Joint Monitoring and Routing in Wireless Sensor Networks using Robust Identifying Codes

Moshe Laifenfeld*, Ari Trachtenberg*, Reuven Cohen* and David Starobinski*
*Department of Electrical and Computer Engineering
Boston University, Boston, MA 02215
Email:{ moshel,trachten,cohenr,staro@bu.edu}

***Abstract*— Wireless Sensor Networks (WSNs) provide an important means of monitoring the physical world, but their limitations present challenges to fundamental network services, such as routing. In this work we utilize an abstraction of these networks based on the theory of identifying codes. This abstraction has been useful in recent literature for a number of important monitoring problems, such as localization and contamination detection. In our case, we use it to provide a joint infrastructure for efficient and robust monitoring and routing in WSNs. Specifically, we provide an efficient and distributed algorithm for generating robust identifying codes with a logarithmic performance guarantee based on a novel reduction to the set k-multicover problem; to the best of our knowledge, this is the first such guarantee for robust identifying codes. We also show how to reuse this same identifying-code infrastructure for near-optimal routing with very small routing tables. We provide various experimental results to demonstrate the benefits of our approach over previous identifying code approaches (for monitoring) and shortest path solutions (for routing).**

I. Introduction

Sensor networks provide a new and potentially revolutionary means of reaching and monitoring our physical surroundings. Important applications of these networks include environmental monitoring of the life-cycle of trees and various animal species [1, 2] and the structural monitoring of buildings, bridges, or even nuclear power stations [3, 4].

A. Identifying code abstraction

For several fundamental monitoring problems, such as localization [5, 6] and identification of contamination sources [7, 8], the theory of *identifying codes* [9] has been found to provide an extremely useful abstraction. Under this abstraction, a monitoring area is divided into a finite number of regions and modeled as a graph, wherein each vertex represents a different region as in Figure 1. In this model, two vertices are connected by a link if they are within communication range. An identifying code for the graph then corresponds to a subset of vertices where monitoring sensors (*i.e., codewords* of the code) are located, such that each vertex is within the communication range of a different set of monitors (referred to as an *identifying set*). Thus, a collection of monitors in a network forms an identifying code if any given identifying set uniquely identifies a vertex in the graph.

An important benefit of identifying codes is that they allow monitoring of an area without the need to place or activate a (possibly expensive) monitor in each sub-region. Since the size of identifying code is typically much smaller than that of the original graph (*e.g.*, for very large random graphs of n vertices, the size of an identifying code is roughly $\log(n)$ [10]), this construction can result in a substantial savings of monitors. Alternatively, for a fixed number of monitors, systems based on identifying codes can achieve much higher resolution and robustness than proximity-based systems in which each sensor only monitors the region surrounding it.

Identifying codes provide also means to quantify energy/robustness trade-offs through the concept of *robust identifying codes*, introduced in [5]. An identifying code is r-robust if the addition or deletion of up to r codewords in the identifying set of *any* vertex does not change its uniqueness. Thus, with an r-robust code, a monitoring system can continue to function properly even if up to r monitors per locality experience failure. Of course, the size of an r-robust code increases with r (typically linearly).

Despite the importance of identifying codes for sensor monitoring applications, the problem of constructing efficient codes (in terms of size) is still unsolved. Specifically, the problem of finding a minimum identifying code for an arbitrary graph has been shown to be NP-hard [11, 12]. In Ray et al. [5], a simple algorithm called ID-CODE was proposed to generate irreducible codes in which no codeword can be removed without violating the unique identifiability of some vertex. However, in some graphs, the size of the resulting code using the ID-CODE algorithm can be arbitrarily poor [13]).

B. Related Work

Considerable work (see e.g., [14–16]) has been done on compact routing using dominating sets, and, in paticular connected dominating sets. Dominating sets are sets of nodes whose combined neighborhoods include all nodes in the graph. Therefore, if information for routing to each node in a dominating set is stored, any node can be reached through one of its neighbors in the dominating set with an almost optimal path length. Most works [14, 16] concentrate on finding connected dominating sets, such that all routing is done only using nodes from the dominating set. This has the advantage that other ndes

can be added and removed from the network without affecting the routing, but also the disadvantage of possibly lengthening the routing distance considerably.

The scheme presented here is base on identifying codes, which are supersets of dominating sets. Dominating sets exist for every graph, while an identifying code may not exist. In such a case we propose here to extend the "almost" identifying code to a dominating set and use this construction for routing. The proposed scheme has the advantage of combining neturally with the monitoring scheme. Furthermore, it has the advantage of naturally producing the labeling of nodes, using the identifying code structure, rather than assuming it is externally given, as in dominating set routing. The scheme gives the same penalty in terms of the routing length as dominatingset routing. The size of the code is in many cases not considerably larger than the dominting set. For example, in a random graph, the identifying code is only larger than a dominating set by a logarithmic factor. The method suggested here uses all nodes, rather than a connected dominating set, for routing. This guarantees a small penalty in the routing length, but uses non code nodes for routing. It may be extended to a dominating set only routing by extending the identifying code to a connected dominating set using similar methods to the literature above.

