example, a three dimensional cube (as depicted in Figure 1(a)) has a three-vertex identifying code (labeled \{1, 2, 3\} in the figure). The neighborhood of each vertex in the graph intersects uniquely with this code, and such an intersection is called an identifying set\(^1\); given an identifying set, one can thus uniquely identify the vertex in the graph that produced it. In this case, the code provided is also optimal, because one needs at least \(\lg 8 = 3\) code vertices to produce 8 distinct identifying sets (corresponding to the 8 vertices of the cube)\(^2\). The goal of

\(^1\)Unlike traditional identifying codes, the empty set is considered a valid identifying set here.

\(^2\)As is common, we use the notation \(\lg(x)\) to denote \(\log_2(x)\).
able to detect and determine the pollution source. The problem of placing the smallest number of sensors was found to be a variant of the identifying code problem [1]. For wireless networks, the identification property of identifying codes was used to uniquely label sensors in a network, providing natural means of routing on top of the traditional monitoring functionality [2].

2) Theoretical connections: From a theoretical perspective, identifying codes are closely linked to error-correcting codes, specifically, super-imposed codes [5], covering codes [5, 9], and locating-dominating sets [11]. The intimate relationship between identifying codes and super-imposed codes was initially pointed out in [5] and further developed in [12, 13], providing optimal constructions and tight bounds on code size. Locating-dominating sets are very similar to identifying codes with the subtle difference that only the vertices not in the locating-dominating set are required to have unique identifying codes, whereas identifying codes include all vertices. Unlike identifying codes, every graph admits a trivial locating-dominating set—the entire set of vertices. Links to other well-studied problems can be found in the literature: to the set cover problem in [1, 14], to the dominating-set in [15], and to the set multicovery test cover problems in [1]. The test cover problem is, in fact, a generalization of the identifying code problem, and some of its results apply throughout.

Many variants of identifying codes have emerged since they were first introduced. In radius \(r > 1\) identifying codes [5], the neighborhood of \(v\) is redefined to include all vertices at distance \(\leq r\) from \(v\). The \((1, \leq l)\)-identifying codes [12] can simultaneously identify any subset of at most \(l\) vertices. Dynamic identifying codes are identifying codes whose codewords form a walk through the graph. This variant was studied in [9] with applications to dynamic agents for fault detection in multiprocessor arrays. Robust identifying codes were suggested in [6, 7] for applications of location detection in harsh environments, where vertices and connecting edges are likely to fail. Intuitively, an \(r\)-robust identifying code is a code that maintains its identification property in the event of a removal or insertion of up to \(r\) different vertices from all identifying sets. Refer to Figure 2 for an example of a 1-robust identifying code in a cube. The observation that \(r\)-robust identifying codes are error correcting codes of minimum Hamming distance of \(2r + 1\) was made in [7]. Theoretical bounds closely related to covering codes, and some efficient constructions for periodic geometries were further developed in [9].

3) Approximating the optimal identifying code: In the most general situation, finding a minimum size identifying code for arbitrary undirected and directed graphs was proven to be NP-complete in [16], based on a reduction to the 3-satisfiability problem [17]. An exception to this result is the specific case of directed [18] and undirected trees, for which there exists a polynomial-time algorithm for finding a minimum radius 1 identifying code.

Significant efforts in the research of identifying codes and their variants have focused on finding efficient constructions in two dimensional lattices, grids and Hamming spaces (see for example [19, 20], and [9] for a summary of recent results). Until recently, little has been published towards a polynomial time approximation algorithm for arbitrary graphs. In [7] a polynomial-time greedy heuristic and its distributed variant were suggested for obtaining an identifying code in an arbitrary graph, and simulations showed it to work well over random graphs. Unfortunately, no guarantees for the quality of the obtained solution were presented, and Moncel later proved in [21] that no such guarantees exist.

Independently and in parallel, several groups have been looking into the question of approximability of identifying codes [1, 3, 14, 15], obtaining polynomial-time approximations within an \(O(\log |V|)\) factor of the optimal solution. In [15] the authors tied identifying codes to the dominating set problem, thereby showing that, under common complexity assumptions, approximating identifying codes within a sub-logarithmic factor is intractable. More precisely, it has been shown that identifying codes can be approximated within \(O(\log |V|)\) factor, but they can not be approximated in polynomial time within \(1 + \alpha \log |V|\) factor for some \(\alpha > 0\). In [1, 3], we have provided an explicit value for \(\alpha\) by demonstrating that identifying codes are not approximable within a \(\ln |V|\) factor unless \(NP \subseteq DTIME^{\log \log |V|}\); our result is based on a reduction from the set cover problem, and we use it to carry over the hardness result of Feige [22]. We further show that this bound is tight by adapting an algorithm developed by Berman et al. in [23] that attains this bound within a small additive constant.

