Hole Detection or: "How much Geometry hides in Connectivity?"

Stefan Funke*
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken, Germany
funke@mpi-inf.mpg.de

Christian Klein
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken, Germany
cklein@mpi-inf.mpg.de

ABSTRACT

Wireless sensor networks typically consist of small, very simple network nodes without any positioning device like GPS. After an initialization phase, the nodes know with whom they can talk directly, but have no idea about their relative geographic locations. We examine how much geometry information is nevertheless hidden in the communication graph of the network: Assuming that the connectivity is determined by the well-known unit-disk graph model, we prove that using an extremely simple linear-time algorithm one can identify nodes on the boundaries of holes of the network. That is, there is enough geometry information hidden in the connectivity structure to identify topological features – in our example the holes in the network. While the theoretical analysis turns out to be quite conservative, an actual implementation shows that the algorithm works well under less stringent conditions.

Categories and Subject Descriptors: C.2.2 [Computer-Communication Networks]: Network Protocols

General Terms: Algorithms, Design

Keywords: Routing, Graph Theory, Embedding, Virtual Coordinates, Topology

1 Introduction

Imagine the following scenario: during a long summer drought, forest fires have started in a large region of a remote nature preserve that is hardly accessible by ground transportation. To be able to continuously assess the situation and plan appropriate countermeasures, airplanes are sent out to deploy thousands of wireless sensor nodes. Due to cost restrictions and to achieve the maximum life-time by energy savings, these sensor nodes are rather low-capability devices equipped just with temperature and humidity sensors, a simple processing unit, and a small radio device that allows for communication between nearby sensor nodes. One of the first goals is now to have this network organize itself such that messages are routed within the network, regions of interest (e.g. the current fire-front) can be identified, and gathered data can be efficiently queried.

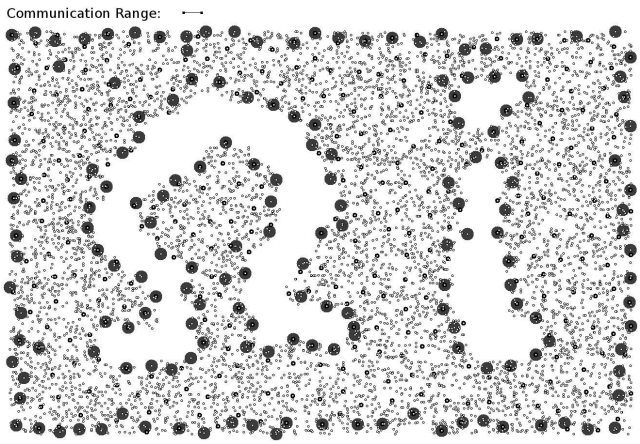

Figure 1. Based just on the *connectivity* of the network our algorithm detects nodes close to boundaries (black circles). At the top, the communication range of the network nodes is depicted.

Achieving this goal becomes quite challenging since the only information a node has about the global network topology are its immediate neighbors with whom it can communicate. Lacking an energy-hungry GPS unit and being deployed from an airplane in a rather uncontrolled fashion, none of the sensor nodes is aware of its geographic location.

Assume the area of interest is some region $\mathcal{R}$. The airplanes have deployed sufficiently many sensors such that – if all of the sensors were in operation after reaching the ground – the area of interest is completely monitored by the sensors. Formally we have that for every point $p \in \mathcal{R}$ there would be at least one sensor s within Euclidean distance $|ps| \leq r_{\text{sense}}$. Where r_{sense} is the *sensing radius* of the sensor nodes, i.e. the radius within which they can monitor or estimate temperature or humidity. Unfortunately, not all sensors will be operational upon reaching the ground. Some of them might fall right into the flames and be destroyed, others might plunge into a lake or pond and be unable to perform their monitoring task. Paradoxically we are particularly interested in those areas where there's an ongoing fire (and maybe also where there is a lake or pond), but sensor nodes that fell into these areas are unable to report this fact.

*Part of this work was done while the first author was member of the Guibas Lab at Stanford University. The first author was supported by the Max Planck Center for Visual Computing and Communication (MPC-VCC) funded by the German Federal Ministry of Education and Research (FKZ 01IMC01).

SCG'06, June 5–7, 2006, Sedona, Arizona, USA.

We want to detect the (boundaries of) such holes in the monitored space created by fire or other phenomena via examination of the *communication graph* of the wireless nodes. The *communication graph* of a wireless network has a node for each wireless station and an (unweighted) edge between two nodes if the respective stations can communicate with each other. For simplicity let us assume that two nodes can communicate with each other if they are within distance of at most 1 (communication radius). So the communication graph is a *unit disk graph* (UDG). Typically, the communication radius is considerably larger than the sensing radius r_{sense}, let $\kappa = 1/r_{\text{sense}}$ be the ratio between these two quantities. Clearly, the larger the value κ becomes, the denser the communication graph gets.

In essence, the problem that we consider in this paper is that of identifying holes just by examining a communication graph. If in a sufficiently large region sensors break down, this hole will also manifest itself in the communication graph, hence holes identified using the communication graph are indicative of some large-scale special event in the region to be monitored. By 'identifying holes' we mean (a) that for every point on the boundary of a hole we want the algorithm to mark a sensor node nearby and (b) every sensor node marked by the algorithm lies near a boundary of a hole.

Related Work

In Fang et al. [3] the authors present an algorithm for detecting holes for the case where the individual nodes know about their geographic location. Fekete et al. in [5] describe a method to identify boundaries in a wireless network which does *not* require the nodes to be aware of their geographic position; their method assumes a *uniform* distribution of the network nodes in non-hole areas, though. In a more recent paper [4] the same authors present a deterministic approach for boundary recognition which does not rely on a uniform node distribution. Their paper also proposes interesting methods to aggregate the information gathered by the boundary recognition step in a higher-level topology sketch of the network. Their boundary detection algorithm does not come with a theoretical correctness guarantee and appears to require a rather high node density in practice, though. In a recent contribution [6] we have developed a heuristic boundary detection algorithm based on a similar intuition as the algorithm presented in this paper; unfortunately correctness cannot be proven for this heuristic algorithm (see Section 2.1 for a brief explanation of the problems with this approach); furthermore its computation is not localized as it requires the computation of distance fields over the whole network.