C. Contributions

Our first contribution is to propose a new polynomial-time approximation algorithm for the minimum r-robust identifying code problem with provable performance guarantees; to the best of our knowledge, this is the first such approximation in the literature. Our algorithm, called $\texttt{rID}-\texttt{LOCAL}$, generates a robust identifying code whose size is guaranteed to be at most $1+2\log(n)$ times larger than the optimum, where n is the number of vertices (a sharper bound is provided in Section II). This approximation is obtained through a reduction of the original problem to a minimum set k-cover problem, for which greedy approximations are well known, and utilizes only localized information that lends itself to a distributed computation.

As such, we also propose $\texttt{rID}-\texttt{SYNC}$ and $\texttt{rID}-\texttt{ASYNC}$, two distributed implementations of our algorithm. The first implementation provides a tradeoff between runtime and performance guarantees while using a low communications load and only coarse synchronization; the second implementation requires no synchronization at the expense of more communication. Through simulations on Erdos-Renyi random graphs and geometric random graphs, we show that these algorithms significantly outperforms the earlier ID-CODE algorithm.

Finally, we demonstrate that the same identifying code-based monitoring infrastructure can be reused to efficiently implement routing between any two nodes in the network. Specifically, we show how to use routing tables of the same size as the network's identifying code to route packets between any two nodes within two hops of the shortest path. The significance of this result is two-fold: (i) we can perform near-optimal routing while significantly compressing routing table memory in each sensor; (ii) one algorithm can be run to simultaneously setup both monitoring and routing infrastructures, thus reducing the network setup overhead.

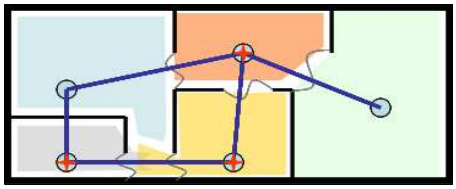


Fig. 1. A floor plan quantized into 5 regions represented by 5 vertices (circles). An edge in the model graph (blue line) represents RF connectivity between a pair of vertices, and the 3 vertices marked by red stars denote an identifying code for the resultant graph.

D. Outline

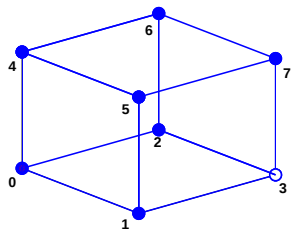
Next, in Section II, we provide a brief introduction to robust identifying codes followed by a centralized approximation algorithm with proven performance bounds. Thereafter, in Section III we provide and analyze a distributed version of this algorithm. Section IV describes a novel technique for reusing identifying codes for routing, and Section V provides some simulation data for the various algorithms considered.

II. Robust identifying codes and the set multicover problem

In this work, we provide a polynomial-time greedy approximation of the NP-complete [11, 12] (robust) identifying code problem. Our approximation is based on an efficient reduction from the set k-multicover problem, for which a greedy approximation is known.

Given a base set U of m elements and a collection S of subsets of U, the set cover problem asks to find a minimum sub-collection of S whose elements have U as their union (*i.e.*, they cover U). The set cover problem is one of the oldest and most studied NP-hard problem [17] and it admits a simple greedy approximation: iteratively choose the heretofore unselected set of S that covers the largest number of uncovered elements in the base set. The classic results of Johnson [18] showed that, for minimum cover s_{min} and greedy cover s_{greedy}, we have that $\frac{s_{\text{greedy}}}{s_{\text{min}}} = \Theta(\ln m)$. Hardness results [19] suggest that this greedy approach is one of the best polynomial approximations to the problem.

The *minimum set k-multicover* problem is a natural generalization of the set cover problem, in which given $(\mathbf{U}, \mathcal{S})$ we are seeking the smallest sub-collection of $\mathcal{S}$ that covers every element in $\mathbf{U}$ at least k times (more formal definitions are in Section II-C). Often this problem is addressed as a special case of the *covering integer problem* [20]. The set k-multicover problem admits a similar greedy heuristic to the set cover problem, with a corresponding performance guarantee [20] of at most $1+\log\left(\max_{S_i\in\mathcal{S}}(|S_i|)\right)$.



$$\begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Fig. 2. A 1-robust identifying code (codewords are solid circles) and the graph's adjacency matrix. The identifying set of vertex 1 is $\{0, 1, 5\}$.

A. Technical definitions

Given an undirected graph $G = (V, E)$, the *ball* $B(v)$ consists of all vertices adjacent to the vertex v, together with v itself. It is possible to generalize this definition (and the corresponding results in the paper) to directed graphs, but this significantly complicates the notation and we omit such results.

A non-empty subset $\mathbb{C} \subseteq V$ is called a *code* and its elements are *codewords*. For a given code $\mathbb{C}$, the *identifying set* $I_{\mathbb{C}}(v)$ of a vertex v is defined to be the codewords directed towards v, *i.e.*, $I_{\mathbb{C}}(v) = B(v) \cap \mathbb{C}$ (if $\mathbb{C}$ is not specified, it is assumed to be the set of all vertices V). A code $\mathbb{C}$ is an *identifying code* if each identifying set of the code is unique, in other words

$$\forall u, v \in V \quad u = v \longleftrightarrow I_{\mathbb{C}}(u) = I_{\mathbb{C}}(v).$$

In our applications, we shall further require that $I_{\mathbb{C}}(v) \neq \emptyset$ for all vertices v, so that an identifying code is also a *vertex cover* or *dominating set*.