4) Random graphs: Recently, random graphs and random geometric graphs were studied in the context of identifying codes [13, 24]. In [13] it was shown that for asymptotically large random graphs, any subset of a certain threshold size (logarithmic in the size of the graph) is almost surely an identifying code. It was also shown that the threshold is asymptotically sharp, i.e., the probability of finding an identifying code of slightly smaller size asymptotically approaches zero. Unit disk geometric random graphs, in which vertices are placed on a two-dimensional plane and connected if their distance is less than some unit, were studied in [24]. Unlike large random graphs, most of the large unit-disk geometric random graphs do not possess identifying codes.

B. Related Combinatorial Problems

1) Set cover: Let \(U\) be a base set of \(m\) elements and let \(S\) be a set of subsets of \(U\). A cover \(C \subseteq S\) is a collection of subsets whose union is \(U\). The set cover problem asks to find a cover \(C\) of smallest cardinality. The set cover problem is one of the oldest and most studied NP-hard problems [17]. It admits the following greedy approximation: at each step, and until exhaustion, choose the heretofore unselected set in \(S\) that covers the largest number of uncovered elements in the base set.

The performance ratio of the greedy set cover algorithm has also been well-studied. The classic result of Johnson [25] showed that \(\frac{s_{\text{greedy}}}{s_{\text{min}}} = \Theta(\ln m)\), where \(s_{\text{min}}, s_{\text{greedy}}\) are the minimum and the greedy covers, and \(m\) is the size of the base
set. Recent studies on the hardness of approximation of the set cover problem can be found in [22, 26]. Raz and Safra [26] showed that the set cover problem is NP-hard and that it cannot be approximated by a polynomial algorithm within a \( O(\log m) \) factor from an optimal solution unless \( P=NP \). A tighter result was obtained by Feige [22] who showed that for any \( \epsilon > 0 \), no polynomial-time algorithm can approximate the minimum set cover within \( (1 - \epsilon) \ln m \) factor unless \( NP \) has deterministic algorithms operating in slightly super-polynomial time, \( i.e.\), \( NP \subset TIME [m^{O(\log \log m)}] \), suggesting that the greedy approach is one of the best polynomial approximations to the problem.

2) Set multicover: The minimum set \( k \)-multicover problem is a natural generalization of the minimum set cover problem, in which one is given a pair \((U, S)\) and seeks the smallest subset of \( S \) that covers every element in \( U \) at least \( k \) times. Often this problem is addressed as a special case of a more general family of integer optimization problems - the covering integer problem [27, 28].

The set multicover problem admits a similar greedy heuristic to the set cover problem: in each iteration select the set which covers the maximum number of non-\( k \)-multicovered elements. It is well known [28] that the performance guarantee of this heuristic is upper bounded by \( 1 + \log \alpha \), where \( \alpha \) is the largest set’s size.

3) Test cover: Another closely related problem is the test cover problem. This problem asks to find the smallest set \( T \) of given tests \( T_i \subseteq U \) such that any pair \( x, y \in U \) is differentiated by at least one test \( T_i \) (\( i.e., |\{x, y\} \cap T_i| = 1 \)).

The test covering problem appears naturally in identification problems, with roots in an agricultural study more than 20 years ago, regaining interest recently due to applications in bioinformatics [23, 29].

Garey and Johnson [30] showed the test cover problem to be NP-hard and later Moret and Shapiro [31] suggested greedy approximations based on a reduction to the set cover problem. More recent work [29, 32] studied different branch-and-bound approximations and established a hardness of approximation by extending the reduction in [31], and using a result of Feige [22]. Berman et al. [23] also suggested a novel greedy approximation and showed its performance ratio to be within a small constant from the hardness result of [32].

The test cover is clearly a general case of the identifying code problem, with tests corresponding to outgoing balls, and as such many of its results can be applied directly \( i.e., \), [23]. Other results, such as the hardness of approximation, require some work due to the dependencies imposed by graph geography on nearby identifying sets.

C. Organization

The rest of the paper is organized as follows: We give the intuition behind the reduction from the set multicover problem and several approximations that result from this reduction in Section II. In Section III, we provide two flavors of distributed implementations of our approximation algorithm, and in Section IV, we provide a hardness of approximation result that shows our algorithms to perform close to the achievability bound (under common complexity assumptions). We further show that for the non-robust identifying codes case our hardness of approximation results is tight by adapting a test cover approximation (based on [23]), which attains our hardness result up to a small additive constant. In Section V, we discuss robust identifying codes in random graphs and finally in Section VI we provide simulations results on random graphs and grids. Due to space limitations many of the details and proof are omitted. More elaborated discussion on the main results can be found in [1–4].

II. APPROXIMATIONS OF OPTIMAL ROBUST IDENTIFYING CODES

In this section we illustrate the reduction of (robust) identifying codes problem from the set (multi) cover problem that yield both provable good approximations and hardness of approximation results. We start with a formal definition of robust identifying code.

Given a directed graph \( G = (V, E) \), the incoming ball \( B^+(v) \) consists of vertices that have an edge directed towards \( v \in V \), together with \( v \); likewise, the outgoing ball \( B^-(v) \) consists of vertices that have an edge directed away from \( v \), together with \( v \). For undirected graphs, we shall simply use the notation \( B(v) = B^+(v) = B^-(v) \).

