Distributed Localization of Coverage Holes using Topological Persistence

Harish Chintakunta, Hamid Krim

Abstract

We develop distributed algorithms to detect and localize coverage holes in sensor networks. We do not assume any localization information of the nodes, nor any distances between the nodes. We use algebraic topological methods to define a coverage hole, and develop provably correct algorithm to detect a hole. We then divide the network into smaller subnetworks, while ensuring that the holes are preserved, and checking for holes in each. We show that repeating this process leads to localizing the coverage holes. We demonstrate the improved complexity of our algorithm using simulations.

I. INTRODUCTION

The infrastructure of computing systems is rapidly transitioning from centralized systems to distributed and pervasive systems. A very important class of such systems are sensor networks which find applications in areas including environmental monitoring, health care and military operations [1]. A unifying theme of many of these problems is to glean consensus information by systematically combining the data collected at individual nodes, in accordance to the structure of the network.

From an engineering perspective, nodes in a sensor network have limited communication capability, power and memory. Furthermore, sensor networks are often deployed in unaccessible locations and environments where maintenance is impractical; this makes careful use of exhaustible resources, such as power, imperative.

Each node in a sensor network carries a set of sensors, each with its own sensing region. For each type of sensor, we would ideally want the union of these sensing regions to completely cover the region...
intended to be monitored. When this condition is not met, the uncovered region(s) in this union is called the coverage hole(s).

In this paper, we present a fast distributed algorithm to detect and localize a hole. Our initial efforts for tackling the problem presented here were appeared in [5]. Similar to assumptions made in related work, we assume the sensor coverage regions are correlated to communication regions. We make this correlation precise in our problem formalization. We then divide the network into two subnetworks, ensuring each coverage hole in the original network is present in at least one of these subnetworks, and check for holes in each. By repeating this process, we converge onto the location of the holes.

A. Related work

The coverage problem has been studied extensively, with solutions including geometrical methods [11], [12], statistical methods [18], [19] and many heuristics [13], [22]. Our exposition here, of the various methods proposed for the coverage problem is in no way comprehensive. So far, solutions based on algebraic topology (specifically, homology spaces), seem to be in the most general setting, with the least set of assumptions. In what follows in this section, we outline the work done on the coverage problem using algebraic topological methods, and describe the contributions of this paper.

The application of homology spaces to the coverage problem was first introduced in [8]. Given a set of “fence” edges which are considered to represent the boundary, the work in [8] provides a necessary and sufficient criterion to verify the coverage inside the boundary, and recent work in [9] provides a distributed algorithm to perform this verification. When coverage cannot be guaranteed, [15] gives a criterion using persistent homology to guarantee the existence of holes.

In the context of the coverage problem, we need a distributive algorithm to localize the holes. The work in [23] describes a methodology to compute localized generators given a cover. Given a subdivision of the entire space into subsets, they find an appropriate cover, and check for existing homology classes. The similarity of the work in [23] and the methodology presented here, is that in both cases, given a cover, a basis for homology is computed such that each element of the basis lies entirely a single element in the cover. While [23] is in a generalized setting for arbitrary simplicial complexes, it assumes an appropriate
cover is available, and the computations are in a centralized setting. This paper provides a distributed algorithm to find an appropriate cover for the network, and precisely defined localized generators. Further, the goals, methodologies adopted, and the end results in both these works are very different.

To the best of our knowledge, [20] is the first attempt at distributively localizing holes, by formulating the localization as an optimization problem. Given a non-contractible cycle $c$, [20] looks for a cycle which minimizes the $l_1$ norm in the set of homologous cycles $[c]$. They show, that under certain conditions, minimizing the $l_1$ norm produces the same result as minimizing the $l_0$ norm, where the latter case localizes the holes. The conditions assumed loosely translate to requiring the coverage holes be geographically separated from each other. Therefore, the work in [20] cannot resolve coverage holes which are not well separated. For general simplicial complexes, the problem of minimizing the $l_1$-norm of a cycle up to a constant factor is NP-hard [4]. Hence, such optimization algorithms necessarily rely on iterative algorithms whose convergence rate is usually slow, and difficult to analyze.

The main idea in the existing literature is to construct a data structure called a flag complex $K_G = F(G)$ from the communication graph $G$, and use the topology of this flag complex to infer topological properties of the coverage area. When the communication graph can be modeled as a unit disk graph, $K_G$ is sufficient to infer coverage properties [3]. In practice however, it is unreasonable to assume perfectly symmetric communication regions for nodes, and as a result, the communication graphs are more appropriately modeled as quasi unit disk graphs. Under these circumstances, the topological features of $K_G$ can be arbitrarily different from that of the coverage area [3]. The idea of topological persistence may be used in these situations to infer accurate (but not precise) topological information [3], [15]. Topological persistence involves inclusion maps between homologies of two simplicial complexes, distributed computation of which is non-trivial. The extant work on distributed verification of coverage and localization of holes [20] works with the unit disk model for the communication graphs. We provide a distributed methodology to confirm the topological persistence, and work with the more general setting of quasi unit disk graphs.

---

1We define the notions of contractible and homologous cycles in Section III. Loosely, contractible cycles are ones which do not surround any hole, and two cycles are homologous if they surround the same hole(s)
Contributions

Our contributions for the coverage problem are:

1) We develop distributed algorithms for the general setting of quasi unit disk graphs.
2) The number of communications required to detect and localize the holes is greatly reduced.
3) Individual holes are identified irrespective to their position.

II. Problem Formalization

Consider $N$ sensor nodes randomly deployed in a region of interest in a plane. We denote the collection of all the nodes as the set $V = \{v_i\}$. Each node $v_i$ can communicate with a set of neighboring nodes $N_i$, in its vicinity. A communication graph $G = (V, E)$ is thus formed as the collection of the set $V$ together with the set of edges $E = \{(v_i, v_j)\}$ where $(v_i, v_j) \in E$, if and only if $v_i, v_j$ can communicate with each other. A graph is said to be a unit disk graph, denoted by $G_1 = (V, E_1)$, when $(v_i, v_j) \in E_1$ if and only if $d(v_i, v_j) \leq 1$. A graph is said to be a $\epsilon$–quasi unit disk graph, denoted by $G_\epsilon = (V, E_\epsilon)$, if 1) $(v_i, v_j) \in E_\epsilon$ whenever $d(v_i, v_j) \leq 1 - \epsilon$, 2) $(v_i, v_j) \in E_\epsilon$ with probability 0.5 whenever $1 - \epsilon < d(v_i, v_j) \leq 1$, and 3) $(v_i, v_j) \notin E_\epsilon$ whenever $d(v_i, v_j) > 1$.

For any clique $q$ in the graph $G_1$, let $c_q$ denote the convex hull of nodes in $q$. The union of the convex hulls of all the cliques in $G_1$ is called the Rips shadow, denoted by $R_s$. Let $R_{c_i}$ denote the coverage area of the sensor on node $v_i$, and the union $R_c = \bigcup_i R_{c_i}$ is the coverage area of the network. We make the following assumption about the relationship between the Rips shadow $R_s$ and the coverage area $R_c$:

$R_s$ is homotopy equivalent to $R_c$.

