Connecting Identifying Codes and Fundamental Bounds

Niloofar Fazlollahi, David Starobinski and Ari Trachtenberg

Dept. of Electrical and Computer Engineering
Boston University, Boston, MA 02215
Email: {nfazl,staro,trachten}@bu.edu

***Abstract*— We consider the problem of generating a *connected robust identifying code* of a graph, by which we mean a subgraph with two properties: (i) it is connected, (ii) it is robust identifying, in the sense that the (subgraph-) induced neighborhoods of any two vertices differ by at least $2r + 1$ vertices, where r is the robustness parameter. This particular formulation builds upon a rich literature on the identifying code problem but adds a property that is important for some practical networking applications. We concretely show that this modified problem is NP-complete and provide an otherwise efficient algorithm for computing it for an arbitrary graph. We demonstrate a connection between the the sizes of certain connected identifying codes and error-correcting code of a given distance. One consequence of this is that robustness leads to connectivity of identifying codes.**

I. Introduction

Although introduced only twelve years ago [1], identifying codes have been linked to a number of deeply researched theoretical foundations, including super-imposed codes [2], covering codes [1, 3], locating-dominating sets [4], and tilings [5–8]. They have also been generalized and used for detecting faults or failures in multi-processor systems [1], RF-based localization in harsh environments [9–11], and routing in networks [12].

Within a number of these contexts, particularly those related to networking, it is important for the codewords of an identifying code to be connected, meaning that it is possible to transmit packets between codewords without going through non-codewords. The reason for this additional requirement is that codewords generally correspond to active or otherwise privileged nodes in the network, and the use of non-codewords for communication introduces inefficiencies (energy, communication, congestion, etc.), an important issue that has not largely been considerd in previous works. As such, we consider the problem of generating a *connected* robust identifying code (CRIC) for an arbitrary graph, where *robust* identifying codes [10] are variants that maintain identifiability even in the presence of a limited number of topological distortions.

We produce bounds on the size of a CRIC, based on bounds on the parameters of error-correcting codes. For increasing robustness, these bounds converge to the size of the best robust idenifying codes, meaning that robustness leads to connectivity for these codes.

We begin in Section II with a review of the related literature. In Section III we formally describe identifying codes, followed by a brief review of existing algorithms for generating them. We outline a proof that this new problem is NP-complete in Section IV, and in Section V we propose an efficient approximation algorithm for its solution. Finally, in Section VI we apply coding-theoretic bounds to our algorithmic framework in order to provide bounds on the sizes of the best connected identifying codes.

II. Related work

There is extensive theoretical work on identifying codes in the literature. In [13, 14] identifying codes are proved to be NP-complete by reduction from the 3-satisfiability problem.

Karpovsky et al [1] provide information theoretic lower bounds on the size of identifying codes over generic graphs and some specific graph topologies like meshes. The works in [2, 15–17] derive upper/lower bounds on size of the minimum identifying codes, with some providing graph constructions based on relating identifying codes to superimposed codes. The work in [17] focuses on random graphs, providing probabilistic conditions for existence together with bounds.

Many variants of identifying codes are defined and studied in the literature: a *robust* identifying code [3, 10] is resilient to changes in the underlying graph, a $(1, l \geq 0)$-identifying code [2, 15] uniquely identifies any subset of at most l vertices, a ρ radius identifying code [1] uniquely identifies every vertex using the set of all codewords within distance ρ or less from the vertex, and a dynamic identifying code [3] is a walk whose vertices form an identifying code.

Identifying codes are also proposed for various applications. The authors in [10] suggest application of identifying code theory for indoor location detection. They introduce robust identifying codes which are supposed to remain functional in case of failure of a limited number of codewords. They also present a heuristic which creates robust identifying codes for an arbitrary graph. The work in [12] uses the same technique for indoor location detection, although the authors introduce a more efficient algorithm for generation of robust identifying codes. They also suggest an additional application of identifying codes for efficient sensor labeling for data routing in the underlying sensor network. Both references implicitly assume that a sensor network can route location detection data toward a sink, which is not satisfied in sensor networks where only vertices corresponding to codewords are active.

Since we will use the algorithms in [10, 12] for generating an identifying code, we will review their techniques in more detail in Section III-B.

The work in [18] studies the problem of sensor placement in a network which may be a water supply network or an air ventilation system with potential contamination source(s) such that the contamination source is identified under either of the following constraints:

- *sensor-constrained version* where the number of sensors is fixed and the identification time has to be minimized,
- *time-constrained version* where the identification time is limited and the number of sensors has to be minimized. The later version of this problem is shown to be a variant of the identifying code problem [19].

Our work in [20] also introduced the concept of a connected identifying code within its networking applications.

Identifying codes are also linked to superimposed codes [1, 2, 15–17], dominating sets [21], locating dominating sets [22], the set cover [19, 23] and the test cover problem [19, 23] and r-robust identifying codes are linked to error correcting codes with minimum Hamming distance $2r + 1$ [10] and the set r-multi-cover problem [23].