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

B. Reduction intuition

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

A graph's adjacency matrix provides an alternate perspective to the identifying code problem; in this matrix, rows or columns are *characteristic vectors* of corresponding balls, in that their i-th entries are 1 if the i-th element of U is in the corresponding ball. Selecting codewords can be 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 [6] that if the Hamming distance of every two rows in the code matrix is at least $2r+1$ then the set of vertices is r-robust.

We next form the $\frac{n(n-1)}{2} \times n$ *difference* matrix by stacking the bitwise XOR results of every two different rows in the adjacency matrix. The problem of finding a minimum size r-robust identifying code is thus equivalent to finding a minimum number of columns in the difference matrix for which the resulting matrix has minimum Hamming distance $2r + 1$ between any two rows. This equivalent problem is nothing but a set $2r+1$-multicover problem, if one regards the columns of the difference matrix as the characteristic vectors of a collection of subsets $\mathcal{S}$ over the base set of all pairs of rows in the adjacency matrix).

In the next subsection we formalize this intuition into a rigorous reduction.

C. Reduction

In this section we formally reduce the problem of finding the smallest sized r-robust identifying code over an arbitrary graph G to a $2r+1$-multicover problem. Formally we connect the following problems:

a) SET MULTI-COVER (SC_k):

`INSTANCE:` Subsets S of U, an integer $k \geq 1$.
`SOLUTION:` $S' \subseteq S$ such that for every element $u \in U$, $|\{s \in S' : u \in s\}| \geq k$.
`MEASURE:` The size of the multicover: $|S'|$.

b) Robust ID-CODE (rID):

`INSTANCE:` Graph $G = (V, E)$, and integer $r \geq 0$.
`SOLUTION:` An r-robust identifying code $C \subseteq V$.
`MEASURE:` The size $|C|$.

Theorem 1 *Given a graph G of n vertices, finding an r-robust identifying code requires no more computations than a $(2r+1)$-multicover solution over a base set of $\frac{n(n-1)}{2}$ elements together with $O(n^3)$ operations of length n binary vectors.*

To prove the theorem we start with few definitions.

Definition 2 *The* difference set *$D_{\mathbb{C}}(u, v)$ is defined to be the symmetric difference between the identifying sets of vertices $u, v \in V$:*

$$D_{\mathbb{C}}(u, v) \quad \doteq \quad I_{\mathbb{C}}(u) \oplus I_{\mathbb{C}}(v),$$

For simplicity of notation, we shall omit the subscript when looking at identifying codes consisting of all graph vertices, i.e., *$D(u, z) = D_V(u, z)$.*

Definition 3 *Let* $\mathrm{U} = \{(u, z) | u \neq z, u, z \in V\}$. *Then the* distinguishing set *δ_c is the set of vertex pairs in* U *for which c is a member of their difference set:*

$$\delta_c = \{(u, z) \in \mathrm{U} \,|\, c \in D_{\mathbb{C}}(u, z)\}.$$

It has been shown in [6] that a code is r-robust if and only if the size of the smallest difference set is at least $2r+1$. Equivalently, a code is r-robust if and only if its distinguishing sets $2r + 1$-multicover all the pairs of vertices in the graph.

Lemma 1 *Given $G = (V, E)$ the following statements are equivalent:*

1) *$\mathbb{C} = \{c_1, ..., c_k\}$ is an r-robust identifying code.*

2) $|D_{\mathbb{C}}(u,v)| \geq 2r+1$, *for all* $u \neq v \in V$
3) *The collection* $\{\delta_{c_1}, ..., \delta_{c_k}\}$ *forms a* $(2r+1)$*-multicover of* $\mathbf{U} = \{(u,v) \,|\, \forall\ u \neq v \in V\}$.

Proof of Theorem 1: Consider the following construction of an r-robust identifying code.

$\mathrm{ID}(G,r) \rightarrow \mathbb{C}$
1. Compute $\{I(u) | u \in V\}$.
2. Compute $\Delta = \{\delta_u | u \in V\}$.
3. $\mathbf{C} \leftarrow \mathrm{Minimum-Set-MultiCover}(2r+1, U, \Delta)$
4. Output $\mathbb{C} \leftarrow \{u \in V | \delta_u \in \mathbf{C}\}$

The resulting code, $\mathbb{C}$, is guaranteed by Lemma 1 to be an r-robust identifying code, and the optimality of the set cover in step 3 guarantees that no smaller identifying code can be found. To complete the proof we observe that computing the identifying sets $I(u)$ naively requires $\theta(n^2)$ additions of binary vectors, and computing Δ requires n operations for each of the $\frac{n(n-1)}{2}$ elements in $|U|$. ∎

D. Localized robust identifying code and its approximation

It was observed in [6, 21] that an r-robust identifying code can be built in a localized manner, where each vertex only considers its two-hop neighborhood. The resulting *localized identifying codes* are the subject of this section, and the approximation algorithm we derive is critical to the distributed algorithm of the next section.

Let $G = (V, E)$ be an undirected graph, we define the distance metric $\rho(u, v)$ to be the number of edges along the shortest path from vertex u to v. The ball of radius l around v is denoted $B(v; l)$ and defined to be $\{w \in V | \rho(w, v) \leq l\}$. So far we encountered balls of radius $l = 1$, which we simply denoted by $B(v)$.