Our Contribution

In this paper we present an algorithm for detecting hole boundaries in a wireless network that is represented purely by its communication graph. If the structure of the communication graph is a unit-disk graph determined by the geographic locations of the nodes, we can prove under some (rather strong) additional conditions that our algorithm correctly identifies nodes near boundaries and never misclassifies any nodes. The employed proof technique and sampling condition has some similarity to the one used in the area of shape reconstruction [1]. The algorithm is very simple and has running time linear in its input. While the theoretical analysis is not very satisfying in that it makes rather strong assumptions about the input setting, our experimental results show that the algorithm performs much better in practice, as it is the case for many shape reconstruction algorithms. The idea of using the geometry information hidden in the connectivity structure of the communication graph to identify topological features has only recently been addressed [4, 5, 6]; the algorithm presented here is to our knowledge the first that comes with some formal guarantee.

As a general topic of interest we propose the problem of recovering geometric information from purely combinatorial connectivity information as it arises for example as communication graphs in the context of wireless networks. While such communication graphs do not explicitly contain geographic location information, they were created based on certain geometric properties and hence implicitly bear some geometry information. Recovering or 'reverse engineering' this information appears to be an interesting challenge. In this paper we have shown that the implicit geometry information suffices to identify certain topological features of the network. The question is how much more can be accomplished using geometry information hidden in the connectivity of the network.

2 Intuition and the Algorithm

In this section we will first introduce the intuition in the continuous setting which our algorithm is based upon. After presenting the algorithm we analyze its runtime at the end of this section. Correctness will be proven in the following section.

2.1 Intuition in the Continuous Case

Picture the following continuous variant of our problem: Given a (possibly non-simply) connected region $R \subset \mathbf{R}^2$ and some point $p \in R$ – we call that point *seed* –, we consider the isolevels of the *geodesic distance* function d_p from p in the domain R as in Figure 2. That is, for any point $x \in R$, $d_p(x)$ denotes the minimum Euclidean length of an open curve $\Gamma \subset R$ with one endpoint being p, the other x. Or in other words, $d_p(x)$ is the length of the shortest path from p to x which stays within R and avoids all holes.

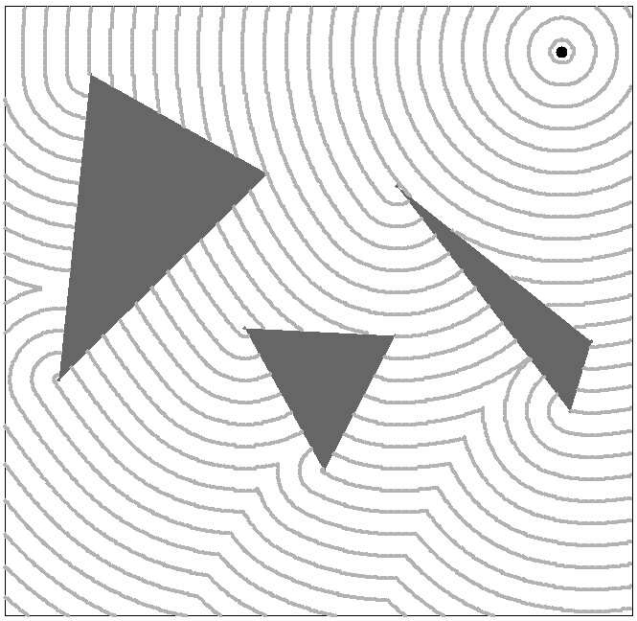

Figure 2. The continuous case: One seed, 3 triangular holes, induced contours/isolines.

The *isolevel, isoline*, or *contour* of level k of the distance function d_p is the set of points $I(k) = \{x \in R : d_p(x) = k\}$. In Figure 2 we have depicted the contours of level $10, 30, 50, \ldots$. If the region

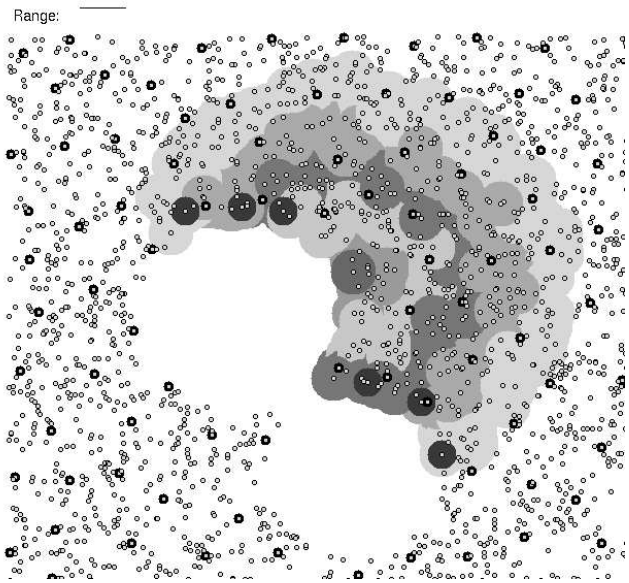

Figure 3. Snapshot while executing the algorithm: isolevels examined from one seed and marked nodes where levels 3,4,5 are broken (large black circles). Small black circles denote the seed set I, little dots the remaining nodes.

R is free of holes and all points on the boundary of R can be seen from p (without obstruction by a hole), the contour of level k is a subset of the circle centered at p with radius k. In the more general case with polygonal obstacles, though, the contour of level k is a collection of (possibly disconnected) circular arcs.

What is interesting for our purposes is the observation that for almost every point x of a hole/outer boundary, some contour is *broken* at that point. We say a contour is *broken* at a point p or p is called an *endpoint* of that contour, if the contour does not intersect an arbitrarily small ball around p in a topological 1-disk. The only way that a boundary point x with $d_p(x) = k$ might not be the endpoint of a component of the contour of level k is in case that the tangents of the contour of level k and of the boundary agree at x. In Figure 2, this is the case for example in the lower right and upper left corner of the region R, where the 'wave-fronts' hit the outer boundary tangentially. The same happens at the upper left tip of the rightmost triangular hole in the picture. The 'reverse observation' – that every breakpoint of a contour is also on a hole/outer boundary – is unfortunately not true, which was incorrectly claimed in [6]. When several holes/obstacles are arranged in a particular order, broken isolevels that are not near any boundary might appear (e.g. 3 wave-fronts that meet each other in a single point induce a contour which consists of a single point). Nevertheless, when restricting for example to circular holes and only isolevels near the seed that do not interact with more than one hole, actually every breakpoint of a contour coincides with a hole/outer boundary point and vice versa.