Of these, the locating dominating sets are closest in flavor to identifying codes, and indeed Suomela [21] links identifying codes and locating dominating sets to dominating sets and shows that it is possible to approximate both problems within a logarithmic factor, and that sub-logarithmic approximation ratios are intractable.

There is also considerable work regarding generation of dominating sets and connected dominating sets [24, 25], but these results do not apply directly to connected identifying codes, since not every dominating set is an identifying code. In other words, the optimal identifying code generally has larger cardinality than that of the optimal dominating set.

III. Background

In this section, first we formally describe identifying codes. Then we briefly review two existing algorithms that we use for generating connected identifying codes.

A. Definitions

Consider a graph G with a set of vertices V and a set of edges E such that every vertex in V is designated either a *codeword* or a *non-codeword*, the set of codewords being denoted $I\ subseteqV$. An *identifying set* for vertex $v \in V$ is the set of all codewords that are within distance one from v (this includes node v itself and all of its neighbors). If the identifying set for every vertex is unique and non empty, we call I an *identifying code*. It is also the case that every super-set of I is an identifying code [10].

It is not difficult to verify that for the graph and codewords shown in Figure 1, the identifying set for every vertex of the graph is unique, i.e., the identifying set for vertex a is $\{a\}$, for vertex b is $\{a, c\}$ and so on. The location of a target can be identified at every region using a look up table that maps identifying sets to vertex IDs.

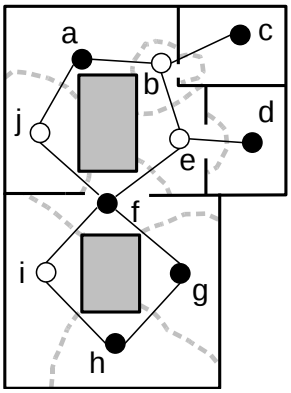

Fig. 1. An example building floor plan and connectivity graph of sensors located at positions marked by circles. The filled circles represent codewords of an identifying code for the sensor network connectivity graph. The dashed lines show the boundaries of distinguishable regions based on the radio range of the active sensors.

An identifying code I over a given graph $G(V, E)$ is said to be r-robust if it remains an identifying code after we arbitrarily add or remove up to r vertices in V to (or from) every identifying set, i.e. $S_I(u)\Delta V_1 \neq S_I(v)\Delta V_2$ for every $u \neq v \in V$ and every $V_1, V_2 \subset V$ such that $|V_1|, |V_2| \leq r$, where operator Δ is the (set) symmetric difference operator. The *minimum symmetric difference* of an identifying code I, $d_{min}(I)$, is defined to be the minimum symmetric difference between every pair of identifying sets, i.e. $d_{min}(I) = \min_{u \neq v \in V} |S_I(u)\Delta S_I(v)|$. It is shown in [10] that an identifying code I is r-robust if and only if $d_{min}(I) \geq 2r+1$, and that every super-set of an r-robust identifying code I is an r-robust identifying code.

B. Existing algorithms

Next, we briefly review two existing algorithms that have polynomial complexity in terms of graph size and generate an identifying code for an arbitrary graph if one exists [10, 12]. We refer the reader to the cited references for further details.

Algorithm `ID-CODE` introduced in [10] initially assigns all vertices V in the input graph as codewords, and then checks, one by one, whether each vertex can be removed from the code without losing the identifying property. This greedy algorithm produces an *irreducible* code, meaning that no codeword may be removed from it while still keeping it an identifying code. The algorithm can be trivially modified to yield r-robust codes by changing the greedy criterion accordingly.

Algorithm `rID`, presented in [12] initially calculates the identifying set of every vertex, assuming that all vertices are codewords. It associates with every vertex v in V the set of vertex pairs which it can identify, i.e., one vertex in the pair is adjacent to v and the other is not. The algorithm iteratively forms an identifying code by selecting the vertex that identifies the most pairs. Using a similar approximation to the *set cover* problem [26], the authors in [12, 23] prove that `rID` achieves a logarithmic approximation ratio upper bounded by $c_1 \ln |V|$ and lower bounded by $c_2 \ln |V|$ for some constants $c_1 > c_2 > 0$. They also show that this bound is tight

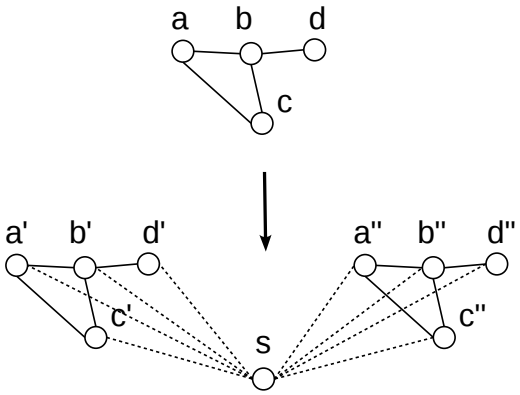

Fig. 2. Graph G with four vertices on top and constructed graph G_t with nine vertices. Vertex s connects vertices a', b', c' and d' in subgraph G' and vertices a'', b'', c'' and d'' in subgraph G'' by edges that are shown dashed.

unless $NP \subset DTIME(n^{\lg \lg n})$. A robust version of `rID` is also presented in [12] using a reduction to the set multi-cover problem [26].