Recall that a vertex cover (or dominating set) is a set of vertices, such that the union of their balls of radius 1 covers V. We further extend this notion and define a *r-dominating* set as the set of vertices which r-multicovers V.

Lemma 2 *Given a graph* $G = (V, E)$*, an* $r+1$*-dominating set* $\mathbb{C}$ *is also an* r*-robust identifying code if and only* $|D(u, v)| \geq 2r + 1$ *for all* $u, v \in V$ *such that* $\rho(u, v) \leq 2$.

Proof: The forward implication is an application of Lemma 1. For the reverse implication we take $\mathbb{C}$ to be an $r + 1$ dominating set and assume that $|D(u, v)| \geq 2r + 1$ for $\rho(u, v) \leq 2$; we will show that this assumption is also valid for $\rho(u, v) > 2$. This is because, for $\rho(u, v) > 2$, we have that $B(v) \cap B(u) = \emptyset$, meaning that $|D(u, v)| = |B(v)| + |B(u)|$. Since $\mathbb{C}$ is an $r + 1$ dominating set, it must be that $|B(y)| \geq r+1$ for all vertices y, giving that $|D(u, v)| > 2r+1$. Applying Lemma 1 we thus see that $\mathbb{C}$ must be r-robust. ∎

The localized robust identifying code approximation Lemma 2 can serve as the basis for a reduction from an identifying code problem to a set cover problem, similar to the one in Theorem 1. The main difference is that we will restrict basis elements to vertex pairs that are at most two hops apart, and we then need to guarantee that the resulting code is still r-robust.

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

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

The resulting *localized identifying code approximation* is thus given by Algorithm 1 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).

Algorithm 1 Localized r-robust code $\mathtt{rID-LOCAL}(r, G)$

We start with a graph $G = (V, E)$ and a non-negative integer r. The greedy set multicover approximation is denoted SET-MULTICOVER$(k, \mathbf{U}, \mathcal{S})$.
1) Compute $\{D(u, v) | u \in V,\ v \in B(u; 2)\}$
2) Compute $\Delta^2 = \{\delta_u^2 | u \in V\}$.
3) $\mathbf{C} \leftarrow$SET-MULTICOVER$(2r + 1, \mathbf{U}^2, \Delta^2)$
4) Output $\mathbb{C}_{local} \leftarrow \{u \in V | \delta_u^2 \in \mathbf{C}\}$

Theorem 2 *Given an undirected graph* $G = (V, E)$ *of* n *vertices, the performance ratio* $\mathtt{rID-LOCAL}$ *is upper bounded by:*

$$\frac{c_{greedy}}{c_{min}} < \ln \gamma + 1,$$

where $\gamma = \max_{v \in V} |B(v)|(|B(v; 3)| - |B(v)| + 1|$.

Proof: The proof derives from the performance guarantee of the greedy set multicover algorithm [20], which is upper bounded by $1 + \ln \alpha$ with α denoting the largest set's size. The size of δ_v^2 is $|B(v)|(|B(v; 3)| - |B(v)| + 1)$, which, at its maximum, can be applied to the performance guarantee in [20] to complete the proof. ∎

In the next subsection we present a distributed implementation of the identifying code localized approximation. The following lemma supplements Lemma 2 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 3 *The distinguishing sets* δ_v^2 *and* δ_u^2 *are disjoint for every pair* (u, v) *with* $\rho(u, v) > 4$.

Proof: Clearly, δ_v^2 includes all vertex pairs $(x, y) \in \mathbf{U}^2$ where x is a neighbor of v and y is not. More precisely, $(x, y) \in \delta_v^2$ if

$$x \in B(v) \text{ and } y \in B(x; 2) - B(v). \tag{1}$$

Moreover, for all such (x, y), $\rho(x, v) \leq 3$ and $\rho(y, v) \leq 3$. On the other hand, for $(x', y') \in \delta_u^2$ with $\rho(u, v) > 4$, either

x' or y' must be a neighbor of u, and hence of distance > 3 from v. Thus, δ_v^2 and δ_u^2 are disjoint. ∎

Lemma 3 implies that when applying the greedy algorithm, a decision on choosing a codeword only affects decisions on vertices within four hops; the algorithm is thus localized to vicinities of balls 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.,* [22]). There are also numerous distributed algorithms for finding a minimum (connected) dominating set based on set cover and other well know approximations such as linear programming relaxation (*e.g.,* [23]). In a recent work Kuhn et. al. [24] 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. The second is a randomized algorithm, which requires a coarse synchronization, guarantees a performance ratio at most $\ln\gamma + 1$ for some design parameter $K \geq 2$, and operates within $O\left(\frac{\gamma n \frac{K+2+\epsilon}{K-1}}{K}\right)$ subslots (resulting in $O(c_{\text{dist}} \max_{v\in V} |B(v;4|))$ communication 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 [6] 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 multi-hop protocols. 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 δ_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.