As such, an identifying code is a set of vertices in a graph \( G \) with the property that any incoming ball in \( G \) has a unique intersection with the identifying code. More precisely, a non-empty subset \( C \subseteq V \) is called a code and its elements are codewords. For a given code \( C \), the identifying set \( I_C(v) \) of a vertex \( v \) is defined to be the codewords directed towards \( v \), \( i.e., I_C(v) = B^+(v) \cap C \) (if \( C \) is not specified, it is assumed to be the set of all vertices \( V \)). A code \( C \) is thus an identifying code if each identifying set of the code is unique, or in other words \( \forall u, v \in V \quad u = v \leftrightarrow I_C(u) = I_C(v) \). 3

Definition 1 An identifying code \( C \) over a given graph \( G = (V, E) \) is said to be \( r \)-robust if \( I_C(u) \oplus A \neq I_C(v) \oplus D \) for all \( v \neq u \) and \( A, D \subseteq V \) with \( |A|, |D| \leq r \). Here \( \oplus \) denotes the symmetric difference.

A. Reduction intuition

Consider a three dimensional cube as in Figure 2 and let \( C = \{0, 1, 2, 4, 5, 6, 7\} \). Clearly, the identifying sets are all unique, and hence the code is an identifying code. A closer look reveals that \( C \) is actually a 1-robust identifying code, so that it remains an identifying code even upon removal or insertion of any vertex into any identifying set.

A graph’s adjacency matrix provides a linear algebra view of the identifying code problem. Specifically, we can consider each row and column of the matrix to be a characteristic vector of the ball around some vertex in the graph, meaning that their \( i \)-th entry of the row \( j \) is 1 if and only if the \( i \)-th vertex of \( V \) is in the ball around node \( j \). Selecting codewords can thus be

3Note that this definition does not include the standard assumption (which we will make later) that all identifying sets are non-empty.
viewed as selecting columns to form a matrix of size $n \times |C|$. We will refer to this matrix as the code matrix. A code is thus identifying if the Hamming distance between every two rows in the code matrix is at least one (recall that the Hamming distance of two binary vectors is the number of ones in their bitwise XOR). It has been shown in \cite{7,8} that an identifying code can be built in a localized manner, where support that corresponds to $v$ in order to operate. It was observed in \cite{7} that an $r$-dominating set is a set of vertices, $S_r$, such that every vertex in $V$ is in the ball of radius 1 of at least one vertex in $S_r$. We extend this notion to define an $r$-dominating set to be a set of vertices $S_r$ such that every vertex in $V$ is in the ball of radius 1 of at least $r$ vertices in $S_r$.

Lemma 1 Given a graph $G = (V, E)$, an $r + 1$-dominating set $C$ is also an $r$-robust identifying code if and only if every pair $(u, v) \in U$ such that $\rho(u, v) \leq 2$ is distinguished by at least $2r + 1$ codewords in $C$.

The proof is omitted and can be found in \cite{2}.

The localized robust identifying code approximation. Lemma 1 can serve as the basis for a similar reduction from a set multicolor problem, with the main difference that the basis elements are limited to vertex pairs that are at most two hops apart.

Towards this end we define $U^2 = \{(u, v) | \rho(u, v) \leq 2\}$, the set of all pairs of vertices (including $(v, v)$) that are at most two hops apart. Similarly, we will localize the distinguishing set $\delta_v$ to $U^2$ as follows:

$$\delta_v^2 = (\delta_v \cap U^2) \cup \{(u, u) | u \in B(v)\}.$$  

The resulting localized identifying code approximation is thus given by Algorithm 2 and can be shown to provide an $r$-robust identifying code for any graph that admits one (we omit the proof due to space considerations).

Theorem 1 Given an undirected graph $G = (V, E)$ of $n$ vertices, the performance ratio $r \text{ID} - \text{LOCAL}$ is upper bounded

Algorithm 1 Centralized $r$-robust code $r \text{ID} - \text{CENTRAL}(r, G)$

We start with a graph $G = (V, E)$ and a non-negative integer $r$. The greedy set multicolor approximation is denoted $\text{SET-MULTICOVER}(k, U, S)$.

1) Compute $\{I(u) | u \in V\}$
2) Compute $\Delta = \{\delta_u | u \in V\}$.
3) $C \leftarrow \text{SET-MULTICOVER}(2r + 1, U, \Delta)$
4) Output $C_{\text{central}} \leftarrow \{u \in V | \delta_u \in C\}$

Algorithm 2 Localized $r$-robust code $r \text{ID} - \text{LOCAL}(r, G)$

We start with a graph $G = (V, E)$ and a non-negative integer $r$. The greedy set multicolor approximation is denoted $\text{SET-MULTICOVER}(k, U, S)$.

1) Compute $U^2$
2) Compute $\Delta^2 = \{\delta_u^2 | u \in V\}$.
3) $C \leftarrow \text{SET-MULTICOVER}(2r + 1, U^2, \Delta^2)$
4) Output $C_{\text{local}} \leftarrow \{u \in V | \delta_u^2 \in C\}$
by:
\[
\frac{c_{\text{greedy}}}{c_{\text{min}}} < \ln \gamma + 1,
\]
where \( \gamma = \max_{v \in V} |B(v)|(|B(v; 3)| - |B(v)| + 1) \).