IV. NP-COMPLETENESS

Theorem 4.1: Given any graph G and an integer k, the decision problem of the existence of a connected identifying code with cardinality at most k in G, is NP-complete.

PROOF. Polynomial verification of a connected identifying code solution over any given graph G is straightforward. The rest of the proof is based on a reduction from the identifying code problem, whose decision instance has been shown to be NP-complete [13, 14].

Consider the problem of existence of an identifying code I with cardinality at most k over any given graph $G(V, E)$. We construct a graph $G_t(V_t, E_t)$ from G as follows:

- We construct two graphs $G'(V', E')$ and $G''(V'', E'')$ as direct copies of G.
- We add a vertex s to G_t such that s connects all vertices V' and all vertices V''.

Clearly the transformation from G to G_t is polynomial and takes $O(4|V| + 2|E|)$ time since $|V_t| = 2|V| + 1$ and $|E_t| = 2|E| + 2|V|$. Figure 2 demonstrates our construction for a sample instance of G.

A careful analysis, which has been omited for sake of brevity, can show that there exists an identifying code with cardinality at most k in G if and only if there exists a connected identifying code with cardinality at most $2k + 1$ in G_t.

□

V. ALGORITHM `ConnectID`

A. Model and notation

For our purposes, we assume an undirected connected graph $G(V, E)$ (or G in short). The *redundancy ratio* of a connected identifying code I_c and a subset identifying code $I \subseteq I_c$ is defined as $R = |I_c|/|I| \geq 1$, with equality meaning that I is connected.

A *maximal component of connectivity* (or a component in short) C of I in graph G is a subset of codewords in I such that the subgraph of G induced by this subset is connected, and any superset of C is not connected. For the example of Figure 1, we have $I = \{a, c, d, f, g, h\}$ with components of connectivity $C_1 = \{a\}$, $C_2 = \{c\}$, $C_3 = \{d\}$ and $C_4 = \{f, g, h\}$.

A *plain path* between components C_1 and C_2 is an ordered subset of non-codeword vertices in V that forms a path connecting a vertex $x_1 \in C_1$ to a vertex $x_2 \in C_2$. By distinction, a *path* may include codewords or non-codewords. As an example, in Figure 1, $\{a, b, e, f\}$ and $\{a, j, f\}$ are the only plain paths between components C_1 and C_4. The path $\{a, j, f, e, d\}$ is not a plain path between C_1 and C_3 because f is a codeword.

The distance between a given pair of components, say C_1 and C_2, denoted $\texttt{dist}(C_1, C_2)$, is defined to be the number of edges on the shortest plain path between C_1 and C_2. If there is no plain path between C_1 and C_2, then $\texttt{dist}(C_1, C_2) = \infty$. As an example, in Figure 1, $\texttt{dist}(C_1, C_2) = 2$, $\texttt{dist}(C_1, C_3) = 3$ and $\texttt{dist}(C_1, C_4) = 2$.

B. Algorithm description

We present algorithm `ConnectID` in the format of a function which receives the set of codewords of an identifying code I for a given graph G and returns the set of codewords of a connected identifying code I_c. First, we present algorithm `ConnectID` informally:

In the initialization phase, function $\texttt{ConnectID}(G, I)$ partitions the identifying code I into a set of N distinct components of connectivity $\{C_1, C_2, ..., C_N\}$ where $1 \leq N \leq |I|$. Note that our standing assumption that G is connected implies that every pair of components is connected by some path in G.

We define C to be a set that stores the growing connected identifying code, initialized to the set of codewords in one of the components, say C_1. We shall use $\widehat{C}$ to denote the set of all components whose codewords are *not yet included* in C, meaning that $\widehat{C}$ can be initialized to $\{C_2, ..., C_N\}$.

At every iteration, we first update the distance $\texttt{dist}(C, C_j)$ between C and every component C_j in $\widehat{C}$. Then, we extract from $\widehat{C}$ the component C^* with minimum $\texttt{dist}(C, C^*)$ (breaking ties arbitrarily). We assign as codewords all vertices on the shortest plain path connecting C and C^* denoted $\texttt{path}^*(C, C^*)$, and unite the codewords in C and C^* and $\texttt{path}^*(C, C^*)$ into C. After this step, we examine if there are any other components in $\widehat{C}$ which become connected to C via the newly selected codewords on $\texttt{path}^*(C, C^*)$. We define $\Gamma \subseteq \widehat{C}$ to be the set of such components. If Γ is non-empty, we unite C with the components in Γ and extract them from $\widehat{C}$. The iteration above is repeated until $\widehat{C}$ becomes empty and the function returns returns the connected identifying code $I_c = C \supseteq I$.