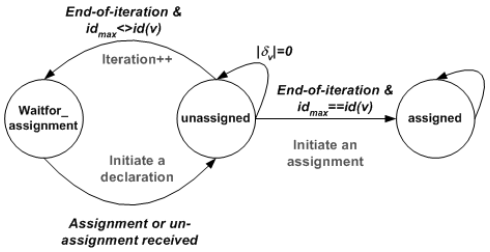


Fig. 3. Asynchronous distributed algorithm state diagram in node $v \in V$

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_{max} and δ_{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_{max} should hold the most distinguishing node in its four hop neighborhood. A node that reaches end-of-iteration event transitions to either the *waitfor_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.

1) Performance evaluation:

Theorem 3 *The algorithm* rID − ASYNC *requires* $\Theta(c_{dist})$

iterations and has a performance ratio

$$\frac{c_{dist}}{c_{min}} < \ln \gamma + 1,$$

where $\gamma = \max_{v \in V} |B(v)|(|B(v;3)| - |B(v)| + 1|$.

The first part of the Theorem follows from Theorem 2 and the fact that only the most distinguishing sets 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 $\Theta(c_{\text{dist}}|V| \max(|B(v;4)|))$, 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 $\texttt{rID} - \texttt{SYNC}$

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.

Algorithm 2 Asynchronous r-robust algorithm ($\texttt{rID} - \texttt{ASYNC}$)

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
- Set $\mathcal{U} = \{\emptyset\}$ and compute $|\delta_v^2(\mathcal{U})| = \delta_v^2 \cap \mathcal{U}$ using (1).
- Initiate a declaration message and set state= "unassigned".
- Set $ID_{max} = ID(v)$, $\delta_{max} = |\delta_v^2(\mathcal{U})|$, and ms to be an empty assignment message.

Iteration
- Increment $hop(ms)$ and forward all messages of $hop(ms) < 4$.
- if received an assignment message ms with $state \neq assigned$ then
  - Update $\mathcal{U}$ and $|\delta_v^2(\mathcal{U})|$.
  - Initiate a declaration message and set state = "unassigned".
  - Reinitialize $ID_{max} = ID(v)$ and $\delta_{max} = |\delta_v^2(\mathcal{U})|$.
- if $state = waitfor\_assignment$ and received an *un-assignment* message then initiate a declaration message.
- if received a declaration message ms with $state \neq assigned$ then
  - if $\delta_{max} < data(ms)$ or $\delta_{max} = data(ms)$ && $ID_{max} > ID(ms)$ then $\delta_{max} = data(ms)$, $ID_{max} = ID(ms)$
- if *end-of-iteration* reached then,
  - if $ID_{max} = ID(v)$ and $|\delta_v^2(\mathcal{U})| > 0$ then $state = assigned$, initiate an assignment message.
  - otherwise $state = waitfor\_assignment$.

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 size of its distinguishing set; the assignment subslot is determined randomly and uniformly at the beginning of every slot.

Algorithm 3 Synchronous r-robust algorithm ($\texttt{rID} - \texttt{SYNC}$)

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

Precomp
- Set: $slot = s_F$, $subslot = L$, $state = unassigned$.
- Calculate the assignment slot.

Iterate: while $state = unassigned$ and $slot \geq s_1$ do,
- $l = random\{1, ..., L\}$
- if received assignment message, ms then,
  - if $hop(ms) < 4$ forward ms with $hop(ms) + +$.
  - Recalculate the assignment slot.
- elseif $subslot = l$ and $slot = as$ then,
  - $state = assigned$
  - Transmit ms with $id(ms) = id(v)$,and $hop(ms) = 1$

2) Performance evaluation: Algorithm $\texttt{rID} - \texttt{SYNC}$ 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, $\texttt{rID} - \texttt{SYNC}$ 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 L, providing a tradeoff between performance ratio guarantees and the runtime of the algorithm as suggested in the following Theorem.

Theorem 4 *For asymptotically large graphs, Algorithm* $\texttt{rID} - \texttt{SYNC}$ *guarantees with high probability a performance*

ratio of

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

where $\gamma = \max_{v\in V} |B(v)|(|B(v;3)| - |B(v)| + 1|$. *The algorithm also requires* $O\left(\frac{\gamma n^{\frac{K+2+\epsilon}{K-1}}}{K}\right)$ *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 2, 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, $P(K) \geq (1-\bar{p}(K))^{c\text{dist}}$, where $\bar{p}(K)$ 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

$$\begin{aligned}\bar{p}(K) &= \textstyle\sum_{k=K}^{t} L\binom{t}{k}L^{-k}\left(1-\frac{1}{L}\right)^{t-k} \leq \sum_{k=K}^{t}\binom{t}{k}L^{1-k} \\ &\leq \textstyle\sum_{k=K}^{t} L\left(\frac{te}{Lk}\right)^{k} \leq tL\left(\frac{te}{LK}\right)^{K},\end{aligned}$$

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

$$P(K) \geq \left(1 - tL\left(\frac{te}{LK}\right)^{K}\right)^{c\text{dist}} \geq \left(1 - \frac{e}{K}\frac{1}{n^{1+\epsilon}}\right)^{n} \to 1.$$

∎

IV. Routing with identifying codes

The existence of an identifying code for a networks permits a natural routing scheme using small routing tables (typically referred to as "compact routing schemes" [25–27]).