The key idea of our hole detection routine is to make use of these 'broken isolines' to determine nodes that are close to the outer boundary or to a boundary of some hole. In the following we describe a rather straightforward translation of the intuition from the continuous setting to the discrete setting. In the absence of geographic location information, the only distance measure available is the hop distance in the unit-disk graph (the graph distance where the weight of every edge is 1).

2.2 The Algorithm

Mimicking the continuous case, we pick a set of nodes I in the network to serve as seeds and determine hop-distances in a bounded neighborhood of $h_{\max}$ hops around each $s \in I$. We then examine the isolevels around s and try to detect whether these isolevels form closed cycles/annuli or are broken up. This can be done by repeated shortest path computations *within* the subgraph induced by the nodes in the respective isolevel L: We first compute graph distances $d_1(.)$ within L from some arbitrary node $v_1 \in L$. We then set v_2 to be a node furthest from v_1. Let v_2' be the node on a shortest path from v_1 to v_2 at distance $\lfloor d(v_1, v_2)/2 \rfloor$. We remove all nodes within a 2-hop neighborhood of v_2'. If in the subgraph induced by $L \setminus \{v' : d(v', v_2') \leq 2\}$ there still exists a path from v_1 to v_2, we take this as an indication that the respective isolevel has a circular shape and return without marking any nodes as being close to a boundary. Otherwise we compute distances $d_2(.)$ from v_2; let v_3 be a node furthest from v_2. After the same check for connectivity without the neighborhood of a node v_3' halfway between v_2 and v_3, we take it for granted that the isolevel is broken with v_2 and v_3 being close to the extreme points of that isolevel.

The high-level description of the algorithm can be found in Figure 4, and the more detailed description of the isolevel examination can be found in Figure 5. See Figure 3 for a snapshot of the algorithm when examining the isolevels 3 to 5 from one seed node.

Restricting the seed set to a maximal independent set allows us to bound the overall running time of the algorithm. The isolevel examination routine only considers isolevels that are connected, since arguing about distances within small connected components seems rather difficult. Nevertheless we will show later in the theoretical analysis, that our algorithm still doesn't miss any boundary, since for any point p on the boundary of a hole, there is at least one seed node $s \in I$ such that some connected isolevel around s is cut near p, and some node near p is actually marked as being on the boundary.

The outcome of the algorithm is a set of marked nodes (hopefully) all close to some boundary and identifying all holes. As the marking happens independently it might be the case that a node is marked several times or neighboring nodes are marked. For a compact representation it might be desirable to only keep a maximal independent set of the marked nodes[1]. On the other hand, if a closed boundary representation is desired, one can locally compute connecting paths between the nodes of the maximal independent set of the marked nodes.

HoleDetection(G)

1. Compute a maximal independent set I in G
2. For each $s \in I$:
   (a) compute shortest distances to s within G up to a distance of $h_{\max}$ hops
   (b) for each $h_{\min} \leq i \leq h_{\max}$ examine the isolevel i consisting of the set of nodes $\{v : d(s, v) = i\}$ by calling
   ExamineISOLevel(G, s, i)
3. Return all marked nodes

Figure 4. The high-level view of the hole detection algorithm.

[1] That is what we have done in our implementation to allow for better visual inspection of the results.

ExamineISOLevel(G, s, i)

1. if $L = \{v : d(s, v) == i\}$ induces a subgraph with more than 1 connected component, return
2. Choose an arbitrary node $v_1 \in L$, compute the hop-distance function $d_1(.)$ to v_1 within L
   (a) Set $v_2 :=$ argmax $d_1(.)$ and let v_2' be the node on a shortest path from v_1 to v_2 at distance $\lfloor d(v_1, v_2)/2 \rfloor$
   (b) if there exists a path from v_1 to v_2 in the graph induced by $L \setminus \{v \in L : d(v, v_2') \leq 2\}$ return
3. compute the hop-distance function $d_2(.)$ to v_2 within L
   (a) Set $v_3 :=$ argmax $d_2(.)$ and let v_3' be the node on a shortest path from v_2 to v_3 at distance $\lfloor d(v_2, v_3)/2 \rfloor$
   (b) if there exists a path from v_2 to v_3 in the graph induced by $L \setminus \{v \in L : d(v, v_3') \leq 2\}$ return
4. mark v_2 and v_3 as nodes on the boundary and return

Figure 5. The Isolevel examination subroutine.

2.3 Running Time

The input to the algorithm is an undirected graph $G(V, E)$ with $|V| = n, |E| = m$. While the algorithm can be run essentially on any graph, its running time as well as the sense of the output depends on G being a *unit-disk graph* in $\mathbb{R}^2$.

THEOREM 2.1. *On unit-disk graphs $G(V, E)$ in $\mathbb{R}^2$ with $|V| = n$, $|E| = m$, the hole detection algorithm run with constant $h_{\max}$ runs in $O(n + m)$ time.*

PROOF. In the first stage the algorithm needs to compute a maximal independent set I. This can be done in a greedy fashion in $O(n + m)$ by repeatedly selecting a node into I and removing all its neighboring nodes. The running time of the second part of the algorithm is determined by the examinations of all nodes and edges within distance of $h_{\max}$ hops around each node $s \in I$. Let us denote by N_s the edges within distance $h_{\max}$ hops from s (we call an edge to be within distance of h hops from s if one of its endpoints has hop distance at most h). Since we assume G to be connected, we can absorb the number of nodes within $h_{\max}$ hops from s using a constant factor c in $|N_s|$. The total running time of the second part is then

$$\sum_{s \in I} c|N_s| = \sum_{s \in I} c \cdot (\sum_{e \in E} (e \in N_s)) = c \cdot (\sum_{e \in E} \sum_{s \in I} (e \in N_s))$$