**Proof:** Recall that \( \delta_v \) includes all vertex pairs where one vertex is in the outgoing ball of \( v \) and the other is not. The size of the intersection of \( \delta_v \) and \( U^2 \) can be upper bounded by \( |B(v)|(|B(v; 3)| - |B(v)|) \), which becomes an equality when all vertices in \( B(v; 3) \) are within distance 2 of each other. Finally, \( \delta_v^2 \) includes also \( |B(v)| \) pairs of the same vertex \((u, u)\). Therefore \( |\delta_v^2| \leq |B(v)|(|B(v; 3)| - |B(v)| + 1) \), which at its maximum, can be applied to the performance guarantee of [28] to complete the proof.

In the next subsection we present a distributed implementation of the identifying code localized approximation. The following lemma supplements Lemma 1 by providing additional “localization”. At the heart of this lemma lies the fact that each codeword distinguishes between its neighbors and the remaining vertices.

**Lemma 2** The distinguishing sets \( \delta_u^2 \) and \( \delta_v^2 \) are disjoint for every pair \((u, v)\) with \( \rho(u, v) > 4 \).

**Proof:** Clearly, \( \delta_u^2 \) includes all vertex pairs \((x, y)\) \( \in U^2 \) where \( x \) is a neighbor of \( v \) and \( y \) is not. More precisely, \( (x, y) \in \delta_u^2 \) if
\[
x \in B(v) \text{ and } y \in B(x; 2) - B(v).
\]
Moreover, for all such \((x, y)\), \( \rho(x, v) \leq 1 \) and \( \rho(y, v) \leq 3 \). On the other hand, for \((x', y') \in \delta_v^2 \), with \( \rho(u, v) > 4 \), either \( x' \) or \( y' \) must be a neighbor of \( u \), and hence of distance 3 from \( v \). Thus, \( \delta_u^2 \) and \( \delta_v^2 \) are disjoint.

Lemma 2 implies that, when applying the greedy algorithm, a decision to choose a codeword only affects decisions on vertices within four hops; the algorithm is thus localized to vicinities of radius four.

**III. DISTRIBUTED ALGORITHMS**

Several parallel algorithms exist in the literature for set cover and for the more general covering integer programs (e.g., [27]). There are also numerous distributed algorithms for finding a minimum (connected) dominating set based on set cover and other well known approximations such as linear programming relaxation (e.g., [33]). In a recent work, Kuhn et. al. [33] devised a distributed algorithm for finding a dominating set with a constant runtime. The distributed algorithm uses a design parameter which provides a tradeoff between the runtime and performance.

Unfortunately the fundamental assumption of these algorithms is that the elements of the basis set are independent computational entities (i.e., the nodes in the network); this makes it non-trivial to apply them in our case where elements correspond to pairs of nodes that can be several hops apart.

Moreover, we assume that the nodes are energy constrained so that reducing communications is very desirable, even at the expense of longer execution times and reduced performance.

We next provide two distributed algorithms. The first is completely asynchronous, guarantees a performance ratio of at most \( \ln \gamma + 1 \), and requires \( \Theta(c_{\text{dist}}) \) iterations at worst, where \( c_{\text{dist}} \) is the size of the distributed identifying code. The second is a randomized algorithm, which requires a coarse synchronization, guarantees a performance ratio of at most \( \ln \gamma + 1 \), and for some design parameter \( K \geq 2 \) and arbitrary small \( \epsilon > 0 \) operates within \( O(\frac{\gamma n \log \frac{\gamma}{\epsilon}}{K}) \) subslots (resulting in the communication of \( O(c_{\text{dist}} \max_{v \in V} |B(v; 4)|) \) messages).

In the next subsection we describe the setup and initialization stages that are common to both distributed algorithms.

**A. Setup and initialization**

With a setup similar to [7] we assume that every vertex (node) is pre-assigned a unique serial number and can communicate reliably and collision-free (perhaps using higher-layer protocols) over a shared medium with its immediate neighborhood. Every node can determine its neighborhood from the IDs on received transmissions, and higher radius balls can be determined by distributing this information over several hops. In our case, we will need to know \( G(v; 4) \) the subgraph induced by all vertices of distance at most four from \( v \).

Our distributed algorithms are based on the fact that, by definition, each node \( v \) can distinguish between the pairs of nodes which appear in its corresponding distinguishing set \( \delta_v^2 \) given in (1). This distinguishing set is updated as new codewords are added to the identifying code being constructed; their presence is advertised by flooding their four-hop neighborhood.

**B. The asynchronous algorithm rID → ASYNC**

The state diagram of the asynchronous distributed algorithm is shown in Figure 3. All nodes are initially in the *unassigned* state, and transitions are effected according to messages received from a node’s four-hop neighborhood. Two types of messages can accompany a transition: *assignment* and *declaration* messages, with the former indicating that the initiating node has transitioned to the *assigned* state, and the latter being used to transmit data. Both types of messages also include five fields: the *type*, which is either “assignment” or “declaration”, the *ID* identifying the initiating node, the
hop number, the iteration number, and data, which contains the size of the distinguishing set in the case of a declaration message.