By routing scheme we mean a combination of a labeling scheme L, where L_i is the label (address) of the node i, routing tables T_i located at every node i, and a routing function $f(L_s, L_t, L_i, T_i)$ using the labels of the source node s, the destination node t, and the current node i and the information in the local routing table T_i to choose the next port through which a packet should be sent. For the scheme to be considered *compact*, the table size should be small (*i.e.,* $\forall i|T_i| \ll O(N)$), and the label size should also be small (usually polylogarithmic in N). Furthermore, the description of f should be of constant size (*e.g.,* we do not want to include the whole graph structure in f) and its time complexity should be low (usually polynomial in label sizes and logarithmic or constant in the table size).

Compact routing has been studied for some time in the computer science literature, with the typical focus being designing routing schemes that give good performance in the worst case scenario for all graphs, or for some class of graphs; a good survey of existing approaches is provided by [28]. The closest related routing scheme in the literature is based on a dominating set, and this is not surprising because identifying codes are a special type of dominating set. Though identifying codes for a graph tend to be slightly larger than the more general dominating sets (*e.g.,* for large random graphs, they both have roughly $O(\log N)$ nodes), they have the significant advantage of providing, for free, a natural labeling on the graph. They can also be utilized for distributed data storage (see future work below).

A. Routing with an identifying code

For a given identifying code for a graph G, our scheme induces a compact routing scheme in the following manner: Number the nodes in $\mathbb{C}$ (codewords) as $c_0, \ldots, c_{|\mathbb{C}|-1}$; the label of node i will be the characteristic vector of its identifying set (and is thus unique). At every node, the routing table will include one entry for each of the codewords, which will include the port leading to the shortest path to this codeword.

The routing function f at some node i will be as follows:

1) If t is i's neighbor, send directly to t.
2) Otherwise, choose a codeword c_j, such that $c_j \in B(i)$, *i.e.,* such that the j-th bit of L_i is one. Route by the port assigned to c_j in the routing table T_i.

We note that the routing scheme presented here may be extended to a hierarchical routing scheme using higher radius identifying codes to further reduce the size of the routing table [29].

For a graph allowing an identifying code we can see that the routing table size is at most $|\mathbb{C}|^2$ bits, the label size is $|\mathbb{C}|$, and the routing function performs linearly in the label size. If $|\mathbb{C}|$ is large but the size of $I_{\mathbb{C}}(u)$ is small for all $u \in V$, a more compact label may be used by choosing a different representation of the list, linked list of codewords or a run length encoding of the label.

Theorem 5 *The function f is a valid routing function.*

Proof: At every node the routing table includes an entry for each of the codewords. The entry contains the next port in the shortest path routing to the codeword. Therefore, shortest path routing to the selected codeword is guaranteed. Since the selected codeword is a neighbor of the destination, the packet will be directly routed once the codeword is reached.

∎

Interestingly, the routing distance $r(s,t)$ between nodes s and t is almost identical to the shortest path distance $d(s,t)$.

Theorem 6 *The routing scheme above guarantees that $r(s,t) \leq d(s,t) + 2$.*

Proof: If t is a codeword then routing to t is done using the shortest path by the routing tables.

Suppose t is not a codeword. Assume c is in the identifying set of t. The routing scheme routes to c by shortest path, and then to t by one more hop. Therefore $r(s,t) \leq d(s,c)+1$. By

the triangle inequality $d(s,c) \leq d(s,t) + d(t,c) = d(s,t) + 1$. The theorem follows. ∎

The possibility of routing using the codewords is based on the code being also a dominating set. The creation of an identifying set for identification purposes allows also the use of this set in a natural way to achieve compact routing. The usage of the identifying code rather than a possibly smaller dominating set has the advantage of labeling the nodes in a natural way without demanding an agreement on arbitrary identification or numbering of all nodes (but rather just the codewords) to begin with. It also allows the distributed construction of the labeling scheme and routing tables based on the distributed algorithm presented above.

V. Simulations

We have simulated the centralized, localized and synchronized distributed identifying code algorithms, and applied them to 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.,* simple identifying code) the simulation results are compared to the algorithm suggested by Ray et. al. in [6]. In addition, we show a combinatorial lower bound derived first by Karpovsky et. al. in [9], and the asymptotic (in n - the size of the graph) result of Moncel et. al. [10], who showed that an arbitrary collection of a threshold number of codewords is an identifying code with high probability and that this number is asymptotically tight.

Fig. 4(a) shows the theoretic 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 Fig. 5. 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, which size satisfies the threshold number of [10].

Fig. 4(b) shows the simulation results for the localized and distributed algorithms compared to the centralized one. Recall that the performance of the asynchronous algorithm, $\texttt{rID} - \texttt{ASYNC}$, is identical to the localized approximation. It can be observed that the results of the localized algorithm nearly match the results of the centralized algorithms. 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, $\texttt{rID} - \texttt{SYNC}$. 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 4(c).

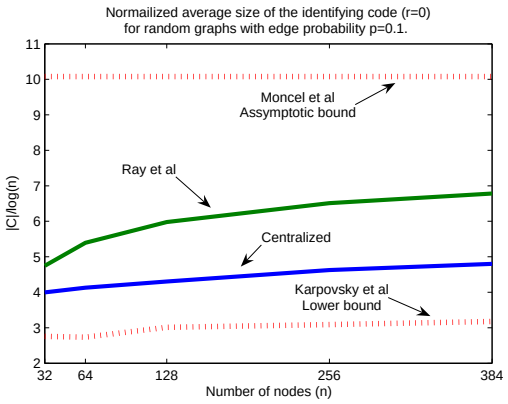


Fig. 5. Average size of the simple identifying code ($r = 0$) for random graphs with edge probability $p = 0.1$, and different number of vertices.