More formally: **Algorithm** $\texttt{ConnectID}(G, I)$**:**
Initialization:

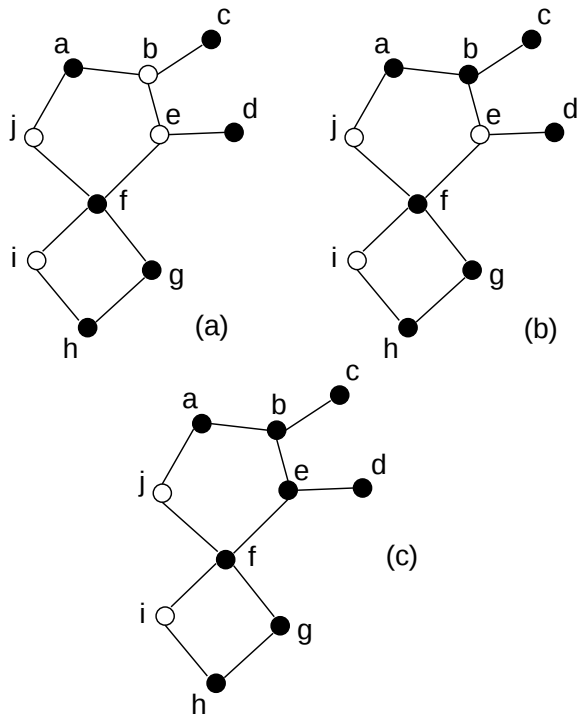

Fig. 3. Progress of $\texttt{ConnectID}(G, I)$. The filled circles represent codewords of an identifying code I for the illustrated graph G (a) initially, I is partitioned to components $C_1 = \{a\}$, $C_2 = \{c\}$, $C_3 = \{d\}$ and $C_4 = \{f, g, h\}$. We set $C = \{a\}$ and $\widehat{C} = \{C_2, C_3, C_4\}$ (b) $C = \{a, b, c\}$ and $\widehat{C} = \{C_3, C_4\}$ (c) $C = \{a, b, c, d, e, f, g, h\}$ and $\widehat{C} = \{\}$.

1) Partition I into a unique set of components of connectivity $\{C_1, C_2, ..., C_N\}$ where $1 \leq N \leq |I|$.
2) Set $\widehat{C} \leftarrow \{C_2, ..., C_N\}$.
3) Set $C \leftarrow C_1$.

Iteration:

7) While $\widehat{C}$ is not empty,
8) Update $\texttt{dist}(C, C_j)$ and $\texttt{path}(C, C_j)$ for every $C_j \in \widehat{C}$ and set $C^* \leftarrow \arg\min_{C_j \in \widehat{C}} \texttt{dist}(C, C_j)$.
9) Extract component C^* from $\widehat{C}$.
10) Set $C \leftarrow C \cup C^* \cup \texttt{path}^*(C, C^*)$.
11) Find the set $\Gamma \subseteq \widehat{C}$ of components that are connected to C.
12) If Γ is not empty,
13) For every component $C_j \in \Gamma$,
14) Extract C_j from $\widehat{C}$.
15) Set $C \leftarrow C \cup C_j$.
16) Return $I_c \leftarrow C$.

Example. Figure 3 shows the progress of $\texttt{ConnectID}(G, I)$ for the same graph and the same input identifying code as shown in Figure 1. In this example, we initialize the algorithm having components

$$C_1 = \{a\}$$
$$C_2 = \{c\}$$
$$C_3 = \{d\}$$
$$C_4 = \{f, g, h\},$$

This way $C = C_1$ and $\widehat{C} = \{C_2, C_3, C_4\}$ as shown in Figure 3(a). At first iteration, after we calculate the distance between C and all components in $\widehat{C}$ at line 8, giving:

$$\texttt{dist}(C, C_2) = 2$$
$$\texttt{dist}(C, C_3) = 3$$
$$\texttt{dist}(C, C_4) = 2.$$

At line 9, we extract one component with minimum `dist` from $\widehat{C}$, which may be C_2 or C_4 (we arbitrarily choose C_2). Then, we unite C and C_2 and vertex b at line 10. Hence, $C = \{a, b, c\}$ as illustrated in figure 3(b). There are no components in $\widehat{C}$ that are connected to C at this stage, i.e. $\Gamma = \{\}$, and we return back to line 7. We update distances and paths again: $\texttt{dist}(C, C_3) = 2$ and $\texttt{dist}(C, C_4) = 2$. We extract the component with minimum `dist`, which may be C_3 or C_4 (we choose C_3). We unite C and C_3 and vertex e and obtain $C = \{a, b, c, d, e\}$. Then, we examine the only component remaining in $\widehat{C}$ which is C_4 to see if it is now connected to C. We get $\Gamma = C_4$ and we unite C and C_4 at line 15. Finally, in figure 3(c) we have $C = \{a, b, c, d, e, f, g, h\}$ which is the connected identifying code I_c output by the algorithm.

Algorithm `ConnectID` resembles the Prim's algorithm for constructing the minimum spanning tree of a graph [27], but has some fundamental differences in the manner in which it grows the connected code and how it handles the connected components.

C. Performance analysis

The correct functioning of `ConnectID` is based upon two fundamental properties of identifying codes.