Here the second sum can also be interpreted as the number of elements $s \in I$ which are at distance at most $h_{\max}$ from one of the endpoints of edge $e \in E$. But this number is bounded by a constant (for constant $h_{\max}$) as can be seen as follows: Draw a ball of radius $1/2$ around each $s \in I$ which is at most $h_{\max} + 1$ hops away from one of the endpoints of e – say v. Clearly these balls are all disjoint (o.w. there would be an edge between the respective nodes and they could not be in an independent set) and are contained within a larger ball $B(v, h_{\max} + 3/2)$. Hence there are at most $2\pi(h_{\max} + 3/2)^2/(\pi/4) = O(h_{\max}^2)$ of them. So the overall running time for the second step sums up to

$$\sum_{s \in I} c \cdot |N_s| = c \cdot \sum_{e \in E} O(h_{\max}^2) = O(m)$$

for constant $h_{\max}, c$. □

2.4 Distributed and localized implementation

For application in a wireless sensor network scenario, it is important that the employed algorithms can be implemented without a centralized control. Fortunately, the formulation of our algorithm allows for straightforward localized implementation. Many algorithms like [8] are known for computing a maximal independent set in a distributed manner. The isolevel examinations themselves are inherently local (restricted to a constant size neighborhood of the respective seed nodes). This is another difference from the heuristic approach presented in [6] where 4 distance functions over the *whole network* had to be computed. Furthermore the (particularly outer) isolevels extend over a long distance within the network and cannot be examined by local computations.

2.5 Extension to $\mathbb{R}^3$

Our algorithm can in fact be extended to $\mathbb{R}^3$ in a straightforward manner. Assuming the sensor nodes densely sample the interior volume of some body possibly with holes or cavities, the goal is to mark sensor nodes that are close to the boundary of the body. But again we can first determine an independent set I and then compute hop-distances from each seed $s \in I$. In the absence of any holes/cavities an isolevel looks like the surface of a sphere; if there are holes, though, 'this surface of a sphere' exhibits holes. That is, for each isolevel in the $\mathbb{R}^3$ setting, we are essentially faced with a hole detection problem in $\mathbb{R}^2$ for which we have already presented a solution. We have not implemented nor analyzed this algorithm but hope to do so in the near future.

3 Analysis

In this section we show that our algorithm under some conditions faithfully marks nodes near all hole boundaries. More precisely, we show that if we have circular holes with a certain minimum radius r_{hole} and a minimum distance between holes Δ_{hole}, there exists a lower bound on the node density in the non-hole areas and respective communication ranges for the sensor nodes such that for every point on a hole boundary, some network node nearby is marked as being close to the boundary, and every marked network node has some boundary nearby.

We need to make the following assumptions about the distribution of the sensors. A priori, the region of interest is the whole two-dimensional plane $\mathbb{R}^2$. We subtract from $\mathbb{R}^2$ a set of (disjoint) holes O and assume the following sampling condition: We call a set S of sensors an ϵ-good sensor distribution if

$$\forall p \in (\mathbb{R}^2 \setminus \cup_{o \in O} o) : \exists s \in S : |sp| \leq \epsilon,$$

where $|.|$ denotes the *Euclidean distance* (all other distance functions in the following refer to the *Graph distance*). That is, for all points of $\mathbb{R}^2$ not belonging to a hole there is a 'nearby' sensor. This is a reasonable assumption if one also wants to ensure that in 'regular' areas of $\mathbb{R}^2$, where sensors have survived, sensing coverage is guaranteed. Note that this definition of holes also allows to consider a bounded region of interest. For example a square region Q of interest can be realized by defining a 'hole' $o := \mathbb{R}^2 - Q$ in O. For convenience let us denote by $\mathcal{R} := \mathbb{R}^2 \setminus \cup_{o \in O} o$ the area where sensors have survived.

In the following analysis we restrict ourselves to the case of *circular* holes. We believe that an extension of the proof to convex fat objects is rather straightforward (we haven't fully worked that out yet). For non-convex objects, a more complicated analysis and maybe also a modified algorithm seems necessary, though. In the next few subsections we will proof several Lemmas each of which

requires certain constraints on the choice of the parameters Δ_{hole}, $h_{\min}$, $h_{\max}$, ϵ, r_{hole}. At the end, in Section 3.3, we will collect all the implied constraints on these parameters and show that they can be chosen such that all Lemmas hold: if holes have radius at least $r_{\text{hole}} = 72$, the minimum distance between two holes is at least $\Delta_{\text{hole}} = 18$, the region is sampled with $\epsilon \leq 1/64$, and our algorithm inspects isolevels $h_{\min} = 4$ to $h_{\max} = 8$, we can guarantee correctness of the output.

3.1 Levels and containing Annuli

Let us now define *levels* with respect to a specific seed node $s \in S$.

Definition 3.1. *Denote by $\mathcal{L}^i(s)$ the* level *i of seed s. We define:*

- $\mathcal{L}^0(s) = \{p \in \mathcal{R} : |ps| \leq 1\}$
- $\mathcal{L}^{i+1}(s) = \{p \in (\mathcal{R} \setminus \cup_{j \leq i} \mathcal{L}^j(s)) : \exists s' \in S \cap \mathcal{L}^i(s) : |s'p| \leq 1\}$

Intuitively, level $\mathcal{L}^{i+1}(s)$ are the points in $\mathcal{R}$ which are within the communication radius of all nodes contained in level $\mathcal{L}^i(s)$ but not within the communication radius of nodes in levels $\mathcal{L}^{<i}(s)$.

Let us now argue about the shape of these levels. If no holes are present, $\mathcal{L}^0(s)$ is simply a disk of radius 1, and we expect the levels $\mathcal{L}^{i \geq 1}$ to be similar to annuli. In the presence of holes, though, parts of these annuli will be cut off.