The formal definition of homotopy equivalent spaces is given in section III. Loosely, it means one space may be continuously deformed to obtain the other. Figure 1 shows an example where the coverage area and the Rips shadow are homotopy equivalent. An important consequence of the above assumption is that holes in the coverage area correspond in a direct, one-to-one manner, to the holes in the Rips shadow. Since the communication region can be controlled to an extent by changing the power in the antennas, this assumption is realistic, and similar assumptions have been adopted in related work.
Fig. 1. An example where the coverage area \( R_c \) and the Rips shadow \( R_s \) are homotopy equivalent. (a) shows coverage regions for individual nodes, the union of which is \( R_c \), and (b) shows \( R_s \), the union of convex hull of cliques in the graph.

The question “is there a coverage hole?” is then equivalently stated in terms of homology space as “is \( H_1(R_s) \neq 0 \)?”. Homology spaces are introduced in Section III. Further, for localizing a detected hole, we want to find a small subset of nodes \( V_c \subseteq V \), for which the induced subgraph contains the shortest cycle surrounding the hole.

In practice, it is difficult to obtain \( G_1 \) due to the asymmetry in communication regions of a node, and we may only obtain \( G^\rho_1 \), for some \( \rho \). The value of \( \rho \) may be varied by adjusting the power in the antennas. We assume that we have a good estimate for \( \epsilon \), and we may obtain the graphs \( G^\epsilon_1 \) and \( G^{1+\epsilon}_1 \), where we have \( G^\epsilon_1 \subseteq G_1 \subseteq G^{1+\epsilon}_1 \).

III. Preliminaries

We briefly introduce simplicial complexes, some concepts of homology theory, Laplacian operators, and justify the use of Laplacian operators in the following sections. We refer the readers to [16], for a great exposition of algebraic topology and homology theory. A brief introduction may also be found in [6].

A. Simplicial Complexes

In this paper, we use data structures called simplicial complexes. Given a set of nodes \( V \), an **abstract simplicial complex** is a family \( K \) of subsets of \( V \), such that if \( X \in K \), then all subsets of \( X \) are also in \( K \). The elements in \( K \) are called **simplices**. Given a graph \( G = (V, E) \), there is a natural simplicial complex \( K_G \), called the flag complex of \( G \), where all the cliques in \( G \) are simplices in \( K_G \). A simplex \((v_0, \ldots, v_k)\) with \( k + 1 \) nodes is called a **\( k \)–simplex**\(^\dagger\). A simplicial complex may be viewed as a

\(^\dagger\)this notation is due to the fact that a \( k \)–simplex has a geometric realization in \( \mathbb{R}^k \).
combinatorial generalization of a graph. A $d$–skeleton of a simplicial complex is a collection of all the $m$–simplices, for all $m \leq d$. Note that the $d$–skeleton of a simplicial complex is also a simplicial complex, and is denoted by $K^d$.

For a geometric graph $G_\rho$ (unit disk graph when $\rho = 1$), the flag complex $K_{G_\rho}$ is called a **Vietoris-Rips complex**, which we call Rips complex in short. When we allow for errors in the proximity detection, such as in quasi unit disk graphs, the resulting flag complex is called a **quasi Rips complex**. In this paper, we deal with quasi Rips complexes.

### B. Homology spaces

For a simplicial complex $K$, abstract vector spaces, called chain spaces $C_0(K), C_1(K)$ and $C_2(K)$, are constructed using 0,1 and 2 simplices respectively, as basis elements. In this paper, we use coefficients in $\mathbb{R}$ for these vector spaces. An $i$–simplex has an arbitrary, but fixed, binary orientation given by the ordering on its vertex set. Given an $i$–simplex, the simplex with opposite orientation is its additive inverse in $C_i$. The linear operators $\partial_i : C_i \to C_{i-1}$, denoted as **boundary operators**, capture the combinatorial structure of the simplicial complex, and are given as $\partial_i(v_{j_0}, \ldots, v_{j_i}) = \sum_{l=0}^i (-1)^l (v_{j_0}, \ldots, \hat{v}_{j_l}, \ldots, v_{j_i})$, where $\hat{v}_{j_l}$ implies that $v_{j_l}$ is removed. As an example, for $e = (v_1, v_2) \in E$, $\partial_1(e) = v_1 - v_2$.

The null space of the first boundary operator $\ker(\partial_1)$ is the space of 1-cycles. We use the word “cycles” to represent both cycles in a graph, or elements in the kernel of $\partial_1$. The image of the second boundary operator $\text{Img}(\partial_2)$ is the space of 1-boundaries. The **first homology space** $H_1(K)$ is defined as the quotient space $\ker(\partial_1)/\text{Img}(\partial_2)$. Any cycle $c \in \ker(\partial_1)$ belongs to some element $[c]$ (equivalence class) in $H_1(K)$. Two cycles $c_1, c_2 \in \ker(\partial_1)$ are said to be **homologous** if they are mapped to the same element in $H_1(K)$, i.e., $[c_1] = [c_2]$. Intuitively, homologous cycles surround the same holes. A cycle $c$ is said to be **contractible** if $[c] = 0$, i.e., it does not surround any hole. From the definition of the homology space, the set of contractible cycles is precisely the set of boundaries $\text{Img}(\partial_2)$. We denote the set of boundaries by $B$.

In addition to abstract simplicial complexes, we can define homology for topological spaces (such as the Rips shadow $R_\alpha$ discussed in Section II), by first constructing a simplicial complex isomorphic to
this space [16]. Given a topological space $X$, we denote its first homology by $H_1(X)$. In what follows, we refer to the first homology simply as homology when the order is clear from the context.

C. Combinatorial Laplacians

The first order combinatorial Laplacian $L_1 : C_1 \to C_1$, is defined as

$$L_1 = \partial_2 \partial_2^T + \partial_1^T \partial_1$$

(1)

We also denote by $L_1$, the matrix representation of the linear operator in the standard basis for $C_1$, and by $\partial_1$ and $\partial_2$, the matrix representation of the boundary operators in the corresponding standard bases. We refer the reader to Section III in [17] for a formula for the elements of $L_1$ and examples of harmonics. We present this formula in our supporting document [7] for the readers’ convenience. The following important relationship between $L_1$ and $H_1$ for a simplicial complex [10] is very useful:

$$\ker(L_1) \cong H_1$$

(2)

The elements in $\ker(L_1)$ are called 1-harmonics or simply harmonics.