Lemma 5.1: Consider any identifying code I that is partitioned into a set of components of connectivity $P = \{C_1, ..., C_{|P|}\}$ over graph G. If $|P| > 1$, then every component $C_i \in P$ is at most three hops away from *some other* component $C_j \in P$.

Lemma 5.2: If vertex v is adjacent to component $C_i \in P$ and $|C_i| = 1$, then v is necessarily adjacent to *some other* component $C_j \in P$.

As a straightforward corollary,

Corollary 5.3: If $|P| > 1$, then every component $C_i \in P$ with $|C_i| = 1$ is at most two hops away from some component $C_j \in P$ with $j \neq i$.

Lemmae 5.1 and 5.2 hold for every identifying code I over graph G, and specifically right after the initialization of algorithm `ConnectID`. Since at every iteration, we add one or more codewords and do not remove any codeword, the set of codewords in C and in every component of $\widehat{C}$ forms an

identifying code. Hence, Lemmas 5.1 and 5.2 invariably hold after every iteration.

The overall analysis of our algorithm is summarized in the following theorem, which is based on Lemmae 5.1 and 5.2.

Theorem 5.4: Given an identifying code I on graph G and $I_c = \texttt{ConnectID}(G, I)$, then

i) I_c is a connected identifying code.
ii) $I_c \supset I$
iii) $|I_c| \leq 2|I| - 1$, where $-$ denotes set difference. This bound is tight.

We omit most of the proof of the theorem for sake of brevity, but the tightness of its bound iii follows from consideration of a ring topology with $2k$ vertices, for a positive integer k. The optimal identifying code for such a graph consists of k maximally separated vertices (i.e. every other vertex is in the code), whereas the connected identifying code for this graph must necessarily contain all but one vertex (i.e. $2k-1$ in total).

Corollary 5.5: For $I_c = \texttt{ConnectID}(G, I)$, the redundancy ratio

$$R = |I_c|/|I| \leq 2.$$

If the input identifying code I to $\texttt{ConnectID}(G, I)$ is an identifying code achieved by the algorithm in [12], then we have $|I| \leq c\ |I^*| \ln |V|$ where $c > 0$ is a constant, I^* is the identifying code with minimum cardinality for graph G and $|V|$ is the number of vertices in graph G. We define I_c^* to be the connected identifying code with minimum cardinality in graph G. Since $|I_c^*| \geq |I^*|$, we have the following corollary.

The following corollary follows from applying ConnectID to the the identifying code produced by the algorithm in [12], and carrying over its approximation guarantee.

Corollary 5.6: For an optimal connected code I_c^* and an identifying code $I_\approx$ produced by [12]:

$$\frac{\texttt{ConnectID}(G, I_\approx)}{I_c^*} \leq k\ |I_c^*| \ln |V|,$$

for a constant $k > 0$.

VI. Bounds

The properties of ConnectID ensure that it produces a connected *robust* code if it is given a robust code as an input. In this section, we combine the results of the algorithm with well-known coding theoretic bounds to derive bounds on the sizes of connected robust identifying codes. We show that as robustness increases, the resulting codes are increasingly connected.

Recall our notation that an r-robust identifying code I over graph G can be partitioned into connected components $P = \{C_1, ...C_{|P|}\}$. We define $S_{\min}$(I) (or just $S_{\min}$in context) to be the minimum non-unitary size of a connected component:

$$S_{\min}(I) = \min_{j \text{ s.t. } |C_j|>1} |C_j|.$$

Our upper bound on the cardinality of I_c depends on $S_{\min}$, for which we shall provide bounds later in this section.

Theorem 6.1: The connected identifying code $I_c = \texttt{ConnectID}(G, I)$ produced by our algorithm from an r-robust identifying code I satisfies

$$|I_c| \leq (1 + \frac{2}{S_{min}})\ |I| - \frac{2}{S_{min}}.$$

The proof of the theorem is based on the following lemma.

Lemma 6.2: Given an $r \geq 1$-robust identifying code I with connected components $P = \{C_1, ..., C_{|P|}\}$, there may be at most one component C_i with cardinality one.

Proof. [Theorem 6.1] If I is already connected, the bound follows trivially. Otherwise, there are at least two components, so that $\widehat{C}$ and C in the ConnectID algorithm are initially not empty and, per Lemma 6.2, there is at most one component with cardinality one.

Three scenarios are possible:

(i) Component C is initialized to the only component with cardinality one. In this case, every component in $\widehat{C}$ has cardinality at least $S_{\min}$and there are $|I| - 1$ codewords not in C. Hence, $\widehat{C}$ contains at most $(|I| - 1)/S_{\min}$ components initially. Using a similar reasoning in Theorem 5.4 based on Lemma 5.1, ConnectID adds at most two codewords per every component that is initially in $\widehat{C}$, and the bound follows.

(ii) There is a component with cardinality one in $\widehat{C}$ at initialization. In this case, there are at most $|I|-S_{\min}$ codewords not in C initially. We add at most one codeword for the component with cardinality one in $\widehat{C}$ based on Lemma 5.2. There are at most $(|I| - S_{\min} - 1)/S_{\min}$ other components in $\widehat{C}$ initially. Therefore, we add at most $2(|I| - S_{\min} - 1)/S_{\min}$ codewords plus one codewords for the component with cardinality one to $|I|$, leading to the desired bound.