Following the initialization stage, every node declares its distinguishing set’s size. As a node’s declaration message propagates through its four hop neighborhood, every forwarding node updates two internal variables, \( ID_{\text{max}} \) and \( \delta_{\text{max}} \), representing the ID and size of the most distinguishing node (ties are broken in favor of the lowest ID). Hence, when a node aggregates the declaration messages initiated by all its four hop neighbors (we say that the node reached its end-of-iteration event), \( ID_{\text{max}} \) should hold the most distinguishing node in its four hop neighborhood. A node that reaches end-of-iteration event transitions to either the wait-for-assignment state or to the final assigned state depending if it is the most distinguishing node.

The operation of the algorithm is completely asynchronous; nodes take action according to their state and messages received. During the iterations stage, nodes initiate a declaration message only if they receive an assignment message or if an updated declaration (called an unassignment message) is received from the most distinguishing node of the previous iteration. All messages are forwarded (and their hop number is increased) if the hop number is less than four. To reduce communications load, a mechanism for detecting and eliminating looping messages should be applied.

Every node, \( v \), terminates in either an “unassigned” state with \( |\delta_v^2| = 0 \) or in the “assigned” state. Clearly, nodes that terminate in the “assigned” state constitute a localized \( r \)-robust identifying code.

**Algorithm 3** Asynchronous \( r \)-robust algorithm (\( r\text{ID} – \text{ASYNC} \))

We start with a graph \( G \), with vertices labeled by \( ID \), and a non-negative integer \( r \). The following distributed algorithm run at node \( v \in V \) produces an \( r \)-robust identifying code.

**Precomp**
- Compute \( \delta_v^2 \) using (1).
- Initiate a declaration message and set state = “unassigned”.
- Set \( ID_{\text{max}} = ID(v) \), \( \delta_{\text{max}} = |\delta_v^2| \), and \( ms \) to be an empty assignment message.

**Iteration**
- Increment \( hop(ms) \) forward all messages of \( hop(ms) < 4 \).
- If received an assignment message \( ms \) with state \( \neq \text{assigned} \) then
  - Update \( \delta_v^2 \) by removing all pairs covered \( 2r + 1 \) times.
  - Initiate a declaration message and set state = “unassigned”.
  - Reinitialize \( ID_{\text{max}} = ID(v) \) and \( \delta_{\text{max}} = |\delta_v^2| \).
- If state = waitforassignment and received an unassignment message then initiate a declaration message.
- If received a declaration message \( ms \) with state \( \neq \text{assigned} \) then
  - if \( \delta_{\text{max}} < \text{data}(ms) \) or \( \delta_{\text{max}} = \text{data}(ms) \) and \( ID_{\text{max}} > ID(ms) \) then \( \delta_{\text{max}} = \text{data}(ms) \), \( ID_{\text{max}} = ID(ms) \).
  - If end-of-iteration reached then
    - if \( ID_{\text{max}} = ID(v) \) and \( |\delta_v^2| > 0 \) then state = assigned, initiate an assignment message.
    - otherwise state = waitforassignment.

1) Performance Evaluation:

**Theorem 2** The algorithm \( r\text{ID} – \text{ASYNC} \) requires \( \Theta(c_{\text{dist}}) \) iterations and has a performance ratio 
\[
\frac{c_{\text{dist}}}{c_{\text{min}}} < \ln \gamma + 1,
\]
where \( \gamma = \max_{v \in V} |B(v)(|B(v); 3| - |B(v)| + 1)|. \)

The first part of the Theorem follows from Theorem 1 and the fact that only the most distinguishing set in a four hop neighborhoods is assigned to be a codeword. To see the number of iterations of the algorithm, we first note that in each iteration at least one codeword is assigned. The case of a cycle graph demonstrates that, in the worst case, exactly one node is assigned per iteration.

It follows that the amount of communications required in the iteration stage is \( \Theta(c_{\text{dist}}|V| \max(|B(v); 3|)) \), which can be a significant load for a battery powered sensor network. This can be significantly reduced if some level of synchronization among the nodes is allowed. In the next section we suggest a synchronized distributed algorithm that eliminates declaration messages altogether.

C. A low-communications randomized algorithm \( r\text{ID} – \text{SYNC} \)

In this subsection we assume that a coarse time synchronization among vertices within a neighborhood of radius four can be achieved. In particular, we will assume that the vertices maintain a basic time slot, which is divided into \( L \) subslots. Each subslot duration is longer than the time required for a four hop one-way communication together with synchronization uncertainty and local clock drift. After an initialization phase, the distributed algorithm operates on a time frame, which consists of \( F \) slots arranged in decreasing fashion from \( s_F \) to \( s_1 \). In general, \( F \) should be at least as large as the largest distinguishing set (e.g., \( F = \frac{n(n-1)}{2} \) will always work). A frame synchronization within a neighborhood of radius four completes the initialization stage.

