On Boundary Recognition without Location Information in Wireless Sensor Networks

Olga Saukh[1], Robert Sauter[1], Matthias Gauger[1], Pedro José Marrón[1], Kurt Rothermel[2]
[1]Universität Bonn and Fraunhofer IAIS, [2]Universität Stuttgart, Germany
{saukh,sauter,gauger,pjmarron}@cs.uni-bonn.de, rothermel@ipvs.uni-stuttgart.de

Abstract

Boundary recognition is an important and challenging issue in wireless sensor networks when no coordinates or distances are available. The distinction between inner and boundary nodes of the network can provide valuable knowledge to a broad spectrum of algorithms. This paper tackles the challenge of providing a scalable and range-free solution for boundary recognition that does not require a high node density. Our solution approximates the boundary of the sensor network by determining the inner nodes using geometric constructions that guarantee that, for a given d, a node lies inside of the construction for a d-quasi unit disk graph model of the wireless sensor network. Moreover, such geometric constructions make it possible to compute a guaranteed distance from a node to the boundary. We provide a thorough evaluation of our approach and show that it is applicable to dense as well as sparse deployments.

1. Introduction

In wireless sensor networks (WSN) it is frequently necessary to know the topology of the deployed network as well as specifics on the coverage of the target area. Such information can be extracted easily if a significant subset of nodes is equipped with GPS receivers. However, such hardware is expensive and requires a lot of energy. In this paper we present a scalable solution that, without the use of position information, recognizes certain topological properties such as inner and boundary nodes in a two-dimensional space. Our approach sets no constraints on the node distribution and the node density. Additionally, each inner node is assigned a minimum guaranteed distance to the boundary.

Many WSN applications demonstrate the need for the extraction of such topological information [6, 8, 11]. For example, since the failure of boundary nodes results in reduced coverage, the load of these nodes should be reduced. Algorithms can adjust their behavior to the actual deployment using the distinction between boundary and inner nodes.

In a large number of scenarios, for example in the environmental monitoring of vineyards, forests, or warehouses, sensor readings differ heavily between the boundary and the center of the network. Therefore, the sensor readings from the boundary nodes might influence the aggregation results considerably because of capturing events occurring outside of the monitored region of interest. A finer differentiation of nodes based on their distance to the boundary makes it possible to detect such an influence.

Whether a node is an inner or a boundary node might be crucial in object tracking scenarios. For example, when tracking events entering and leaving a region, boundary nodes might be involved in more complex sensing tasks whereas inner nodes might spend more energy on performing routing tasks. This lets boundary nodes play a special role that cannot be assigned prior to deployment. In addition, grouping nodes by nesting levels allows the definition of further security perimeters with different alert degrees.

Accurately defining the boundary of a wireless network is a challenge that should not be underestimated. The authors of [8] provide an implicit definition of a network boundary in terms of nodes being close to the boundary of the continuous domain. The authors of [9, 13] define the boundary with the help of shortest cycles. In this paper we discuss the problem of defining the boundary, even with known node positions, and its relation to two intuitive properties: uniqueness and continuity. We show that the definitions in prior work are incomplete. Then we generalize the definition of boundary for the case where no location information is available.

The approach presented in this paper provides a close approximation of the generalized boundary without producing false negatives. Additionally, this boundary recognition approach is the first that is able to considerably relax the assumptions on node density and provides a good solution for both sparse and dense networks. Moreover, our result is a *generalization* of the approach described in [9]. Our approach is based on the recognition of inner nodes of the

978-0-7695-3157-1/08 $25.00 © 2008 IEEE
DOI 10.1109/IPSN.2008.11

network and considers all other nodes to be part of the outer boundary or the boundary of a hole. We introduce geometric constructions, called patterns, that guarantee that a particular node lies inside of the pattern for all deployments of the sensor network with a given connectivity graph. Our patterns are generic, simple, parameterized and no global knowledge of the network connectivity graph is required to recognize them. These properties make our approach scalable and applicable to sparse networks. Moreover, many of the patterns allow us to additionally assign a guaranteed minimum distance from a given node to the boundary. Nodes with the same guaranteed distance from the boundary form a nesting level. Although our patterns do not cover all nodes that lie inside of the network in all cases, simulation results show that the patterns are powerful enough to detect almost all inner nodes of the network and, therefore, provide a good approximation of the network boundaries. We provide pattern rules that considerably simplify the recognition of patterns and allow the algorithm to run in polynomial time.