D. Homotopy

Consider two topological spaces $X$ and $Y$. Two maps $f_1, f_2 : X \to Y$ are said to be homotopic ($f_1 \approx f_2$) to each other if $\exists$ a continuous map $F : X \times I \to Y$ (where $I = [0, 1]$) such that $F(s, 0) = f_1(s)$ and $F(s, 1) = f_2(s)$. Such a function $F$ is called a Homotopy between $f_1$ and $f_2$. Two spaces $X$ and $Y$ are said to be Homotopy Equivalent if $\exists$ continuous maps $f : X \to Y$ and $g : Y \to X$ such that $f \circ g \approx \text{id}$ and $g \circ f \approx \text{id}$. Such a map $f$ is called a homotopy equivalence. A useful property of homotopy equivalent spaces is that they have isomorphic homology spaces.

E. Application to the coverage problem

The usefulness of homology stems from the fact that its dimension is equal to the number of holes in our space of interest, and non-zero elements in it “surround” a set of holes. An example illustrating this
feature of homology may be found in [6]. From our assumption that the coverage area \( R_c \) is homotopy equivalent to the Rips shadow \( R_s \), we have \( H_1(R_c) \cong H_1(R_s) \). Therefore, in the context of detecting holes, \( R_s \) acts as a proxy for \( R_c \).

The Rips shadow \( R_s \) is the natural projection of \( K_{G_1} \) onto the plane, and we have \( H_1(R_s) \cong H_1(K_{G_1}) \) [3]. An example for \( G_1 \) and the corresponding Rips shadow \( R_s \) is shown in Figure 7(a). This means that non-contractible cycles in \( K_{G_1} \) correspond to non-contractible cycles in \( R_s \). However, we do not have \( G_1 \), but instead, as described in Section II, we may obtain \( G_1^\epsilon \) and \( G_1^{1+\epsilon} \). The homologies of \( K_{G_1} \) and \( K_{G_1^{1+\epsilon}} \) by themselves are not reliable for assessing the homology of \( R_s \) [3], but as suggested in [3], reliable information can be obtained by cycles persisting in both of them. The inclusion maps

\[
K_{G_1} \xrightarrow{i_1} K_{G_1^\epsilon} \xrightarrow{i_2} K_{G_1^{1+\epsilon}}
\]

induce homomorphisms on the corresponding homology spaces

\[
H_1(K_{G_1}) \xrightarrow{i_1^*} H_1(K_{G_1^\epsilon}) \xrightarrow{i_2^*} H_1(K_{G_1^{1+\epsilon}}).
\]

For a non-contractible cycle \( c, [c] \neq 0 \) in \( K_{G_1} \), if \( i_2^*i_1^*([c]) \neq 0 \), this implies \( i_1^*([c]) \neq 0 \). Therefore, if a non-contractible cycle in \( K_{G_1} \) is also non-contractible in \( K_{G_1^{1+\epsilon}} \), then it is guaranteed to correspond to a hole in \( R_s \). It suffices to find all the non-contractible cycles in \( K_{G_1} \) and check if they persist in \( K_{G_1^{1+\epsilon}} \). Note however, that there might be cycles which persist in \( K_{G_1} \), but not in \( K_{G_1^{1+\epsilon}} \). In these cases, these holes are missed.

Figure 2(b) shows a Rips complex \( K_{G_1} \), with the same homology as the Rips shadow \( R_s \). In fact \( R_s \) is contractible. But the quasi Rips complex \( K_{G_1^\epsilon} \) obtained from graph shown in Figure 2(a), has a homology or rank 1, and produces a false alarm. However, the cycle \( (v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4 \rightarrow v_1) \) creating this false alarm does not persist in \( K_{G_1^{1+\epsilon}} \). Therefore, checking for persistence helps eliminate these false alarms. Figure 3 gives an example where checking for persistence might cause an existing cycle to go undetected. The likelihood of such misses decreases with the error \( \epsilon \) in proximity detection.

We use the combinatorial laplacian to check for triviality of homology. As a matrix, \( L_1 \) is a square
matrix of dimension $|E|$, where $|E|$ is the number of edges in $G$. Note that the $ij$th element $[L_1]_{ij}$ is non-zero only when the edges $e_i$ and $e_j$ share a node (see the formula for elements of $L_1$ in our supporting document [7]). Therefore, $L_1$ can be computed distributively in the network, with the $i^{th}$ row stored in one of the nodes of the $i^{th}$ edge. Note that if two edges $e_1, e_2$ share a node, then any two nodes in $e_1$ and $e_2$ are at most two hops away.

IV. COVERAGE HOLE LOCALIZATION

A. Algorithm overview

The initial task is to find non-contractible cycles in $K_{G_1}$. Section IV-C describes the distributed computation of cycles, and Section IV-D describes the categorization of these cycles as contractible/non-contractible using harmonics (the elements in $ker(L_1)$). Distributed computation of harmonics is discussed in Section IV-B. Presence of non-contractible cycles indicates non-triviality of homology. When we detect non-trivial homology, we divide the network (in turn, the complex) into two subnetworks, and check for triviality of homology of the flag complex in each, and repeat this process. This division of the network is described in Section IV-F. A division of a subnetwork stops when 1) it has a trivial homology, in which case, it is discarded, or 2) it has a rank 1 homology and a predetermined percentage of the nodes lie on a
single non-contractible cycle. The stopping criteria is described in Section IV-E. Finally, the persistence of the non-contractible cycles identified in $K_{G_{1}}$, into $K_{G_{1+r}}$, is verified using a harmonic in $K_{G_{1+r}}$. The complexity of the algorithms is analyzed in Section V with simulations in Section VI.

B. Computing harmonics

Consider the following dynamic system: \( \frac{dy(t)}{dt} = -L_1y(t) \). Note that the stable point of the above dynamic system is a harmonic. $L_1$ here is computed for $K_{G_{1}}$. It is shown in [17] that the above dynamic system converges for any initial point $y(0)$. A discrete version of the above system is given by:

\[
y^{k+1} = y^k - \delta L_1 y^k
\]  

Here, we derive the sufficient conditions for the range of $\delta$ to guarantee convergence. We also show that, 1) under these conditions, Iteration 3 has a unique convergence point, 2) that it converges exponentially, and 3) derive the convergence rate. The distributed computation of $L_1$ is described in our supporting document [7].

Let $y^0$ be a random vector of dimension $E$, where the elements are generated independently from a uniform distribution on the interval $[-0.5, 0.5]$. Since $L_1$ is a diagonalizable matrix, vector $y^0$ may be expressed as a linear combination of eigenvectors of $L_1$. Let $0 < \lambda_1 \leq \lambda_2 \leq \ldots \lambda_m$ be positive eigenvalues of $L_1$, and let $y^0 = \sum_i \alpha_i v_i$, where $v_i$ are orthonormal eigenvectors of $L_1$. Let $K$ be a matrix with column space equal to the null space of $L_1$. The projection of $y^0$ onto the null space of $L_1$ is equal to $KK^T y^0$. Using Equation 3

\[
y^1 = KK^T y^0 + \sum_i (1 - \delta \lambda_i) \alpha_i v_i
\]

and in general,

\[
y^k = KK^T y^0 + \sum_i (1 - \delta \lambda_i)^k \alpha_i v_i
\]
The sequence \( \{y^k\} \) converges if and only if \( |1 - \delta \lambda_i| < 1 \) for all \( i \), or equivalently,

\[
0 < \delta < \frac{2}{\lambda_i}, \quad \forall i
\]  

(5)

Given a starting vector \( y^0 \), and a scalar \( \delta \) satisfying inequality (5), the sequence \( \{y^k\} \) converges uniquely to \( KK^T y^0 \) and the convergence rate is dominated by the smallest non-zero eigen value. The above discussion is summarized in the following lemma.