The frame synchronization enables us to eliminate all the declaration messages of the asynchronous algorithm. Recall that the declaration messages were required to perform two tasks: (i) determine the most distinguishing node in its four hop neighborhood, and (ii) form an iteration boundary, i.e., end-of-iteration event. The second task is naturally fulfilled by maintaining the slot synchronization. The first task is performed using the frame synchronization: every node maintains a synchronized slot counter, which corresponds to the size of the current most distinguishing node. If the slot counter reaches the size of a node’s distinguishing set, the node assigns itself to the code. The subslots are used to randomly break ties.

1) Iterations stage: Each iteration takes place in one time slot, starting from slot \( s_F \). During a slot period, a node may transmit a message \( ms \) indicating that it is assigning itself as a codeword; the message will have two fields: the identification number of the initiating node, \( id(ms) \), and the hop number, \( hop(ms) \). A node assigns itself to be a codeword if its assignment time, which refers to a slot \( as \) and subslot \( l \), has been reached. Every time an assignment message is received, the assignment slot \( as \) of a node is updated to match the size of its distinguishing set; the assignment subslot is determined randomly and uniformly at the beginning of every slot.
2) Performance evaluation: Algorithm rID–SYNC requires at most \( O(n^2) \) slots (\( O(Ln^2) \) subslots), though it can be reduced to \( O(L\gamma) \) if the maximum size of a distinguishing set is propagated throughout the network in the precomputation phase. The communications load is low (i.e., \( O(c_{\text{dist}} \cdot \max_{v \in V}(|B(v;4)|)) \)), and includes only assignment messages, which are propagated to four hop neighborhoods.

In the case of ties, rID–SYNC can provide a larger code than gained from the localized approximation. This is because ties in the distributed algorithm are broken arbitrarily, and there is a positive probability (shrinking as the number of subslots \( L \) increases) that more than one node will choose the same subslot within a four hop neighborhood. As such, the \( L \) is a design parameter, providing a tradeoff between performance ratio guarantees and the runtime of the algorithm as suggested in the following Theorem.

**Theorem 3** For asymptotically large graphs, Algorithm rID–SYNC guarantees (with high probability) a performance ratio of

\[
\frac{c_{\text{dist}}}{c_{\text{min}}} < K(\ln \gamma + 1),
\]

where \( \gamma = \max_{v \in V} |B(v)| |B(v;3)| - |B(v)| + 1 \). The algorithm also requires \( O\left(\frac{2^n K + \epsilon}{K} \right) \) subslots to complete for design parameter \( K \geq 2 \) and arbitrarily small \( \epsilon > 0 \).

**Proof:** If no more than \( K \) tied nodes assign themselves simultaneously on every assignment slot, then we can upper bound the performance ratio by a factor \( K \) of Theorem 1, as in the theorem statement. We next determine the number of subslots \( L \) needed to guarantee the above assumption asymptotically with high probability.

Let \( P(K) \) denote the probability that no more than \( K \) tied nodes assign themselves in every assignment slot. Clearly, \( P(K) \geq (1 - p(K))^d_{\text{dist}} \), where \( p(K) \) is the probability that, when \( t \) nodes are assigned independently and uniformly to \( L \) subslots, there are at least \( K < t \) assignments to the same subslot. One can see that

\[
p(K) = \sum_{k=K}^{t} L\left(\frac{k}{t}\right) (1 - \frac{t}{L})^{t-k} \leq \sum_{k=K}^{t} L\left(\frac{k}{t}\right) K^{k} = tL\left(\frac{t}{K}\right)^K,
\]

where \( e \) being the natural logarithm and based on the assumption that \( K \leq \frac{t}{L} < 1 \). Let \( t = c_{\text{dist}} = n \) (this only loosens the bound) and \( L = \frac{1}{t} K^{\frac{2n+e}{n^{1+e}}} \). Then,

\[
P(K) \geq \left(1 - tL\left(\frac{te}{LK}\right)^K\right)^{c_{\text{dist}}} \geq \left(1 - e\frac{1}{Kn^{1+e}}\right)^n \rightarrow 1.
\]

**IV. HARDNESS OF APPROXIMATION**

The performance of the the approximation algorithms discussed in the previous sections are derived from the performance of the set multicover greedy heuristic [28], and are roughly \( O(\ln n) \), where \( n \) is the number of vertices in the graph. A natural question that arises is how good are these performance guarantees, and can we find polynomial time approximations that significantly outperform the suggested approximations. The main result of this section addresses these questions and is based on the work of Feige [22] proving that (for any \( \epsilon > 0 \) no polynomial-time approximation of the set cover problem can attain a performance ratio of \( (1 - \epsilon) \ln m \) unless \( \text{NP} \subset \text{DTIME}(m^{O(\log m)}) \). Our proof differs from the more general hardness result of [32] for the test cover problem because of the constraints imposed by the (undirected) graph structure on which identifying codes are defined (rather than the arbitrary “tests” permitted in the test cover problem). We state the main theorem here. The quite involved proof can be found in [2, 4].