(iii) There is no component with cardinality one. In this case, there are at most $(|I|-S_{\min})/S_{\min}$ components in $\widehat{C}$ initially, and we add at most two codewords per every component in $\widehat{C}$ leading to the bound.

□

The value $S_{\min}$ is lower bounded by the minimum size of an r-robust identifying code with more than one codeword, which, in turn, is related to the size of a minimum error-correcting code, as expressed in the following lemma. Recall that the characteristic vector of a set is the binary vector whose i-th bit is 1 if and only if the i-th element of a given universe (in this case, the set of vertices in the graph) is in the set.

Lemma 6.3: The characteristic vectors of the identifying sets of an r-robust identifying code I form a binary r-error correcting code of length $|I|$. The reverse does not necessarily hold.

With the aid of Lemma 6.3, we can form a relationship between $S_{\min}$ and the coding-theoretic function $A(n, d)$ denoting the maximal size of a (binary) code of length n and minimum distance d. This leads us to our theorem linking bounds on connected identifying codes and error-correcting codes.

Theorem 6.4: Given the upper bound $f(n, d)$ on the maximum size $A(n, d)$ of binary code of length n and (odd) minimum distance d, and *any* $\frac{d-1}{2}$-robust identifying code I:

$$S_{\min}(I) \geq \arg\min_{q>1}(q \leq f(q, d)).$$

PROOF. The proof is based on the following relation between identifying codes and error correcting codes. For any given $r \geq 0$-robust identifying code I with n codewords over an arbitrary graph G, we know from [10] that $d_{min}(I) \geq 2r+1$, meaning that the characteristic vectors of at least n identifying sets are at Hamming distance $d = 2r+1$ from one another. In other words, an r-robust identifying code with n codewords over any given graph G exists *only if* an r-error correcting code exists with length n and $A(n, d = 2r+1) \geq n$ codewords.

Let $q_{\min} = \arg\min_{q>1}(q \leq f(q,d))$. If $q_{\min} = 2$, then $S_{\min} \geq q_{min}$ trivially since $S_{\min}$ is an integer strictly larger than 1. Otherwise, for $q_{min} > 2$, it must be, by definition, that $q' > f(q', d) \geq A(q', d)$ for every q' such that $1 < q' < q_{min}$. In other words, for these q' there does not exist a r-error correcting code of length q' with q' codewords, and, by the above logic, there does not exist an r-robust identifying code with q' codewords (over a graph with q' vertices). This means that $S_{\min} > q'$ for all $1 < q' < q_{min}$, or, more succinctly, $S_{\min} \geq q_{\min}$, proving the theorem. □

Having established a connection between connected identifying codes and error-correcting codes, we can now link Theorem 6.4 with Theorem 6.1 with the well-known bounds on $A(n, d)$ from the coding theory literature.

The following bound is based on the Singleton bound [28] that $A(n, d) \leq 2^{n-d+1}$

Corollary 6.5:

$$S_{\min} \geq -\frac{W_{-1}(-2^{-2r}\ln 2)}{\ln 2} \approx d - 1 + \log_2(d-1),$$

where $W_{-1}(\cdot)$ is the negative branch of the Lambert W function [29].

PROOF. By the Singleton bound applied to Theorem 6.4,

$$S_{\min} \geq \arg\min_{q>1}\left(q \leq 2^{q-2r}\right).$$

Minimizing the right hand side is equivalent to minimizing (for $q > 1$):

$$q2^{-q} \leq 2^{-2r}$$
$$(-q\ln 2)e^{(-q\ln 2)} \geq -2^{-2r}\ln 2.$$

Since $-q\ln 2 \leq -1$, the solution involves the negative branch of the Lambert W function, giving

$$(-q\ln 2) \leq W_{-1}(-2^{-2r}\ln 2)$$
$$q \geq -\frac{W_{-1}(-2^{-2r}\ln 2)}{\ln 2}.$$

Using the three most significant terms of a series expansion for W_{-1}[30], namely $W_{-1}(z) \approx ln(-z) - ln(-ln(-z)) + \frac{ln(-ln(-z))}{ln(-z)}$ the theorem is proved. □

One can similarly apply other coding-theoretic upper bounds, such as the Hamming bound [28] to get different lower bounds for $S_{\min}$. Using some best known codes, we can get a good result with simpler exposition.

Lemma 6.6: $S_{\min}(I) \geq d + 3$.

PROOF.

We relate this analysis to r-error correcting codes as before. For robustness $r = 1$, S_{min} is at least six because no binary code construction with length $n = 5$ exists that achieves at least five identifying sets[1] with minimum Hamming distance $d = 3$, i.e., $A(5, 3) < 5$ [28]. For code length $n = 6$, we have $A(6, 3) \geq 6$. For example the lexicode [28, 31] has 8 identifying sets. This implies $S_{min} \geq 6$ for $r = 1$.