**Theorem 4.1:** Let \( y^0 \) be an initial random vector for Iteration (3). Let \( K \) be a matrix with column space equal to the null space of \( L_1 \). Then the iteration converges to \( y^\infty = KK^T y^0 \) if and only if \( \delta \) satisfies inequality (5).

Further, when inequality (5) is satisfied, the iteration converges exponentially with rate \( 1 - \delta \lambda_1 \).

We may easily estimate the spectral radius of the matrix using a well known inequality, \( \|A\|_2 \leq \sqrt{\|A\|_1 \|A\|_\infty} \), for any finite matrix \( A \). Since \( L_1 \) is symmetric, we have \( \|L_1\|_1 = \|L_1\|_\infty \) and therefore \( \|L_1\|_2 \leq \|L_1\|_1 \). The value \( \delta = \frac{1}{\|L_1\|_1} \) satisfies inequality (5). The inequality \( \|L_1\|_1 < \sqrt{n} \|L_1\|_2 \) ensures that the convergence rate is not affected severely by approximating the spectral radius with \( \|L_1\|_1 \).

\( \|L_1\|_1 = \|L_1\|_\infty \) is the maximum absolute sum of rows in \( L_1 \). As discussed in Section III, the row in \( L_1 \) corresponding to an edge \( e = (v_i, v_j) \) is stored in \( v_i \) or \( v_j \). Each node \( v_i \) computes the maximum absolute sum \( x_i \) of the rows stored locally. If there are no rows stored in \( v_i \), then \( x_i \) is set to 0. The value \( \|L_1\|_\infty = \max_i x_i \) may be obtained using the distributed algorithm given in Algorithm 1.

**Algorithm 1** algorithm for computing max

1: at node \( i \):
2: \( m \leftarrow x_i \) \hfill \( \triangleright \) stores the max value
3: broadcast \( m \) to \( N_i \)
4: repeat
5: receive set \( v \) broadcasted by \( N_i \) \hfill \( \triangleright \) at most one value from each neighbor
6: if \( \max(v) > m \) then
7: \( m \leftarrow \max v \)
8: broadcast \( m \) to \( N_i \)
9: end if
10: until
11: No packets are received for a predetermined number of intervals.

It is easy to see that Algorithm 1 converges to the correct value after number of iterations equal to
the diameter of the network. We may set the terminating criteria for the algorithm by waiting a certain number of intervals, which is equal to an estimate for the upper bound of the diameter.

The $i^{th}$ element of the product $L_1y^k$,

$$y^{k+1}(i) = \delta \sum_{j=1}^{E} [L_1]_{ij} y^k(j) = \delta \sum_{l; [L_1]_{il} \neq 0} [L_1]_{il} y^k(l),$$

can be obtained at node storing row $i$, by gathering the values $\{y^k(l), [L_1]_{il} \neq 0\}$. As discussed in Section III, the nodes storing these values are at most 2 hops away. For each the complexes $K_{G_1}$ and $K_{G_1^\epsilon}$, the harmonic need only be computed once. The $i^{th}$ element of the harmonic is stored in the same node which stores the $i^{th}$ row of $L_1$. The harmonic computed initially is used in all the subsequent sub networks.

C. Computing cycles

We start by constructing a spanning tree $T = (V, E_T)$, $E_T \subseteq E_1^\epsilon$ on $G_1^\epsilon$. Fast distributed algorithms for minimum spanning tree of general weighted graphs are well known [14]. Since our graph is unweighted, an arbitrary spanning tree can be obtained by a simple algorithm, such as one shown in Algorithm 2. At the end of Algorithm 2, each node knows its unique parent node, has a list of its children nodes, and the hop length to the root node. The root node $v_r$ is arbitrary, and may be chosen as the node with the maximum index, which in turn can be distributively identified using Algorithm 1.

Algorithm 2 Algorithm to compute spanning tree

1: At root node $v_r$:
2: \(\text{hop\_count} = 0\) \quad \triangleright \text{stores the hop length to the root node}
3: broadcast $[v_r, \text{hop\_count} + 1]$ to $N_r$

4: At any other node $v_i$
5: \(\text{hop\_count} \leftarrow \infty\)
6: \(\text{parent} \leftarrow \emptyset\)
7: when received $[v_j, h]$
8: if $h < \text{hop\_count}$ then
9: \(\text{hop\_count} \leftarrow h\)
10: broadcast $[v_i, \text{hop\_count} + 1]$ to $N_i$ \quad \triangleright \text{all nodes know the identity of the parent node}
11: \(\text{parent} \leftarrow v_j\)
12: transmit “$v_i$ is a child” to $v_j$ \quad \triangleright \text{all nodes know the identity of the children nodes}
13: end if
A sequence of directed edges \( p = (e_{i_1}, \cdots, e_{i_k}) \) in \( G \) is called a path if the last node of \( e_{i_{k-1}} \) coincides with the first node of \( e_{i_1} \). Let the space of all paths in \( G \) be denoted by \( \mathcal{P}_G \). We define the map \( \pi : \mathcal{P}_G \to C_1(K_G) \), as \( \pi(p) = \sum_{j=1}^k \alpha_j \sigma_j \), where \( \sigma_j \) is the standard basis element with the support set equal to the incident nodes of \( e_{i_j} \), and \( \alpha_j = 1 \) if \( \sigma_j \) and \( e_{i_j} \) have the same orientation or \( \alpha_j = -1 \) otherwise. We define the integral of a harmonic \( y \) on a path \( p \) (or on the 1-chain \( \pi(p) \)) to be the dot product \( \langle y, \pi(p) \rangle \). If the first node of the first edge in \( p \) coincides with the last node of last edge, then \( p \) is a cycle in \( G \). Note that in this case, \( \partial_1(\pi(p)) = 0 \).

For an edge \( e = (v_1, v_2) \in E \setminus E_T \), let \( p_1 \) be a path in \( T \) joining \( v_r \) to \( v_1 \), and \( p_2 \) be a path in \( T \) joining \( v_2 \) to \( v_r \). Then the path \( (p_1, e, p_2) \) is a cycle in \( G_1^r \). Denote by \( \gamma(T, e) \), the corresponding 1-cycle. The set of 1-cycles \( Z \) defined as \( Z = \bigcup_{e \in E \setminus E_T} \gamma(T, e) \) forms a basis for all 1-cycles. The reader may refer to [2] for a simple proof. Loosely, having a basis for 1-cycles implies that we have all the cycles required to represent the non-contractible cycles in \( K_{G_1^r} \).