**Theorem 4** For any \( \epsilon > 0 \) the identifying code problem has no polynomial time approximation with performance ratio \( (1 - \epsilon) \ln n \) (for directed and undirected graphs) unless \( \text{NP} \subset \text{DTIME}(n^{O(\log n)}) \).

Although Feige’s result is most appealing due to its tightness, other set cover hardness results can be applied too, e.g., [26], that have weaker complexity assumptions (\( P \neq \text{NP} \)).

The approximations suggested in the previous sections are roughly within a factor of 2 from the achievability bound of Theorem 4. We next show that this bound is also tight for non-robust identifying codes.

As mentioned in the introduction, the non-robust identifying codes problem is actually a special case of the test cover problem [31]. Recall that a test cover problem asks to find the smallest set \( T \) of tests \( T_i \subset U \) such that any pair \( x, y \in U \) is distinguished by at least one test \( T_i \) (i.e., \( |\{x, y\} \cap T_i| = 1 \)). Then simply consider the base set to be the set of vertices of the graph, i.e., \( U = V \), and its outgoing balls as the collection of tests, \( T = \{B^-(v) \forall v \in V \} \). Each pair of vertices will be distinguished by a code if and only if the corresponding set of tests constitutes a test cover of \( (U, T) \). It follows that test cover approximations can be applied to produce “good” identifying codes. One such greedy approximation was recently devised by Berman et al. [23] using a modified notion of entropy as the local optimization measure. This greedy approximation was proven to have a performance ratio of

---

**Algorithm 4** Synchronous \( r \)-robust algorithm (rID–SYNC)

We start with a graph \( G \) and non-negative integer \( r \). The following distributed algorithm run at node \( v \in \bar{V} \) produces an \( r \)-robust identifying code.

Precompute
- Set: slot = sp, subslot = L, state = unassigned.
- Calculate the assignment slot.

Iterate: while state = unassigned and slot \( \geq s_t \) do,
- \( l = \text{random}[1, \ldots, L] \):
- if received assignment message, message then,
  - if hop(ms) \( < 4 \) forward ms with hop(ms) + .
  - Recalculate the assignment slot.
- else subslot = 1 and slot = as then,
  - state = assigned
  - Transmit ms with id(ms) = id(v), hop(ms) = 1.
1 + \ln n$, where $n$ is the number of elements in the basis set. Applying this algorithm to graphs of size $n$ guarantees identifying codes with the same performance ratio, closing the gap (up to a small constant) from the lower bound of Theorem 4.

Although this performance guarantee outperforms our set-cover based approximation, it is not obvious how to generalize the algorithm of Berman et al. to robust identifying codes.

V. RANDOM GRAPHS

The study of random graphs in the context of robust identifying codes is important since it can offer insights and provide random designs of such codes. In this section we extend the work in [13] to provide a threshold on the size of an $r$-robust identifying code in asymptotically large random graphs.

**Theorem 5** Let $G(n, p) = (V, E)$ be a random graph of $n$ vertices where the existence of an edge between any pair of distinct vertices is determined independently and in random with probability $p$. Let $r \leq O(\log \log \log n)$, and $q = p^2 + (1 - p)^2$ such that $1 - q \geq \Omega \left( \frac{\log n}{n} \right)$. Then asymptotically in $n$ and for an arbitrary small $\epsilon > 0$ any subset of size

$$c \geq \frac{2 \log n + (2r - 1 + \epsilon) \log \log n}{\log 1/q}$$

is almost surely an $r$-robust identifying code.

The proof of this theorem is omitted due to space limitations and can be found in [4].

Recall that Moncel et al. showed in [13] that any subset of size $(2r + 1) \log n \log 1/q$ is almost surely an identifying code. Theorem 5 implies that for a small $r$ a relatively small addition of $(2r - 1) \log \log n$ vertices almost surely makes the code $r$-robust. Theorem 5 also provide means of random design of $r$-robust identifying codes. Figure 4 shows a numerically derived upper bound on the probability that a set of the size of Theorem 5 is not $r$-robust, for random graph of 10000 vertices and $r \leq 5$. A simple trial and error approach can be used to generate good $r$-robust identifying codes.

**Figure 4.** An upper bound on the probability that a set of vertices of the size of Theorem 5 is not a robust identifying code.

In [13] it was also shown that the threshold size of an identifying code is tight in the sense that all slightly smaller subsets are almost surely not identifying codes. Clearly, this result also holds for robust identifying codes. It is not obvious if this bound can be tightened for robust identifying codes.

VI. SIMULATIONS

Here we have simulated the centralized, localized and synchronized distributed identifying code algorithms, and applied them to grids, random graphs with different edge probabilities, and to geometric random graphs with different nodes densities. As a performance measure, we use the averaged size of the identifying code. For the case of $r = 0$ (i.e., non-robust identifying code) the simulation results are compared to the algorithm suggested by Ray et al. (ID-CODE) in [7], and in addition, we show a combinatorial lower bound derived by Karpovsky et al. in [5]. We also show the asymptotic (in $n$ - the size of the graph) results of Moncel et al. [13] and of Theorem 5.