For a lower bound on $S_{\min}$ (for $r > 1$), let us assume, for sake of argument, that it is impossible to produce q different identifying sets at distances $2r+1$ from one another using q codewords, i.e., $A(q, 2r+1) < q$. This implies that in every collection of q identifying sets built by q codewords, there is at least a pair of identifying sets with Hamming distance of at most $2r$. For every additional codeword we can increase the Hamming distance between every pair of identifying sets by at most one. Therefore, in every collection of q identifying sets built by $q+2$ codewords, there is at least a pair of identifying sets with Hamming distance of at most $2r+2$, i.e., $A(q+2, 2r+3) < q < q+2$. Hence, it is *necessary* to add at least two codewords for every additional degree of robustness. The lemma follows by induction starting with $S_{\min} \geq 6$ for $r = 1$ and adding two codewords to the lower bound for every increment in robustness. □

Combining Theorem 6.1 with Lemma 6.6 we have the following simple bound on the size of a connected code generated by our algorithm.

Theorem 6.7: Given an $r \geq 1$-robust identifying code I,

$$|\texttt{ConnectID}(G, I)| \leq (1 + \frac{1}{r+2})\,|I| - \frac{1}{r+2}.$$

Tables I and II summarize the lower bounds on $S_{\min}$ and the corresponding upper bounds on the redundancy ratio R for a few values of robustness r based on the best codes known in Appendix A of [28], Lemma 6.6, the Singleton bound and the Hamming bound.

TABLE I

LOWER BOUND ON S_{MIN} AND UPPER BOUND ON REDUNDANCY RATIO R OF CONNECTED IDENTIFYING CODE I_c VS. ROBUSTNESS r FOR BEST CODES KNOWN AND FROM LEMMA 6.6.

	best codes known		Lemma 6.6	
r	$S_{\min}$	Redundancy ratio	$S_{\min}$	Redundancy ratio
1	6	$4/3$	6	$4/3$
2	10	$6/5$	8	$5/4$
3	14	$8/7$	10	$6/5$
4	18	$10/9$	12	$7/6$

We observe that with increase of r, $S_{\min}$ increases and the upper bound on $|I_c|$ becomes closer to $|I|$. This implies for larger robustness r, I tends to be more connected and we usually require fewer additional codewords to make it connected. Furthermore, according to corollary 6.7 for large values of robustness r, $|I_c|$ tends to $|I|$.

[1]These are called *codewords* in the context of coding theory, but we maintain the identifying set terminology for sake of consistency.

TABLE II

LOWER BOUND ON S_{MIN} AND UPPER BOUND ON REDUNDANCY RATIO R OF CONNECTED IDENTIFYING CODE I_c VS. ROBUSTNESS r USING THE SINGLETON BOUND AND THE HAMMING BOUND.

	Singleton bound		Hamming bound	
r	S_{min}	Redundancy ratio	S_{min}	Redundancy ratio
1	4	$3/2$	5	$7/5$
2	7	$9/7$	9	$11/9$
3	10	$6/5$	12	$7/6$
4	12	$7/6$	15	$17/15$

VII. CONCLUSION

In this work we have developed an approach for generating connected robust identifying codes from a given robust identifying code for a graph. This problem has practical significance in the application of identifying codes to networking problems, such as routing, but it also provides connections with the wealth of coding theory literature. Most notably, we show how to use coding-theoretic bounds to provide bounds on connected codes, with the interesting realization that increasing robustness leads to increasing connectivity.

ACKNOWLEDGMENT

This work was supported in part by the US National Science Foundation under grants under grants CCF-0729158, CCF-0916892, and CNS-1012910.

REFERENCES

[1] M. G. Karpovsky, K. Chakrabarty and L. B. Levitin, "On a new class of codes for identifying vertices in graphs," *IEEE Transactions on Information Theory*, vol. **44**, no. 2, pp. 599–611, March 1998.

[2] S. Gravier and J. Moncel, "Construction of codes identifying sets of vertices," *The Electronic Journal of Combinatorics*, vol. **12**, no. 1, 2005.

[3] I. Honkala, M. Karpovsky and L. Levitin, "On robust and dynamic identifying codes," *IEEE Transactions on Information Theory*, vol. **52**, no. 2, pp. 599–612, February 2006.

[4] I. Honkala and A. Lobstein, "On identifying codes in binary hamming spaces," *Journal of Comb. Theory, Series A*, vol. 99, no. 2, pp. 232–243, 2002.

[5] G. Cohen, I. Honkala, A. Lobstein, and G. Zémor, "New bounds for codes identifying vertices in graphs." *Electr. J. Comb.*, vol. 6, 1999.

[6] I. Charon, I. Honkala, O. Hudry, and A. Lobstein, "General bounds for identifying codes in some infinite regular graphs." *Electr. J. Comb.*, vol. 8, no. 1, 2001.

[7] I. Charon, O. Hudry, and A. Lobstein, "Identifying codes with small radius in some infinite regular graphs." *Electr. J. Comb.*, vol. 9, no. 1, 2002.