**D. Identifying contractible cycles**

The following lemma gives a necessary condition for contractible cycles.

**Lemma 4.1:** Let \( y \) be a harmonic and \( c \in B \) a boundary. Then, \( \langle y, c \rangle = 0 \)

**Proof:** see our supporting document [7].

The above lemma implies that all harmonics integrate to zero on (or are orthogonal to) contractible cycles. The question now is when does \( \langle y, c \rangle = 0 \) imply \( c \) is contractible. To answer this, we will first look at the set of harmonics which are orthogonal to non-contractible cycles in \( Z \).

All the possible cycles we consider are obtained from the tree, and therefore, as a vector of coefficients, the elements in \( c \) are from the set \( \{-1, 0, 1\} \). As a result, we need to consider only finitely many vectors. For a given cycle \( c \), the set of harmonics which are orthogonal to \( c \) is the intersection of the hyperplane \( c^\perp \) with \( \ker(L_1) \), the space of harmonics. Therefore, the set of harmonics which can possibly be orthogonal to at-least one of the non-contractible cycles is given by

\[
S = \bigcup_{c \notin B} c^\perp \cap \ker(L_1)
\]
**Lemma 4.2:** Let \( c \) be a non-contractible cycle, and \( b_1 \) be the first Betti number. The dimension of the set \( c^\perp \cap \ker(L_1) \) is strictly less than \( b_1 \), where \( b_1 \) is the dimension of \( \ker(L_1) \).

**Proof:** We prove the above statement by finding a harmonic which is not orthogonal to \( c \).

The set of cycles \( Z \) may be decomposed as \( Z = B \oplus H_1 \). The cycle \( c \notin B \) may be expressed as \( c = \alpha c_b + \beta c_h \) with \( \beta \neq 0 \). Let \( c_h \) also denote the coefficient vector (of length \( b_1 \)) expressed in some basis \( B_{H_1} \) for \( H_1 \). Let \( K \) be a square matrix of size \( b_1 \), where each column represents a basis element of harmonics expressed in terms of elements in \( B_{H_1} \).

For a given cycle \( c \), let \( \hat{y} = KK^Tc_h \) be the projection onto the harmonic space. The dot product of \( \hat{y} \) with \( c \) gives

\[
\hat{y}^Tc = \hat{y}^T (\alpha c_b + \beta c_h)
\]

\[
= \beta
\]

A direct consequence of Lemma 4.2 is that the the set \( c^\perp \cap \ker(L_1) \) has measure zero. The set \( S \) given in (6) is a finite union of measure zero sets, and therefore has measure zero. This means that the statement “\( \langle y, c \rangle = 0 \Rightarrow c \in B \)” is false on a set of measure zero!

As stated in Theorem 4.1, our process of computing harmonics is equivalent to projecting a random vector on to the space of all harmonics. The above discussion leads to the following important theorem.

**Theorem 4.2:** Let \( c \in Z \) be a cycle in \( Z \), and let \( y \) be a harmonic generated using iteration (3). Then \( c \in B \iff \langle y, c \rangle = 0 \), with probability 1.

Theorem 4.2 allows us to easily identify all the non-contractible cycles in \( Z \). Note that for \( c_1, c_2 \in Z \), by definition, \( c_1 \) is homologous to \( c_2 \) if and only if either \( c_1 + c_2 \in B \) or \( c_1 - c_2 \in B \). Further, the integration of a harmonic on cycles is a linear process. These facts, along with Theorem 4.2 results in the following corollary.

**Corollary 4.1:** Let \( c_1, c_2 \in Z \) be cycles in \( Z \). Then \( c_1 \) is homologous to \( c_2 \) if and only if \( |\langle y, c_1 \rangle| = \)
\[ |\langle y, c_2 \rangle|, \text{ with probability 1.} \]

The spanning tree \( T \) can be effectively utilized to compute these integrals efficiently. We do this in two steps:

1) compute an integral function \( f : V \rightarrow \mathbb{R} \) on the nodes such that \( f(v_i) = \langle y, \pi(p_i) \rangle \), where \( p_i \) is the path in \( T \) joining the root node \( v_r \) to \( v_i \).

2) for a cycle \( \gamma(T,e) \in Z, e = (v_j, v_k) \), the integral is computed as

\[ \langle y, \gamma(T,e) \rangle = f(v_j) + \langle y, e \rangle - f(v_k) \]  

(7)

Algorithm 3 describes the distributed computation of the integral function \( f \). The root node initiates a broadcast which travels down the tree while computing the integral for each node. Each node broadcasts precisely once. Step 2 can be performed locally at one of the incident nodes for each edge.

**Algorithm 3** Algorithm for computing the integral function \( f 

1: \text{At root node } v_r:
2: \quad f(v_r) \leftarrow 0
3: \quad \text{broadcast } [v_r,0] \text{ to } N_r
4: \quad *
5: \quad \text{At any other node } v_i:
6: \quad \text{when received } [v_j,x]
7: \quad \text{if } v_j == \text{parent then}
8: \quad \quad \text{if } v_i > v_j \text{ then}
9: \quad \quad \quad \text{temp} \leftarrow y((v_i, v_j)) \quad \quad \triangleright y \text{ is the given harmonic to be integrated}
10: \quad \quad \quad \text{else}
11: \quad \quad \quad \text{temp} \leftarrow -y((v_i, v_j))
12: \quad \quad \end{if}
13: \quad \quad \quad f(v_i) \leftarrow x + \text{temp}
14: \quad \quad \quad \text{broadcast } [v_i,f(v_i)] \text{ to } N_i
15: \quad \quad \end{if}

**E. Stopping criteria**

The goal of dividing the whole network into subnetworks is to capture the non-contractible cycles in \( K_{G_1} \) (which potentially correspond to holes) in small subnetworks, and hence localizing the holes. For the same reason, for each subnetwork \( X \), we would require the homology of \( K_X \) to be of rank 1, so
that it contains only 1 hole. In the ideal case, we would also require that the only nodes a subnetwork contains are those on a shortest path around the hole. This condition may relaxed to requiring a fraction $\phi \approx 1$ of the nodes to be on the shortest cycle, or by limiting the number of nodes not on the shortest cycle to a small number. On the other hand, if $K_X$ has a trivial homology, i.e., it does not contain any hole, we discard $X$ from consideration and stop further dividing it.

$K_X$ has trivial homology if and only if all the cycles are contractible. The homology of $K_X$ is of rank 1 if and only if all the non-contractible cycles are homologous to each other. As described in Sections IV-C and IV-D, the tree $T_X$ constructed on $X$ can be used to find all the cycles, and the integrals of the harmonic computed for $K_{G_1}$ can be used to identify homologous cycles. Specifically, Theorem 4.2 specifies that if the integral $\langle y, c \rangle$ of the harmonic on a cycle $c$ is 0, $c$ is contractible (wit probability 1), and Corollary 4.1 specifies that two cycles $c_1$ and $c_2$ are homologous if and only if $|\langle y, c_1 \rangle| = |\langle y, c_2 \rangle|$ (with probability 1). Algorithm 4 describes the procedure to check for the stopping criteria.