Fig. 6 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 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. However, the monitoring and routing functionality can still be restored by a special treatment of a small fraction of indistinguishable nodes. It should be noted that approaches such that of ID-CODE [6] are not designed to cope with graphs which do not have identifying codes, resulting in a code of all vertices.

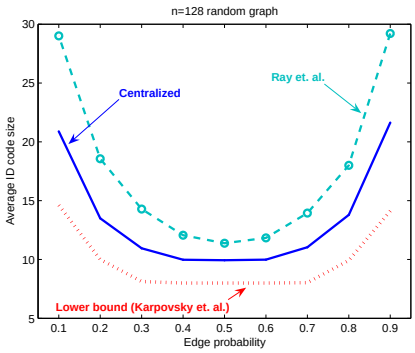
Fig. 7 presents the ratio of the size of the full routing table containing all nodes to the random table containing only codewords for geometric random graphs. In cases where no identifying code exists, information on indistinguishable nodes was added to the routing table. As can be seen the table size is significantly lower for routing using codewords. It should be noted that for non-geometric random graphs the improvement will be significantly higher since small (logarithmic) identifying codes exist. For geometric random graphs further improvement can be obtained using a hierarchical approach as discussed above.

VI. Future work

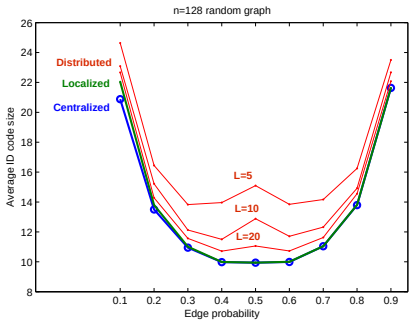
The connections to traditional identifying codes has been sufficiently size-preserving to produce an approximation algorithm, and should be useful for the number of identifying code applications that have been developed in the literature.

Several extensions of identifying codes may prove to hold some promise as future research directions:

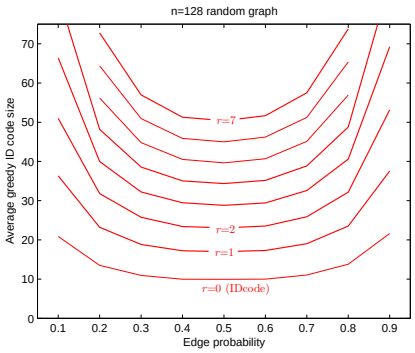
Distributed storage and identifying codes Another extension of routing based upon identifying codes is routing object location in distributed data storage systems. In these systems, a physical or virtual network of storage location is used, and every object can be found in one or more of the



(a) Centralized greedy algorithm



(b) Localized (and `rID − ASYNC`) and distributed algorithm, `rID − SYNC`, for different subslot (L) values



(c) Centralized r-robust identifying codes algorithm.

Fig. 4. Average size of the minimum identifying code for random graphs with edge probability p, and $n = 128$ vertices.

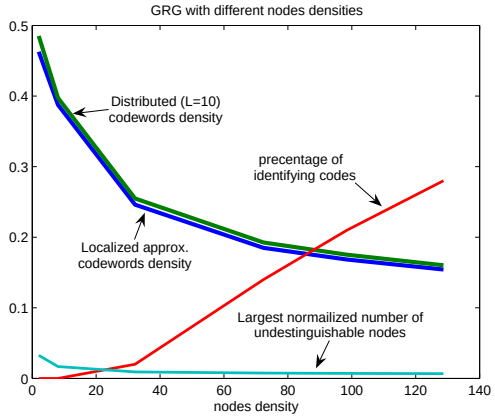


Fig. 6. Normalized size of the code for the localized (and `rID − ASYNC`) and distributed `rID − SYNC`algorithms, fraction of graphs admitting an identifying code, and maximum fraction of indistinguishable nodes for geometric random graphs with different node densities.

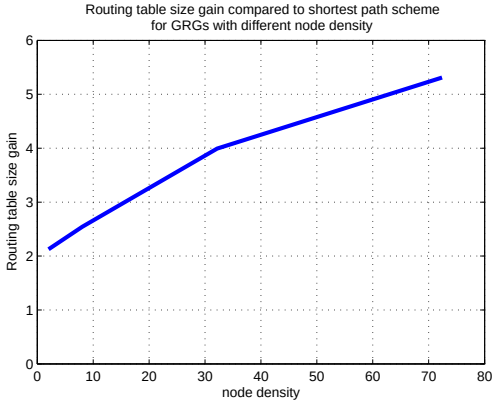


Fig. 7. Ratio of the graph size to the size of the routing table using the identifying code approach. This represents the ratio of the full routing table containing all nodes to the more compact routing table presented here. Results are for geometric random graphs with different node densities.

locations in the network.In this case there is a routing scheme based upon $(1, \leq l)$-identifying codes [30]. These identifying codes produce a different identifying set (and therefore a different label) for every different set of up to l nodes. To locate an object one can use its label (which may also be derived from the object's description by an appropriate hash function), and then use the above routing method to reach one of the codewords neighboring one of the object's storage locations similarly to the scheme presented above.