Lemma 3.1. *If $r_{\text{hole}} \geq (i+1)/\sqrt{\epsilon}$ we have $\forall i, 0 \leq i \leq \Delta_{\text{hole}}/2 - 1$*

$$B(s, i - 4i\epsilon + 1) \cap \mathcal{R} \subseteq \cup_{j \leq i} \mathcal{L}^j(s) \subseteq B(s, (i+1)) \cap \mathcal{R}$$

Proof. The upper bound is obvious, the lower bound requires proof, though. The case $i = 0$ is trivial, let us now consider the case $i > 0$ and assume that $B(s, i - 4i\epsilon + 1) \subseteq \cup_{j \leq i} \mathcal{L}^j(s)$, where $B(c, r)$ denotes the ball with radius r centered at c. Now consider a point $p \in \mathcal{R}$ with $i - 4i\epsilon + 1 < |ps| \leq (i+1) - 4(i+1)\epsilon + 1$ and the intersection $I := B(p, 1) \cap B(s, i - 4i\epsilon + 1)$. If we can show that there exists a ball of radius ϵ contained in I and with center in $\mathcal{R}$ we are done since due to the sampling condition we know that there must be a sample contained in this ball and hence p lies within the communication range of this sample point. The difficulty lies in the fact that not necessarily all of I has to be contained in $\mathcal{R}$. Nevertheless, by requiring the circular obstacles to be rather large, we can prove that there exists a sufficiently large area in $I \cap \mathcal{R}$.

Let M be the point on $\overline{ps}$ between p and s with $|pM| = 1 - 2\epsilon$, m be the line perpendicular to $\overline{ps}$ and passing through M. We are interested in finding a point Z on m which is as far as possible away from M such that a ball of radius ϵ centered at Z is still contained in I. Some simple trigonometry shows that if Z has distance at most $|ZM| \leq \sqrt{\epsilon}$ from M for $\epsilon \leq 1/3$, we have $B(z, \epsilon) \subset I$.

If $\Delta_{\text{hole}} \geq 2(i+1)$, clearly only one hole can interfere with $\mathcal{L}^i(s)$. Assume w.l.o.g. that the respective circular hole o has its center *below* the line $\overline{ps}$. In the worst case, i.e. obstructing as much from I as possible, o has both s and p on its boundary and 'eats away' the whole portion of I that lies below $\overline{ps}$. But since the radius of o has to be at least $(i+1)/\sqrt{\epsilon}$ it cannot extend further than $\sqrt{\epsilon}$ above $\overline{ps}$ as an elementary trigonometric calculation shows.

Therefore, there is always a point z in $I \cap \mathcal{R}$ such that $B(z, \epsilon) \subset I$. And hence any point $p \in \mathcal{R}$ with $i - 4i\epsilon + 1 < |ps| \leq (i+1) - 4(i+1)\epsilon + 1$ is certainly contained within $\mathcal{L}^{i+1}(s)$. □

We denote by $\mathcal{A}(s, r_1, r_2) := \{p \in \mathbb{R} : r_1 \leq |ps| \leq r_2\}$ the *annulus* with center s, inner radius r_1 and outer radius r_2.

Corollary 3.1. *The i-th level $\mathcal{L}^i(s)$ is contained in the intersection of $\mathcal{A}^i_{\text{outer}}(s) := \mathcal{A}(s, i - 4(i-1)\epsilon, i+1)$ with $\mathcal{R}$.*

Corollary 3.2. *The i-th level $\mathcal{L}^i(s)$ contains the intersection of the annulus $\mathcal{A}^i_{\text{inner}}(s) := \mathcal{A}(s, i, i - 4i\epsilon + 1)$ with $\mathcal{R}$.*

3.2 Non-contractible graph cycles

When our algorithm examines an isolevel i (which is the set of nodes contained in level $\mathcal{L}^i(s)$) it first tries to decide whether there exists a cycle containing s in its interior.

Definition 3.2. *We call a cycle in the graph induced by the nodes in isolevel i* non-contractible *if its corresponding polygon cannot be contracted to a single point without sweeping over s (the seed point).*

Lemma 3.2. *If $\mathcal{L}^i(s)$ contains $\mathcal{A}_{\text{inner}}(s, i, i - 2i\epsilon + 1)$, our algorithm finds a non-contractible cycle and returns without marking any nodes as being close to a boundary.*

Proof. The containment of $\mathcal{A}^i_{\text{inner}}(s)$ within $\mathcal{L}^i$ guarantees that there is an alternative path from v_1 to v_2 in the subgraph induced by $L \setminus \{v \in L : d(v, v'_2) \leq 2\}$ via the part of the annulus 'opposite' of v'_2 (it cannot pass nearby v'_2 since the removal of the 2-hop neighborhood around v'_2 has cut $\mathcal{L}^i(s)$ there). The original path from v_1 to v_2 together with the alternative path yields a non-contractible cycle. □

This Lemma guarantees that if no obstacles are near $\mathcal{L}^i(s)$, no nodes in level i will be marked while examining $\mathcal{L}^i(s)$. Let us now show that whenever our algorithm marks a node, there is some boundary point nearby:

Lemma 3.3. *If the isolevel examination routine upon processing $\mathcal{L}^i(s)$ marks nodes v_2, v_3 as being close to the boundary, there exist boundary points $p_1 \in \delta o_1$, $p_2 \in \delta o_2$ with $|v_i p_i| \leq 2.8$ and furthermore we have that the graph distance $d^G(v_2, v_3)$ is at least $\pi(i - 2(i-1)\epsilon)$.*

Proof. Consider the neighborhood within distance 2.8 of v_2. If there is some hole boundary nearby, we're done, so assume otherwise. Let t_{l1}, t_{l2} and t_{r1}, t_{r2} be the intersection points of $B(v_2, 2.8)$ with the boundary of $\mathcal{A}^i_{\text{inner}}(s)$, q_l the point on $B(v_2, 2.8)$ between t_{l1} and t_{l2} (and q_r analogously). Let further s_l, s_r be the closest nodes to q_l, q_r respectively, see Figure 6 for an illustration. Clearly s_l and s_r are contained within $\mathcal{A}^i_{\text{inner}}(s)$. We have $d_1(s_{l/r}) \leq d_1(v_2) \leq d_1(s_{l/r}) + 3$ (the first inequality holds by definition, if the second wasn't true, it would be easy to exhibit a shorter path to v_2). W.l.o.g. assume that the shortest path from v_1 to v_2 passes by s_l. Then the shortest path from v_1 to s_r also has to pass by s_l otherwise our routine would have exhibited an alternative path from v_2 to v_1 passing by s_r and avoiding the neighborhood of v'_2. But this implies that $d_1(s_r) \geq d_1(s_l) + 4$ contradicting the above inequality which implies $d_1(s_r) \leq d_1(s_l) + 3$. The same argumentation also holds for v_3 and the distance function $d_2(.)$. The statement about the graph distance easily follows from the fact that in case of circular holes at most half of the containing annulus of $\mathcal{L}^i(s)$ is cut off. □

It remains to show that for every boundary point p there exists a seed $s \in I$ such that our algorithm inspects an isolevel of s broken 'close' to p, for which it does not find a non-contractible cycle and for which it marks a node near p.