The algorithm assumes that the the integral function $f$, and the tree $T_X$ have been constructed. If any node encounters more than 1 distinct integral values, the result will be to divide the subnetwork. If all the integral values are 0\(^\dagger\), then the subnetwork will be discarded. The integrals in line 4.10 can be computed using Equation (7). In line 4.33 The root node $v_r$ computes hop length of the shortest cycle as $\min\{x_2\}$, and the total number of nodes as $\sum\{x_3\} + \#$ of children + 1. These two values can be used to check if “most” of the nodes lie on the shortest cycle. When the stopping criteria is not met, the division process is initiated which is described in the following section.

F. Dividing the network

When a network (or a subnetwork) does not meet the stopping criterion, discussed in Section IV-E, it is divided into two subnetworks. We make sure that such a division does not destroy any holes, a condition necessary for this strategy to successfully localize the coverage holes. Furthermore, it will be efficient to divide the network in such a way that resulting subnetworks are approximately of the same size. In order to perform such a division, one may find a pair of nodes which are farthest from each other.

\(^\dagger\)In practice, a small threshold may be used to account for numerical roundoff errors.
Algorithm 4 Algorithm to check for stopping criteria in a subnetwork $X$

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>At each node $v_i \neq v_r$: $\triangleright$ $v_r$ here is the root node</td>
</tr>
<tr>
<td>2</td>
<td>if received “divide” from a children node then</td>
</tr>
<tr>
<td>3</td>
<td>send “divide” to parent node</td>
</tr>
<tr>
<td>4</td>
<td>Stop</td>
</tr>
<tr>
<td>5</td>
<td>else</td>
</tr>
<tr>
<td>6</td>
<td>$x = {[x_1, x_2, x_3]}$ $\triangleright$ messages from children</td>
</tr>
<tr>
<td>7</td>
<td>end if</td>
</tr>
<tr>
<td>8</td>
<td>$z_1 \leftarrow {x_1}$</td>
</tr>
<tr>
<td>9</td>
<td>$S \leftarrow {(v_i, v_j), v_j \in \mathcal{N}_i, (v_i, v_j) \notin T_X}$</td>
</tr>
<tr>
<td>10</td>
<td>$z_2 \leftarrow {{y, \gamma(T_X, e)}, e \in S}$</td>
</tr>
<tr>
<td>11</td>
<td>if values in $z_1 \cup z_2$ are not all equal then $\triangleright$ cycles are not homologous</td>
</tr>
<tr>
<td>12</td>
<td>send “divide” to parent node</td>
</tr>
<tr>
<td>13</td>
<td>Stop</td>
</tr>
<tr>
<td>14</td>
<td>else</td>
</tr>
<tr>
<td>15</td>
<td>$a_1 \leftarrow z_2(1)$</td>
</tr>
<tr>
<td>16</td>
<td>$Q \leftarrow {\gamma(T_X, e), e \in S}$</td>
</tr>
<tr>
<td>17</td>
<td>$a_2 \leftarrow \min{\text{hop count of cycles in } Q}$</td>
</tr>
<tr>
<td>18</td>
<td>$a_2 \leftarrow \min(a_2, \min{x_2})$ $\triangleright {x_2}$ are values from neighbors</td>
</tr>
<tr>
<td>19</td>
<td>$a_3 \leftarrow \sum {x_3} + #\text{of children} + 1$</td>
</tr>
<tr>
<td>20</td>
<td>send $[a_1, a_2, a_3]$ to parent node</td>
</tr>
<tr>
<td>21</td>
<td>end if</td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>At root node $v_r$:</td>
</tr>
<tr>
<td>24</td>
<td>if received “divide” from a children node then</td>
</tr>
<tr>
<td>25</td>
<td>Initiate division process</td>
</tr>
<tr>
<td>26</td>
<td>Stop</td>
</tr>
<tr>
<td>27</td>
<td>else</td>
</tr>
<tr>
<td>28</td>
<td>$x = {[x_1, x_2, x_3]}$</td>
</tr>
<tr>
<td>29</td>
<td>if not all values in ${x_1}$ are equal then</td>
</tr>
<tr>
<td>30</td>
<td>Initiate division process</td>
</tr>
<tr>
<td>31</td>
<td>Stop</td>
</tr>
<tr>
<td>32</td>
<td>else</td>
</tr>
<tr>
<td>33</td>
<td>use values ${x_2}$ and ${x_3}$ to check if “most” of the nodes are on the shortest cycle</td>
</tr>
<tr>
<td>34</td>
<td>end if</td>
</tr>
<tr>
<td>35</td>
<td>end if</td>
</tr>
</tbody>
</table>

Other in the network, and divide the network along the nodes which are equi-distant ($\pm 1$ hops) in hop length from this pair. The nodes in such a pair are denoted as diameter nodes, and the node equidistant from them as boundary nodes.

1) Finding diameter nodes: Distributed computation of precise diameter, along with the diameter nodes, is expensive. Since the algorithm presented does not rely on this precision, we describe here, an
Fig. 4. Figure shows the iterative procedure described in (8). In (a), the red square is the node with the highest index and the green square is the farthest node (in hop distance from it). In (b), the green square is the farthest node from the node found in (a). In this example, the procedure converges in 2 iterations to the correct value.

iterative procedure which finds a pair of nodes with hop length between them close to the diameter.

Let $v_0$ be an arbitrary node, and let $d$ denote the hop-distance metric. Consider the following iterative procedure:

$$v_{i+1} = \arg \max_{v \in V} d(v_i, v)$$  \hspace{1cm} (8)

The iteration stops when

$$d(v_{i+1}, v_i) = d(v_i, v_{i-1})$$  \hspace{1cm} (9)

**Theorem 4.3:** For a finite graph $G$, the iterative procedure (8) with the stopping condition (9) stops in finite time.

**Proof:** From (8), the symmetric property of $d$ results in the inequality $d(v_{i+1}, v_i) \geq d(v_i, v_{i+1})$. Therefore, for each $i$, the value $\max_{v \in V} d(v_i, v)$ either increases in value, or remains the same. Since the diameter of $G$ is finite, the iteration stops in finite steps. \hfill \blacksquare

For a big set of shapes for the deployment region, the iterative procedure (8) provides a good approximation to the diameter. Consider a compact metric space $(X, d)$ with finite diameter. For a point $v \in X$, denote the maximum distance from this point to any point in $X$ by $d_m(v)$, i.e., $d_m(v) = \max_{v_i \in X} d(v, v_i)$. A pair $\{v_1, v_2\}$ of points is called an extreme pair if $d(v_1, v_2) = d_m(v_1) = d_m(v_2)$. The space $(X, d)$ is said to be isodiametric if the distances between all the extreme pairs are equal.