The remainder of this paper is structured as follows. First, we discuss related work in Section 2. In Section 3, we describe the problems related to the definition of the network boundary. We then present in Section 4 geometric patterns that are the foundation of our approach. The algorithmic aspects and the time, space and message complexities are discussed in Section 5 followed by simulation results in Section 6. Our conclusions complete this paper.

2. Related Work

There is a strong need to extract spatial information of deployed sensor and ad-hoc networks without node coordinates. Even if distance information between the nodes is available, the problem of accurate node localization is NP-hard [1]. However, a number of approaches provide reasonable approximations of different topological characteristics of the network such as the outer boundary of the network and boundaries of holes [7–9, 13], isolines or contours [8] and medial axis lines [4, 13] or streets [9] that express topological levels based on the number of hops to the boundary.

Related work from the area of boundary recognition in sensor networks without location information can be classified into three groups based on the respective assumptions on node distribution, node density and the communication model. The approaches in the first group rely on a certain node distribution of the sensor nodes in the non-hole regions. For example, the approach described in [6] requires a uniform distribution of sensor nodes.

In the second group, a number of approaches [7, 8, 13] present solutions for boundary recognition based on the assumption that the length of the shortest path between two nodes provides a reasonable approximation of the geometric distance between the nodes. However, this assumption requires a rather high average node degree in the network (in the range of 25) for the approaches to perform reasonably well [7, 8]. The required node degree can be reduced to 10 if the topology conforms to a more regular node distribution like a grid or a perturbed grid [8]. Under the unit disk graph assumption, sufficient node density and further assumptions on hole size and hole placement the algorithm marks the nodes close to the boundary with certain guarantees [8]. In [7] the authors additionally discuss that the success rate of their method decreases with the decrease of d when the network topology follows the d-quasi unit disk graph model. Another approach in this group [13] also assumes that the distance among nodes can be approximated reasonably well based on the shortest path length and requires the lowest density of all approaches of this group with a node degree of 10 to 16. Such node densities are realistic for dense deployments [2].

The last group of approaches does not constrain the node distribution or make assumptions regarding node density but sets constraints on the radio model of the sensor nodes. The unit disk graph model is a weak approximation of the properties of the wireless radio. Therefore, the more general d-quasi unit disk graph model [12] is preferable. The approach presented in [9] is the only work so far that provides a solution for the problem of boundary recognition based on the single assumption that the input network follows a d-quasi unit disk graph for a given $d \geq \frac{\sqrt{2}}{2}$. The algorithm searches for several types of patterns, so called "flowers", that are further extended and merged in the augmenting phase of the algorithm to form a boundary of the network. However, the presented flowers are extremely complex and in random topologies they only exist with a high probability if the average node degree is very high (20–30) [5,9]. Moreover, a sensor node requires knowledge of its 8-hop neighborhood to be able to start searching for a flower. Evaluation results showed that this algorithm did not find a flower for network topologies with an average node degree smaller or equal to 10 [13].

The approach presented in this paper belongs to the last group and introduces the concept of patterns that are generic, simple and parameterizable by limiting the amount of neighborhood knowledge used. Therefore, even sparse networks with an average node degree of 4 include many simple patterns. The simplicity and generality of patterns considerably reduce the requirements on the node density of the network and the message complexity of pattern recognition. Flowers [9] form a tiny subset of our patterns. An additional feature of our pattern-based inner node recognition is that the majority of inner nodes are able to calculate a *guaranteed geometric distance* to the boundary of the network. We define a *nesting level* as a group of nodes with the same guaranteed geometric distance to the bound-

ary which is the main difference to hop-based isolines or contours. Moreover, the cluster of nodes with the highest nesting level is located in the *geometric center* of the region, which is a stronger topological characteristic than the hop-based medial axis lines or streets.