A. Infinite grids

A significant amount of the study efforts into identifying codes was focused on infinite grids (e.g., [5, 9]). Many of these studies use tilings to suggest constructive upper bounds on the size of an identifying codes. Table I summarizes the results of running the centralized approximation on different types of grids. In some cases these results come pretty close in terms of density to efficient constructions and bounds provided in [9].

**Table I**

| Density of Identifying Codes for Infinite Grids Taken from [9] vs. Simulation Results Using Centralized Approximation (in Brackets). |
|-----------------|-----------------|-----------------|-----------------|
| ID code         | square          | king            | triangular      |
| 1-Robust        | $\frac{21}{64}$ ($\frac{21}{64}$) | $\frac{21}{91}$ ($\frac{21}{71}$) | $\frac{21}{110}$ ($\frac{21}{101}$) |

In [4] we develop two branch and bound algorithms for finding optimal identifying codes, and extend one of them to $(r > 0)$-robust identifying codes.

Figure 5 shows two examples of the branch-and-bound algorithm results for an hexagonal and triangular finite grids.

**Figure 5.** Branch and bound algorithm results for 1-robust identifying codes for the hexagonal (left) and triangular (right) grids. The red circles represent codewords, and the tile is captured by a gray square.
The resulting 1-robust identifying code has a density which is slightly better than the best known constructions in Table I. In terms of run-time performance both algorithms experience exponential growth in the size of the problem. The same observation was made in [29] for a large variety of test-cover problems, suggesting that the both problems are “hard” for a broad family of problems.

B. (Geometric) random graphs

Figure 6(a) shows the theoretical lower bound and the results of the centralized greedy algorithm. It can be seen that a significant enhancement in performance over the algorithm devised by Ray et al. is achieved. It should be noted that as \( n \) grows the curves for basically any algorithm should converge to Moncel’s asymptotic result, as illustrated in Figure 7. Still, the convergence rate appears to be very slow, suggesting that for reasonably large networks there is a lot to gain from the suggested algorithms compared to the simple approach of arbitrarily picking a code, whose size satisfies the threshold number of [13].

Figure 6(b) shows the simulation results for the localized and distributed algorithms compared to the centralized one. Recall that the performance of the asynchronous algorithm, \( rID - \text{ASYNC} \), is identical to the localized approximation. It can be observed that the results of the localized algorithm nearly match the results of the centralized algorithm. Divergence is evident for low edge probabilities, where it is harder to find a dominating set. Recall that there is a tradeoff between performance and the runtime of the synchronized distributed algorithm, \( rID - \text{SYNC} \). The smaller the number of subslots parameter, \( L \), the shorter the runtime and the larger the degradation in performance due to unresolved ties. Degradation in performance is also more evident when ties are more likely to happen, i.e., when the edge probability is approaching 0.5.

The results of the centralized \( r \)-robust identifying code algorithm are shown in Figure 6(c) in comparison to the asymptotic bound of Theorem 5. The asymptotic bound is a lower bound on the size of any subset of vertices that with high probability is an \( r \)-robust identifying code, asymptotically in the number of vertices in the graph, \( n \), as long as \( r = O(\log \log \log n) \). Figure 6(c) demonstrates a good fit between the experimental results and the theoretical bound for \( r = 0, 1 \) but for \( r \geq 2 \) the bound is found to be too optimistic as its assumptions on \( r \) seem not to hold any more.

Figure 8 shows the codeword density for geometric random graphs using the localized and distributed approaches, and the fraction of such graphs admitting an identifying code. It also presents the largest fraction of indistinguishable nodes, i.e., nodes that share the same identifying set, obtained in the simulation. As can be seen, the localized and distributed approaches (with \( L = 10 \)) yield very similar code sizes. The fraction of graphs admitting identifying codes is rather small (less than half the graphs) even for high node densities, which corresponds to the recent theoretical analysis of these.
graphs [24]. However, the size of indistinguishable nodes set is still small, suggesting that the system’s reduction in resolution, i.e. the geometric size of the largest indistinguishable set, is not too high. It should be noted that approaches such that of ID-CODE [7] are not designed to cope with graphs which do not have identifying codes, resulting in a code of all vertices.

VII. CONCLUSION

In this paper we discussed robust identifying codes - an optimization problem that arises naturally in localization and identification problems in networks, and which is intimately linked to error-correcting codes. By linking robust identifying codes to well studied combinatorial problems we were able to derive provable good approximations and tight hardness of approximation results. Simulations show that these algorithms perform well on random graphs as well as two dimensional grids. We also provided theoretical results on random graphs, that suggest that with relatively small addition of codewords, and with high probability identifying codes become robust. However, recent studies show that most of the random graphs do not possess identifying codes when geographical structure is embedded into them. These graphs should be further explored in the context of codes that identify almost all vertices except to $cn$ few.

ACKNOWLEDGMENT

Many of the results in this work are the product of a fruitful collaboration, chiefly with Prof. Ari Trachtenberg, Prof. David Starobinski and others. This work was supported in part by NSF CAREER grant CCR-0133521 and NSF grants CNS-0435312 and CCF-0729158.

REFERENCES