When the metric space is isodiametric, the procedure (8) computes the diameter precisely. The most common shapes for deployment regions, such as circular, square or rectangular regions are iso-diametric.
Since the hop-length between nodes in a geometric graph is highly correlated with the Euclidean distance between them, procedure (8) results in a good approximation.

The tree $T$ constructed to compute the cycles (as described in Section IV-C) can be used to perform the above procedure. The root node $v_r$ can be designated as $v_0$. The node with maximum hop distance from $v_0$ will be one of the leaves, and at the end of Algorithm 2, all the leaves (in fact, all the nodes) know the hop length to $v_0$. The leaf nodes transmit their hop-length, along with their ID, back to $v_0$ up the tree. When an intermediate node receives values from multiple children nodes, it transmits the maximum value (along with the corresponding ID) to its parent. When $v_0$ receives the messages from all its children nodes, it computes the maximum of these values, along with the argument $v_1$. The node $v_0$ sends this information to $v_1$ by flooding the network (or by using the tree as a routing structure). With $v_i$ as the root node, we compute another spanning tree, and repeat the above procedure to obtain $v_{i+1}$.

For deployment in a square or circular region, with average degree of the nodes ranging from 10 to 15, our simulations show that this process converges in a few iterations (3 to 4) irrespective of the number of nodes. An example of this procedure is shown in Figure 4.

2) Finding boundary nodes: As the physical positions of the nodes do not change, we form a virtual segmentation by finding boundary nodes $B = \{b_i\}$ within a partition which stop messages from passing through. This effectively separates a given partition into two parts with non-intercommunicating nodes. For a set $B$ to behave like a boundary, it has to satisfy certain properties:

**Definition** Let $X = (V_X, E_X)$ be a connected sub-graph. The set of nodes $V_Y$ is said to be a boundary in $X$, if and only if $\exists$ two disjoint sets $V_{X_1}, V_{X_2} \subset V_X$ such that there is a node $b_i \in V_Y$ in any path $(v_i, \ldots, v_j)$, where $v_i \in V_{X_1}$ and $v_j \in V_{X_2}$. Furthermore, $V_{X_1} \cup V_{X_2} \cup V_Y = V_X$. We say that $V_Y$ divides $V_{X_1}$ and $V_{X_2}$, and denote by $X_1$, $X_2$, and $Y$, the subgraphs induced by $V_Y \cup V_{X_1}$, $V_Y \cup V_{X_2}$, and $V_Y$ respectively.

To avoid confusion in notation, we emphasize here that $X_1$ is a subgraph induced by nodes in $V_Y \cup V_{X_1}$ and not just $V_{X_1}$ (and same is true for $X_2$). Figure 5 shows an example of boundary nodes. If every path

---

$^\dagger$This definition of a boundary should not be confused with the conventional notion, confounded with the closure of a graph or a region. The particular definition we are using will be clear from the context.
Fig. 5. The boundary nodes on the left do not induce a connected subgraph, and therefore, do not preserve topology.

from $V_{X1}$ to $V_{X2}$ contains a boundary node, this means there is no path along which a message from $V_{X1}$ can reach $V_{X2}$, thus virtually separating both. This justifies the above definition for the boundary. The boundary nodes identify their neighbors as belonging to $V_{X1}$ or $V_{X2}$, and do not transmit messages from one to the other.

For a given subgraph $X$, and the diameter nodes $v_1, v_2$, we choose the boundary nodes to be the set of nodes,

$$V_Y = \{v, |d(v, v_1) - d(v, v_2)| \leq 1\}. \quad (10)$$

$V_Y$ divides the sets $V_{X1} = \{v, d(v, v_1) - d(v, v_2) > 1\}$ and $V_{X2} = \{v, d(v, v_1) - d(v, v_2) < -1\}$. An important property that such a boundary should satisfy, is that it should preserve the topology (see Figure 5).

**Theorem 4.4:** For a given subgraph $X$, let $V_Y$ divide the sets $V_{X1}$ and $V_{X2}$. For any cycle $\rho \in X$, let $c = \pi(\rho)$. If the subgraph $Y$ induced by nodes in $V_Y$ is connected, then $c$ can be written as $c = c_1 + c_2$, where $c_1 = \sum_i \pi(\rho_{1i})$ and $c_2 = \sum_i \pi(\rho_{2i})$, such that $\forall i$, $\rho_{1i}$ is a cycle in $X_1$, and $\rho_{2i}$ is a cycle in $X_2$.

**Proof:** see our supporting document [7].

The map $\pi$ in the above theorem was defined in Section IV-C, as a map from the set of edges to $C_1$. The consequence of the above is that, at any stage of the algorithm, a basis for $ker(\partial_1)$ may be obtained by a set of cycles, where each cycle is restricted to a subnetwork.
The boundary nodes computed using (10) are not necessarily connected, as seen in the example in Figure 5. In this case, the following procedure may be used to connect the components. Each connected component in $Y$ elects a representative node, by choosing the node with the maximum index (using Algorithm 1). This index is flooded into the network, so that all the nodes have a list of representatives. Denote the list of indices by $(i_1, \ldots, i_k)$ in an ascending order. Then, all the nodes in the shortest paths joining $v_{i_{l-1}}$ to $v_{i_l}$ are included in $V_Y$. This ensures the subgraph induced by $V_Y$ is connected. The shortest paths may be found by constructing a spanning tree using Algorithm 2.

The following corollary of Theorem 4.4 captures the topology preservation of the division process.

**Corollary 4.2:** Suppose that the division process creates the set of subnetworks $\mathcal{X} = \{X_i\}, i = 1 \ldots m$, of $G'_1$. Then, $\exists$ a basis $[c_j], j = 1, \ldots, b_1$, for $H_1(K_{G'_1})$, such that $\forall [c_j], c_j = \pi(\rho_j)$, where $\rho_j$ is a cycle which lies entirely in one of the subnetworks in $\mathcal{X}$.

**Proof:** Let $[h_j], j = 1, \ldots, b_1$, be an arbitrary basis for $H_1(K_{G'_1})$. Then, using Theorem 4.4, we have, $\forall j, h_j = (\sum_{l=1}^m \sum_k \pi(\rho_{lk}))$, such that for each $k$, $\rho_{lk}$ lies entirely in the subnetwork $X_l$. The stopping criteria dictates all non-contractible cycles in $X_l$ are homologous to each other. If $X_l$ has at least one contractible cycle, then $[\sum_k \pi(\rho_{lk})]$ is an integer multiple of $[\pi(\rho_{l1})]$. 

Corollary 4.2 states that the homology of $K_{G'_1}$ is generated by cycles which are restricted to individual subnetworks. The stopping criteria discussed in Section IV-E states that each subnetwork either does not contain any non-contractible cycles, or almost entirely is made of a single cycle.