3. The Boundary of a Sensor Network

A sensor network is usually modeled as a graph $G(V, E)$. The boundary of a sensor network is a complex spatial property even when a straight-line embedding of this graph into the two dimensional space is known. The problem of defining the boundary lies in the intuitive properties it should possess: uniqueness and continuity. Previous definitions of the term boundary of a network found in the literature fail to preserve these properties [8,9,13]. This section discusses the problem of defining the boundary in sensor networks both for the case when position information is available and for the case when it is not.

Let us first consider the case when a straight-line embedding $p : V \rightarrow \mathbb{R}^2$ of the network is given. We say that p is a *d-quasi unit disk embedding* (d-QUDE) of G for a parameter $d \leq 1$ if both $uv \in E \Longrightarrow |p(u) - p(v)| \leq 1$ and $|p(u) - p(v)| \leq d \Longrightarrow uv \in E$ hold. G itself is called a *d-quasi unit disk graph* (d-QUDG) if such an embedding exists. A 1-QUDG is called a *unit disk graph* (UDG) and the corresponding embedding is a *unit disk embedding* (UDE). A *valid embedding* of the network is a d-QUDE for some fixed value of d.

According to the Jordan curve theorem, a simple closed curve divides the plane into two components – the outside and the inside. We call the outside the *infinite face* F^p_{inf} and the inside a *finite face* F^p. A sensor network can have several boundaries in an embedding: the outer boundary (border to the infinite face F^p_{inf}) and boundaries of holes (borders to finite faces $F^p \in F^p_{fin}$). These faces represent faces of interest defining uncovered regions in the network deployment. The elements of F^p_{fin} can be defined based on different spatial characteristics of a hole, e.g., the minimum size of its area, the length of its perimeter or the area of its convex hull [9].

The *perimeter* P_{F^p} of a face of interest F^p is defined as the set of segments of straight-line embeddings of communication links (edges) that belong to the face F^p. The perimeter P_{F^p} is both unique and continuous for a given embedding. Note that P_{F^p} does not have to include any end-point of an edge (node) as illustrated by the example in Fig. 1a). However, this purely edge-based perimeter is hardly useful in sensor networks. Instead, there is a need to define a *geometric boundary* B_{F^p} that approximates P_{F^p} based on nodes.

We define the geometric boundary B_{F^p} as the set of nodes that belong to the perimeter P_{F^p} (if such nodes exist). Obviously, the geometric boundary defined this way is unique but not continuous in the sense that B_{F^p} does not necessarily form a connected cycle (see Fig. 1b)). For this reason, a number of other approaches define the boundary as the (shortest) cycle that follows the perimeter of F^p [9, 13]. However, this definition does not possess the uniqueness property of the boundary. As an illustrating example, both the nodes 1 and 3 in Fig. 1b) might belong to the shortest cycle which contradicts the statement in [9]: "There is always one cycle for the outer perimeter. If the region has holes, there is an additional cycle for each of them." Moreover, such a boundary can even include nodes that are guaranteed to be inner nodes of the network for any valid embedding. We show an example in Fig. 1c) and d) for which it can be shown that nodes 1 and 2 lie inside of the network for any UDE. We highlight the subgraph that guarantees for node 1 to lie inside in c), and the subgraph that guarantees for node 2 to lie inside in d). The gray boundary nodes in the example break the continuity of the boundary without using either node 1 or 2.

The authors of [8] state that the goal of their algorithm is to mark a node near every point of the boundary of the continuous domain and that all nodes that are marked are near this boundary. This implicit definition of the network boundary does neither preserve the uniqueness nor the continuity property.