[8] I. Charon, I. Honkala, O. Hudry, and A. Lobstein, "The minimum density of an identifying code in the king lattice." *Discrete Mathematics*, vol. 276, no. 1-3, pp. 95–109, 2004.

[9] S. Ray, R. Ungrangsi, F. De Pellegrinin, A. Trachtenberg and D. Starobinski, "Robust location detection in emergency sensor networks," *Proc. INFOCOM*, April 2003.

[10] S. Ray, D. Starobinski, A. Trachtenberg and R. Ungrangsi, "Robust location detection with sensor networks," *IEEE JSAC (secial Issue on fundamental performance limits of wireless sensor networks)*, vol. **22**, no. 6, pp. 1016–1025, August 2004.

[11] G. Mao and B. Fidan, Eds., *Localization Algorithms and Strategies for Wireless Sensor Networks: Monitoring and Surveillance Techniques for Target Tracking*, 2009, ch. Robust Localization Using Identifying Codes, pp. 321–347.

[12] M. Laifenfeld, A. Trachtenberg, R. Cohen and D. Starobinski, "Joint monitoring and routing in wireless sensor networks using robust identifying codes," *Springer Journal on Mobile Networks and Applications (MONET)*, vol. **14**, no. 4, pp. 415–432, August 2009.

[13] I. Charon, O. Hudry and A. Lobstein, "Identifying and locating-dominating codes: NP-completeness results for directed graphs," *IEEE Transactions on Information Theory*, vol. **48**, no. 8, pp. 2192–2200, August 2002.

[14] ——, "Minimizing the size of an identifying or locating-dominating code in a graph is NP-hard," *Theoretical Computer Science*, vol. **290**, no. 3, pp. 2109–2120, 2003.

[15] S. R. T. Laihonen, *Codes identifying sets of vertices*. Berlin, Germany: Springer-Verlag, 2001, vol. 2227, in Lecture Notes in Computer Science.

[16] . Ben-Haim and S. Litsyn, "Exact minimum density of codes identifying vertices in the square grid," *SIAM Journal on Discrete Mathematics*, vol. **19**, no. 1, pp. 69–82, 2005.

[17] A. Frieze, R. Martin, J. Moncel, M. Ruszink and C. Smyth,, "Codes identifying sets of vertices in random networks," *Discrete Mathematics*, vol. **307**, no. 9-10, pp. 1094–1107, May 2007.

[18] T. Berger-Wolf, W. Hart and J. Saia, "Discrete sensor placement problems in distribution networks," *Mathematical and Computer Modelling*, vol. **42**, no. 13, pp. 1385–1396, December 2005.

[19] M. Laifenfeld, A. Trachtenberg, and T. Berger-Wolf, "Identifying codes and the set cover problem," in *Proceedings of the 44th Annual Allerton Conference on Communication, Control, and Computing*, September 2006.

[20] N. Fazlollahi, D. Starobinski, and A. Trachtenberg, "Connected identifying codes for sensor network monitoring," in *IEEE WCNC*, 2011.

[21] J. Suomela, "Approximability of identifying codes and locating-dominating codes," *Information Processing Letters*, vol. **103**, no. 1, pp. 28–33, 2007.

[22] I. Honkala and A. Lobstein, "On identifying codes in binary hamming spaces," *Journal of Combinatorial Theory, Series A*, vol. **99**, no. 2, pp. 232–243, August 2002.

[23] M. Laifenfeld and A. Trachtenberg, "Identifying codes and covering problems," *IEEE Transactions on Information Theory*, vol. **54**, no. 9, pp. 3929–3950, September 2008.

[24] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," *Algorithmica*, vol. **20**, no. 4, pp. 374–387, April 1998.

[25] J. Blum, M. Ding, A. Thaeler and X. Cheng, *Connected dominating set in sensor networks and MANETs*. Kluwer Academic Publishers, in: Du D-Z, Pardalos P (eds) Handbook of combinatorial optimization.

[26] U. Feige, "A threshold of ln n for approximating set cover," *Journal of the ACM*, vol. **45**, no. 4, pp. 634–652, 1998.

[27] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 1990.

[28] F. J. Macwilliams and N. J. A. Sloane, *The theory of error-correcting codes*. The Netherlands: North-Holland, 1977, vol. 16.

[29] R. Corless, G. Gonnet, D. Hare, D. Jeffrey, and D. Knuth, "On the lambertw function," *Advances in Computational Mathematics*, vol. 5, pp. 329–359, 1996, 10.1007/BF02124750. [Online]. Available: http://dx.doi.org/10.1007/BF02124750

[30] F. Chapeau-Blondeau and A. Monir, "Numerical evaluation of the lambert w function and application to generation of generalized gaussian noise with exponent 1/2," *Signal Processing, IEEE Transactions on*, vol. 50, no. 9, pp. 2160 – 2165, Sept. 2002.

[31] A. Trachtenberg, "Designing lexicographic codes with a given trellis complexity," *IEEE Trans. on Info. Theory*, vol. 48, no. 1, pp. 89–100, January 2002.