**G. Checking for persistence**

A harmonic is computed for both the complexes $K_{G'_1}$ and $K_{G'_1+\epsilon}$. Distributed computation of harmonics is described in Section IV-B. When a subnetwork $X \subseteq K_{G'_1}$ satisfies the stopping criterion (Algorithm 4), with a non-contractible cycle $c$, $c$ also exists in $K_{G'_1+\epsilon}$ since $K_{G'_1} \subseteq K_{G'_1+\epsilon}$. Its persistence in $K_{G'_1+\epsilon}$ may be tested by computing the integral $\langle y', c \rangle$, where $y'$ is a harmonic computed for $K_{G'_1+\epsilon}$. If $\langle y', c \rangle = 0$, $c$ does not persist in $K_{G'_1+\epsilon}$ (and persists otherwise).
V. COMPLEXITY

All the subroutines and procedures described above, except those for computing harmonics and maximum value, require the nodes to transmit a constant number of times. The message complexity for computing maximum (Algorithm 1) is upper bounded by the diameter of the network, and in practice, the diameter is a very loose upper bound. Further, as the size of the subnetworks decreases, so does the complexity of computing maximum. The run time for both the maximum and spanning tree algorithm is upper bounded by the diameter of the network. The bottleneck for both the message and run time complexity is that of computing harmonics. Note the we compute a harmonic only once for each of $K_{G_1}$ and $K_{G_1^+}$.

The subroutines for computing the 1) spanning tree, 2) the stopping criteria, 3) the integral function and 4) maximum, are executed once for every subnetwork formed. The first three subroutines require nodes to transmit a constant number of messages irrespective to the size of the subnetwork. Therefore, the message complexity due to these subroutines is proportional to the number of subnetworks formed. Note that when a subnetwork is found to have no non-contractible cycles, it is discarded. Therefore, the worst case scenario is when all the subnetworks formed during the division process have at least one non-contractible cycle. The maximum number of subnetworks that can be formed is $2N (2^N - 1 = 1 + 2 + 2^2 + \ldots + 2^k, 2^k = N)$, where $N$ is the number of nodes in the network. Hence, the worst case message complexity per node, of the first three subroutines is $O(N)$, and in practice, is much lower.

The diameter for geometric graphs with $N$ nodes is of the order $\sqrt{N}$. If none of the subnetworks are discarded, the number of subnetworks doubles every iteration, and the number of nodes in each subnetwork approximately halves. Since the worst case message complexity for the subroutine to find maximum is equal to the diameter, the total number of messages (upper bound) would be $\sqrt{N} + 2\sqrt{N/2} + 2^2\sqrt{N/4} + \ldots$ which again is of the order $N$. Therefore, the worst case complexity to compute the maximum is also $O(N)$.

As stated in Theorem 4.1, Iteration 3 converges with exponentially with a rate $1 - \lambda_1/\|L_1\|_1$ (here $\delta = 1/\|L_1\|_1$). Let the required precision be $\epsilon$ and the number of iterations required to reach this precision
be $\rho$. As per Equation 4, the error term after the $k^{th}$ iteration is given as

$$\sum_i (1 - \delta \lambda_i)^k \alpha_i v_i \approx (1 - \delta \lambda_1)^k \alpha_1$$

(11)

for precision $\epsilon$, we have

$$\epsilon = (1 - \delta \lambda_1)^\rho |\alpha_1|$$

(12)

When the first Betti number $b_1$ is much smaller than $|E|$, then $E[||\alpha||]$ approaches a constant value (See our supporting document [7]). The number of iterations $\rho$ required for a given precision $\epsilon$ hence varies as $O\left(\frac{\log \epsilon}{\log(1 - \delta \lambda_1)}\right)$.

The size of a packet corresponding to edge $e_i$ is proportional to the number of non-zero elements in $i^{th}$ row of $L_1$, the average of which we denote by $|L_1|_{avg}$. For geometric graphs, the expected total number of non-zero elements in $L_1$ (denoted $|L_1|$) varies as $2k(k - 1/4)N$, where $k$ is the average degree for the nodes, and $N$ is the total number of nodes (See our supporting document [7]). This implies $|L_1|_{avg}$ is a constant value determined by the average degree of the nodes. Therefore, both the run time and message complexity of the methodology presented in this paper is given as $O\left(\frac{\log \epsilon}{\log(1 - \delta \lambda_1)}\right)$, $\delta = 1/||L_1||_1$. Figure 6 shows experimental results on the number of iterations required for varying precision and varying number of nodes. Experiments show that the number of iterations required grows linearly with the number of nodes.

We now compare the above complexity with methodology used in [20]. For localizing the holes, the work in [20] first computes a harmonic in the network, and then uses a sub-gradient method to minimize the $l_1$ norm of cycles homologous to this harmonic. Therefore, the complexity is necessarily higher than that presented here. Further, sub-gradient methods for minimizing $l_1$ norm are very slow to converge. This can also be seen in the number of iterations required to converge, in the experimental results (Figure 8) presented in [20].
VI. SIMULATIONS

An experimental illustration of the methodology presented in this paper is shown in Figure 7. For clarity of the division process, the graph shown is obtained by assuming circular communication regions. Figure 7(a) shows the graph, with the Rips shadow $R_s$ as the shaded region. Figure 7(b) shows the spanning tree $T$ constructed using Algorithm 2, and the integral of the harmonic is shown in the scatter plot 7(c). The edges in blue shown in Figure 7(d) is the set $\{e_i\}$, for which the cycles $\gamma(T, e_i)$ created are non-contractible. In this case, there are several non-contractible cycles which are not homologous to...
each other, and the stopping criteria is not met. The diameter nodes (in red), and the boundary nodes (in blue) computed in the division process are shown in Figure 7(e). The process is repeated, during which subnetworks with no non-contractible cycles are discarded. Figure 7(f) shows the union of the subnetworks which have not been discarded after 6 iterations. The stopping criteria used here required all the nodes in a subnetwork to belong to a single non-contractible cycle, and all the subnetworks meet this criteria after 14 iterations. Some of these subnetworks are shows in Figure 8. Note that all the holes in the Rips shadow have been accurately localized by these subnetworks, irrespective of their geographic separation.

VII. CONCLUSION

A distributed algorithm for localizing coverage holes is presented. The methodology works under a basic assumption, that the coverage area is homotopy equivalent to the Rips shadow of the communication graph (obtained with no noise in proximity detection). When there is noise in the proximity detection, we obtain communication graphs at two different resolutions, and infer the topological properties of the Rips shadow using persisting features.

The methodology derives its power from being able to distributively, and efficiently, verify whether a given cycle is contractible, and whether two cycles are homologous to each other. This is made possible by the use of harmonics, which in turn can be efficiently computed. The work presented here provides an improvement in the complexity of localizing holes, and resolves individual holes irrespective to their geographic separation.

REFERENCES

Fig. 8. Subnetworks formed after 14 iterations