Robust routing using robust identifying codes In this paper we have discussed the construction of robust identifying codes. Robust identifying codes provide some minimum number of monitors, r, adjacent to each node of the network. This property may be useful in devising methods for compact routing which are resilient to a constant number of failures in the network. It may be possible to construct appropriate routing tables that will allow such resilient routing.

REFERENCES

[1] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet," in *ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, 2002, pp. 96–107.

[2] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong, "A macroscope in the redwoods," in *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, 2005, pp. 51–63.

[3] N. Xu, S. Rangwala, K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004.

[4] "Pump monitoring at a nuclear generating station," 2005, sensicast Systems, Inc. [Online]. Available: http://www.sensicast.com/solutions/casestudys.php

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

[6] S. Ray, D. Starobinski, A. Trachtenberg, and R. Ungrangsi, "Robust location detection with sensor networks," *IEEE Journal on Selected Areas in Communications (Special Issue on Fundamental Performance Limits of Wireless Sensor Networks)*, vol. 22, no. 6, August 2004.

[7] T. Berger-Wolf, W. Hart, and J. Saia, "Discrete sensor placement problems in distribution networks," *SIAM Conference on Mathematics for Industry*, October 2003.

[8] ——, "Discrete sensor placement problems in distribution networks," *Journal of Mathematical and Computer Modelling*, vol. 42, no. 13, 2005.
[9] M. G. Karpovsky, K. Chakrabarty, and L. B. Levitin, "A new class of codes for identification of vertices in graphs," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 599–611, March 1998.
[10] J. Moncel, A. Frieze, R. Martin, M. Ruszink, and C. Smyth, "Identifying codes in random networks," *IEEE International Symposium in Information Theory, Adelaide, 4-9 Sept.*, 2005.
[11] I. Charon, O. Hudry, and A. Lobstein, "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.
[12] ——, "Identifying and locating-dominating codes: Np-completeness results for directed graphs," pp. 2192–2200, August 2002.
[13] J. Moncel, "Optimal graphs for identification of vertices in networks," preprint at www-leibniz.imag.fr/NEWLEIBNIZ/LesCahiers/2005/Cahier138/CLLeib138.pdf.
[14] A. T. Jeremy Blum, Min Ding and X. Cheng, in *Handbook of Combinatorial Optimization*, D.-Z. Du and P. Pardalos, Eds. Kluwer Academic Publishers, 2004, ch. Connected Dominating Set in Sensor Networks and MANETs.
[15] J. Wu and H. Li, "A dominating-set-based routing scheme in ad hoc wireless networks," *'Telecommunication Systems*.
[16] J. Wu, in *Handbook of Wireless Networks and Mobile Computing*, I. Stojmenovic, Ed. Wiley, 2002, ch. Dominating-Set-Based Routing in Ad Hoc Wireless Networks.
[17] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2001.
[18] D. S. Johnson, "Approximation algorithms for combinatorial problems," *Joutnal of Computer and System Sciences*, vol. 9, pp. 256–278, 1974.
[19] U. Feige, "A threshold of $\ln n$ for approximating set cover," *In. Proc. ACM Symposium on Theory of NP-Completeness, New York.*, 1996.
[20] V.Vazirani, *Approximation Algorithms*. Springer-Verlag, July 2001.
[21] M. Laifenfeld and A. Trachtenberg, "Disjoint identifying codes for arbitrary graphs," *IEEE International Symposium on Information Theory, Adelaide Australia*, 4-9 Sept 2005.
[22] S. Rajagopalan and V. Vazirani, "Primal-dual rnc approximation algorithms for set cover and covering integer programs," *SIAM Journal on Computing*, vol. 28, pp. 525–540, 1998.
[23] Y. Bartal, J. W. Byers, and D. Raz, "Global optimization using local information with applications to flow control," in *IEEE Symposium on Foundations of Computer Science*, 1997, pp. 303–312. [Online]. Available: citeseer.ist.psu.edu/bartal97global.html
[24] F. Kuhn and R. Wattenhofer, "Constant-time distributed dominating set approximation," 2003. [Online]. Available: citeseer.ist.psu.edu/kuhn03constanttime.html
[25] D. Peleg and E. Upfal, "A tradeoff between space and efficiency for routing tables," *J. ACM*, vol. 36, pp. 510–530, 1989.
[26] M. Thorup and U. Zwick, "Compact routing schemes," in *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*. ACM Press, 2001, pp. 1–10.
[27] L. Cowen, "Compact routing with minimum stretch," *Journal of Algorithms*, vol. 38, no. 1, pp. 170–183, 2001.
[28] C. Gavoille, "A survey on interval routing schemes," *Theoret. Comput. Sci.*, vol. 249, pp. 217–253, 1999.
[29] I. Abraham and D. Malkhi, "Compact routing on euclidean metrics," in *23rd ACM Symposium on Principles of Distributed Computing (PODC 2004)*. ACM Press, 2004.
[30] S. Gravier and J. Moncel, "Construction of codes identifying sets of vertices," *The Electronic Journal of Combinatorics*, vol. 12, no. 1, 2005.