Recognizing a boundary without position information given a network connectivity graph needs consideration of all valid straight-line embeddings of this graph, which leads to the following problems: First, the problem of finding valid embeddings of a graph is NP-hard even when a UDG is used to model the network [10]. Second, the notion of holes in the deployment needs revision. Consider a straight-line embedding p of the network with a hole F^p of a certain size $size(F^p) \geq \min_C$ (e.g., defined by the perimeter or the area). It is generally impossible to construct a bijection between holes in different embeddings. Therefore, a hole is an *embedding-specific* geometric property.

In the literature, a hole F^p is approximated by the shortest cycle which contains the perimeter of the hole P_{F^p} [9, 13]. However, this approximation is unstable as well when looking at different valid embeddings. Consider a unit disk embedding p_1 of the network as shown in Fig. 1e). The embedded shortest cycles $p_1(C_0)$ and $p_1(C_1)$ contain the holes $F_0^{p_1}$ and $F_1^{p_1}$ respectively. Let the length of these cycles be $|C_0|$ and $|C_1|$ and assume that $|C_1| < |C_0|$. If we are interested in recognizing holes with a size of at least min_C with $|C_1| < min_C < |C_0|$, then faces $F_0^{p_1}$ and $F_{inf}^{p_1}$ must be detected by the boundary recognition algorithm. In Fig. 1f) we show a different unit disk embedding p_2 of the same network. Here, two faces are contained in $p_2(C_1)$, which is the shortest cycle containing them. Consequently, in p_2 both faces are too small to qualify as holes as their

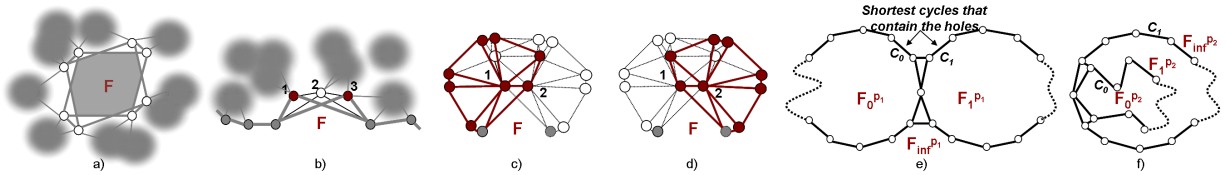


Figure 1: Problems related to boundary definitions

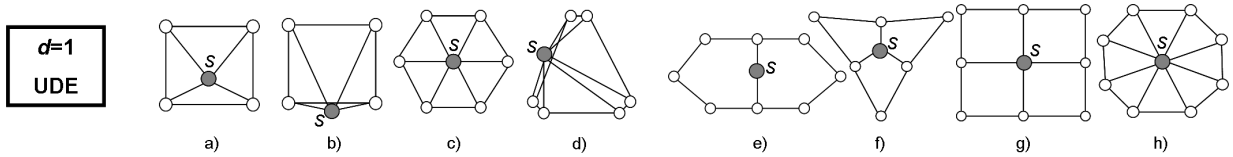


Figure 2: a-d) Insufficient constructions for UDG; e-h) Patterns for UDG

shortest cycles are shorter than min_C. Note that it is possible that a shorter cycle contains a longer one even in a UDE (see Table. 1). For the same reason, it is generally impossible to distinguish different hole boundaries without additional location information (e.g., the assumption that the shortest path approximates distance).

We now introduce our definition of boundary that addresses the problems described above. If there exists a valid embedding p and a face F^p such that the shortest cycle in this embedding which contains P_{F^p} has a minimum length min_C, then F^p is an element of the *generalized set of holes* F_{fin}. We define the set of infinite faces F_{inf} that includes all infinite faces F^p_{inf} for all valid embeddings p. We define the *generalized boundary* of the network as the set of nodes such that for every node in this set there is a valid embedding p in which this node belongs to the geometric boundary of a face $F^p \in F_{fin} \cup F_{inf}$. The generalized boundary is unique unless the connectivity graph changes. Moreover, it is minimal as it only contains nodes that belong to the outer boundary or the perimeter of a hole in at least one valid embedding. However, the generalized boundary is not continuous as can be seen in Fig. 1c) and d) in which the highlighted nodes 1 and 2 never belong to the generalized boundary as explained later (see Fig. 2h)).