Lemma 3.4. *Let $p \in \delta o$, $o \in O$ be a point on a hole boundary. Then there exists $s \in I$ and i, $h_{\min} \leq i \leq h_{\max}$ such that the examination of isolevel $\mathcal{L}^i(s)$ reaches step 4 and a node within distance at most* 4.8 *from p is marked.*

Proof. Consider the tangent t to o at p and some point s' on t at distance $(h_{\max} - 3)$ from p. Let $s \in I$ be the closest seed to s'. Clearly $|ss'| \leq 1 + \epsilon$ due to the sampling condition and $h_{\max} <$

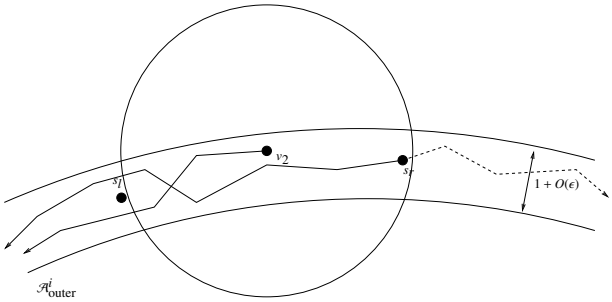

Figure 6. Situation for extremal node v_2 under $d_1(.)$.

Δ_{hole} and because any maximal independent set in a graph is also a dominating set. Let s_p be the sensor closest to p; clearly $|ps_p| \leq \epsilon$ due to the sampling condition. We have $|ss_p| \leq h_{\max} - 1$. Let $\mathcal{L}^i(s)$ be the level which contains s_p. As $h_{\max} < r_{\text{hole}}$, the contained annulus $\mathcal{A}^i_{\text{inner}}(s)$, the containing annulus $\mathcal{A}^i_{\text{outer}}(s)$ (and also $\mathcal{L}^i(s)$) are completely cut by o but remain connected (because $2h_{\max} < \Delta_{\text{hole}}$, i.e. there is no other hole 'nearby'). And since the removal of the 2-hop neighborhood of v'_2 again cuts $\mathcal{L}^i(s)$ around v'_2 there does not exist an alternative path from v_1 to v_2 (same argumentation for v'_3), so the algorithm does not return in steps 1, 2b, or 3b. We still need to show that the algorithm marks a node near p. Since i is small compared to the radius of the obstacle o, $\mathcal{L}^i(s)$ is almost orthogonally cut by o. So the nodes within $\mathcal{L}^i(s)$ that are within ϵ distance from o are grouped into two small clusters of diameter ≤ 2 that are far away from each other in graph distance. According to Lemma 3.3 our algorithm marks one in each of the clusters. The distance of the marked node to p follows from the cluster diameter and the previous Lemma. □

3.3 Plugging everything together

With the above Lemmas we have shown that for rather large and well-separated (relative to the communication range 1 of the sensor nodes) circular holes and sufficiently dense (but not necessarily uniform) distribution of sensor nodes our algorithm faithfully marks nodes close to the boundaries of all holes and no others. We have several conditions on the parameters Δ_{hole}, $h_{\min}$, $h_{\max}$, ϵ, r_{hole}: For the inner annuli not getting too thin and the outer annuli not getting too thick, we want $4h_{\max}\epsilon \leq 1/2$, for Lemma 3.1 we need $h_{\max} \leq \Delta_{\text{hole}}/2 - 1$ as well as $r_{\text{hole}} \geq (h_{\max} + 1)/\sqrt{\epsilon}$. Furthermore for Lemma 3.4 we need $h_{\max} - h_{\min} \geq 3$ and $h_{\min} \geq 4$ to guarantee that the removal of the neighborhood of v'_2/v'_3 does not also remove v_1 or v_2 / v_2 or v_3 respectively. We haven't tried hard to determine the best parameter values, but the following values allow us to state the main theorem:

THEOREM 3.1. *For* $\Delta_{\text{hole}} = 18$, $h_{\min} = 4$, $h_{\max} = 8$, $\epsilon = 1/64$, $r_{\text{hole}} = 72$ *our hole detection algorithm marks for each boundary point a sensor node within distance* 4.8*, and every marked sensor node has a boundary point within distance* 2.8.

We want to emphasize that this analysis is very pessimistic and as we will see in the experimental section, the algorithm works very well under much smaller sampling density as well as for non-circular, non-convex holes.

4 Experimental Results

We have implemented our algorithm using the LEDA software library and simulated different deployments of sensor nodes in the real plane. As to be expected, in practice the requirements on the node density as well as the shape and distance between holes are by far not as strong as the theoretical analysis suggests. Our experiments were conducted with algorithm parameters $h_{\min} = 2$, $h_{\max} = 6$, and instead of removing all nodes within a 2-hop neighborhood of v'_2/v'_3 we only removed nodes within a 1-hop neighborhood. We first generated sensor nodes uniformly at random and then built the unit-disk graph based on varying communication ranges. See Figures 7, for examples of the algorithm's output.

Our observation was that for node distributions where the average degree in the resulting communication graph was above around 25, our algorithm seems to perform reasonably well. We note that for more regular node distributions like a grid or a perturbed grid, our algorithm also works very well for average node degrees of about 10. Compared to the heuristic approach presented in [6] the algorithm does slightly worse, mainly because we have avoided to add other heuristics that improve the practical performance but would make the algorithm somewhat different from the one we have proven correctness for.

Non-uniform node-distributions

The whole reasoning why our algorithm actually works does not rely on a uniform density of the sensor nodes in non-hole regions, as for example the algorithm presented in [5]. The uniform distribution case is simpler as one can roughly speaking determine the status of a node – being close or not close to some boundary – by examining to how many neighbors it can talk to. For nodes not close to some boundary, this should be always around the same number, whereas nodes close to a boundary should have less immediate neighbors they can talk to. See Figure 8 for an example of a non-uniform sensor distribution and our algorithm's output.

Shortcomings of the Algorithm

There are settings, of course, where the algorithm does not perform as well as in all the examples shown before. It has, for example problems, distinguishing holes that are close to each other (remember in the analysis we required a certain minimum distance Δ_{hole} between holes). See Figure 9 for an example. The problem here is that essentially for all seeds that could detect the boundary nodes between the two holes, the respective isolevels consist of more than one component and hence the isolevel examination is aborted. One might also let the algorithm examine all the connected components of an isolevel if there are more than one, but this would not allow for a proof of correctness in the circular hole case anymore.

Also, our algorithm heavily relies on the non-hole regions being sufficiently densely sampled such that the isolevels around a seed node form closed cycles. When decreasing the node density (or equivalently decreasing the communication range) the algorithm breaks down more and more. See Figure 10 for a sequence of outputs with decreasing communication ranges. The roughly equivalent value for ϵ is determined as $\epsilon \approx 1/\sqrt{\text{avg.deg.}}$; in these cases that would be 0.18, 0.21, and 0.25 .

5 Discussion

The tight connection between geographic location and connectivity of wireless networks gives rise to many interesting problems. While the topology of a wireless network does not explicitly hold any geometric information, the fact that its connectivity is determined by geographic proximity relations allows for techniques to extract (part) of the underlying geometry. This paper and few papers before have explored that direction but we believe that there

are still many challenging problems to be solved. Ultimately, of course, one would like to recover the exact relative positions of the network nodes; unfortunately this problem is NP-hard for general unit-disk graphs as was shown by Kuhn et al. in [7], but a constant approximation has not been ruled out yet.

Apart from the application in the described scenario, the identification of hole boundaries or holes as such has in fact many other applications. A topological description of a network like 'The network has one large hole' can be much more compact than remembering the connectivity between all nodes. Furthermore such a topological description tends to be more stable under small changes of nodes or network links (only many changes might close a large hole or create another one). There are routing schemata like [2] which are based on such topological 'sketches' of the network; simulation results show that they enjoy an inherent stability against small network changes. The recent work by Fekete et al. [4] also describes methods how to use boundary recognition to obtain a compact topology sketch of the network. Another application of the boundary detection routine appears in methods that – in spite of its NP-hardness – try to find a faithful embedding of the whole unit-disk graph in the plane, like the work by Rao et al. [9]. In their algorithm the decisions of the algorithm are based on the assumption that hop-distances in the graph in some sense approximate Euclidean distances in the plane. This doesn't have to be true in the presence of holes, though, since these might cause the geodesic and the graph distance to differ vastly from the Euclidean distance. If holes are detected, though, one could check whether a graph distance – measured by a shortest hop path in the graph – is 'truthful' by assuring that the respective shortest path does not come close to any hole boundary (in which case it might not reflect Euclidean distances). See [6] for more details.

References

[1] N. Amenta, M. Bern, and D. Eppstein. The crust and the β-skeleton: combinatorial curve reconstruction. *Graph. Models Image Process.*, 60(2):125–135, 1998.

[2] J. Bruck, J. Gao, and A. Jiang. Map: Medial-axis-based geometric routing in sensor networks. Proc. *MobiCom*, 2005.

[3] Q. Fang, J. Gao, and L. Guibas. Locating and bypassing routing holes in sensor networks. In *23rd Conf. of the IEEE Communications Society (INFOCOM)*, 2004.

[4] S. P. Fekete, A. Kröller, D. Pfisterer, and S. Fischer. Deterministic boundary recognition and topology extraction for large sensor networks. In *Proc. of SODA*, 2006.

[5] S. P. Fekete, A. Kröller, D. Pfisterer, S. Fischer, and C. Buschmann. Neighborhood-based topology recognition in sensor networks. In *ALGOSENSORS*, 2004.

[6] Stefan Funke. Topological hole detection in wireless sensor networks and its applications. In *DIALM-POMC: Joint workshop on Foundations of mobile computing*, 2005.

[7] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Unit disk graph approximation. In *Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, 2004.

[8] T. Moscibroda and R. Wattenhofer. Maximal independent sets in radio networks. In *24th ACM Symp. on the Principles of Distributed Computing (PODC)*, 2005.

[9] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *Proc. MobiCom*, 2003.

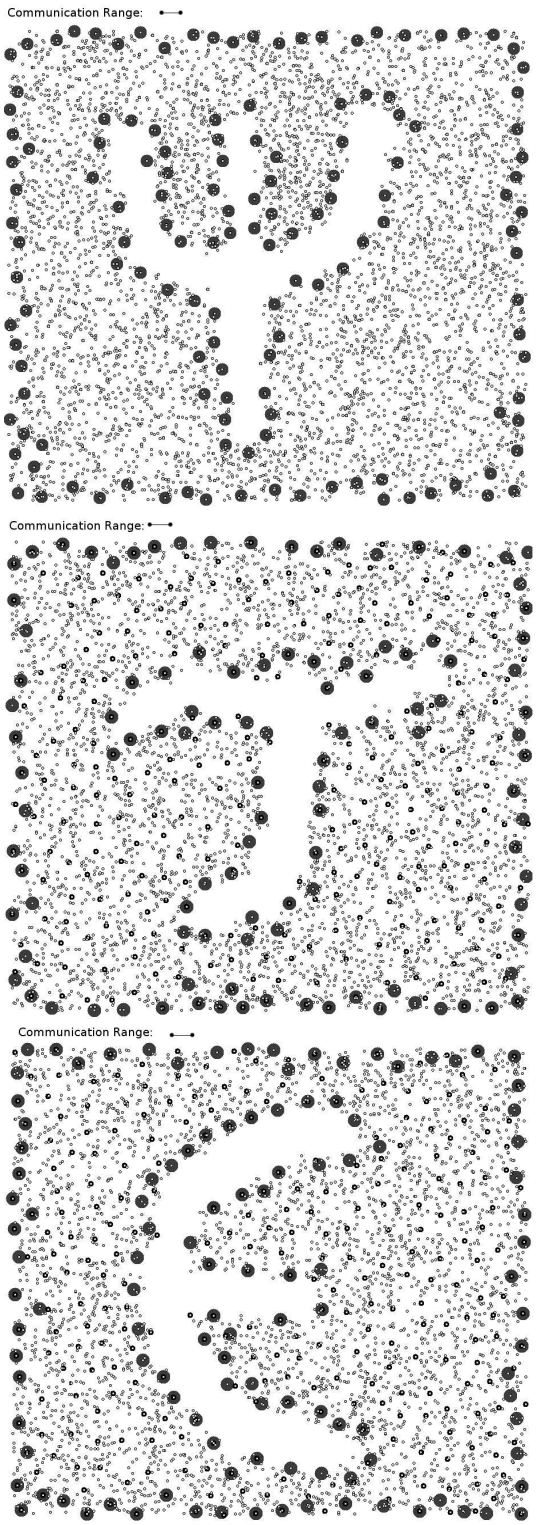

Figure 7. Examples for algorithm output; around 5000 randomly distributed nodes, communication ranges of 30, 33, 33 and respective average degrees of the communication graph of 25, 27, 28.

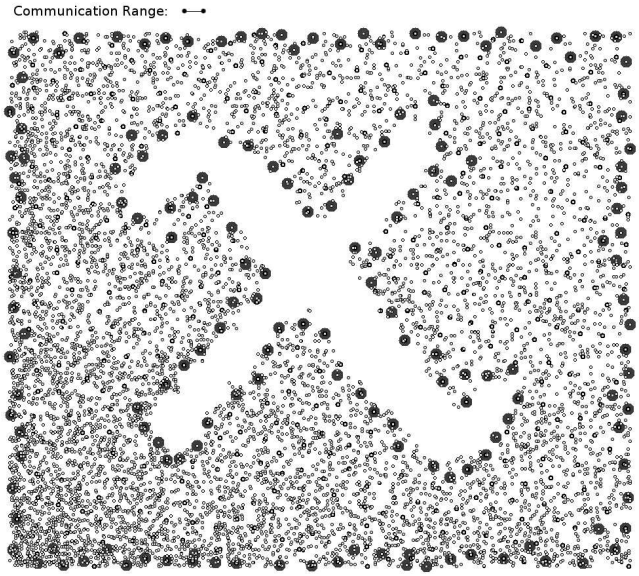

Figure 8. Algorithm output in a non-uniform sensor node deployment.

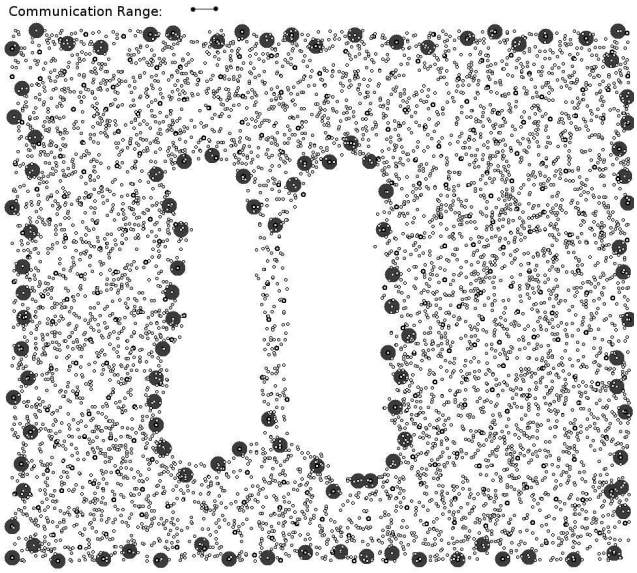

Figure 9. Problems of the algorithm when holes are too close to each other.

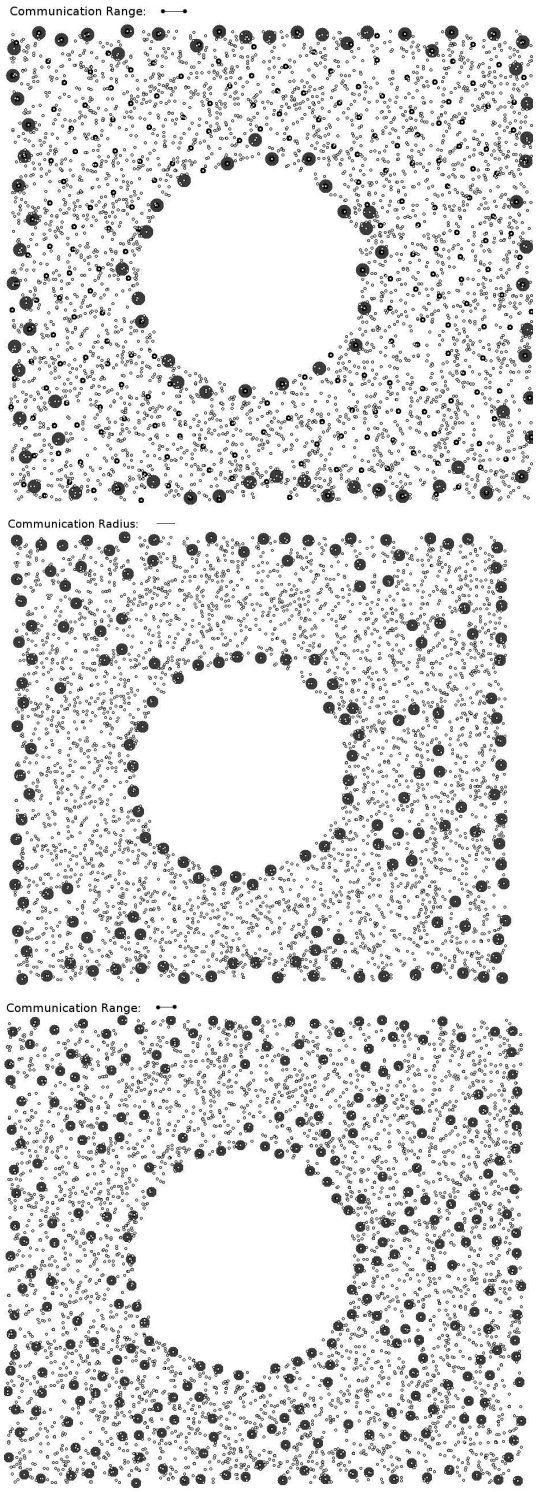

Figure 10. Output of the algorithm for decreasing communication ranges 35, 30, and 25 (respective avg. degrees of the communication graph 31, 23, 16).