Any reasonable boundary acquired without location information includes the nodes of the generalized boundary. In this paper, we use the property of the generalized boundary that its complement consists of all nodes that do not belong to the outer boundary or the boundary of a hole for any valid embedding. Our approach approximates the generalized boundary of the network by recognizing nodes that always lie inside of the network for any valid embedding and assumes that all other nodes belong to the generalized boundary.

4. Patterns

We introduce the concept of *patterns* – subgraphs that guarantee for any valid d-QUDG embedding that a certain node lies inside. This inner node is the *seed* of the pattern. Let us now consider Fig. 2a) and c). At first sight both subgraphs seem to fulfill these requirements but as shown by the counter-examples in Fig. 2b) and d), it is not guaranteed that the seed S lies inside of the construction for *all* valid UDEs. We show some real patterns in Fig. 2e-h). Fig. 2h) is the smallest pattern for a UDG we were able to find: node S only needs the knowledge of the communication links between its direct neighbors to detect this pattern. Due to space limitations, we do not present individual proofs for these patterns. Instead, we provide our generic pattern rules for the unit disk graph model of a network and extend these rules later for the more general d-QUDG model, which better captures the properties of wireless links [12]. We prove that all constructions generated by these rules, which include Fig. 2e,g), are indeed patterns and, therefore, guarantee for the seed to lie inside for *any* valid embedding. Our approach works for all d-QUDGs with $d \geq \frac{\sqrt{2}}{2}$. This lower bound for d is fundamental for the mathematical concepts of our approach and results from Lemma 4.1. This lemma has been proven for UDE in [3] and improved for d-QUDE with $d \geq \frac{\sqrt{2}}{2}$ by [9].

Lemma 4.1. *Let x, y, w, v be four different nodes in V, where $xy \in E$ and $wv \in E$. Assume the straight-line d-QUDE (for $d \geq \frac{\sqrt{2}}{2}$) of xy and wv intersect. Then at least one of the edges in $F = \{xv, vy, yw, wx\}$ is also in E.*

4.1. Terminology

$D(V_D, E_D)$ is defined to be a *vertex-induced subgraph* of $G(V, E)$, if $V_D \subseteq V$, $E_D \subseteq E$ and the following condition holds: $\nexists v_i, v_j \in V_D | v_i v_j \in E$ and $v_i v_j \notin E_D$.

$N_k(D)$, the *k-hop neighborhood* of the vertex-induced subgraph $D \subseteq G$, is the vertex-induced subgraph of G that includes all nodes reachable from at least one node in D within a maximum of k hops.

We model a sensor network using a d-QUDG as defined in Section 3 as it can model radio irregularities to some ex-

tent. The smaller the value of d, the more general and realistic is the model.

Let C be a vertex-induced subgraph of G and $V_C = \{v_0, v_1, \ldots, v_{k-1}\} \subset V$ be a sequence of $k > 3$ distinct vertices such that $v_i v_{(i+1) \bmod k} \in E_C, \forall i = 0 \,..\, k-1$ and no other edge exists between any two of these vertices. We refer to C as a *chordless cycle* of length $|V_C| = k$. A d-QUDE of a chordless cycle C is a polygon which decomposes a plane into the infinite face and at least one finite face. Every point in the infinite face is defined to lie *outside* of this polygon. There exist multiple finite faces if the embedded chordless cycle is self-intersecting.

We use the following properties to check whether a connected subgraph can be placed inside of a chordless cycle for a d-QUDE. Consider a chordless cycle $C \subset G$ and a connected vertex-induced subgraph $D \subset G \setminus N_1(C)$. A *maximum independent set* $I_C(D) = \{v_i\} \subset V_D$ with respect to C is a maximum subset of V_D such that: $\forall v_i, v_j \in I_C(D)$, $v_i v_j \notin E$. The elements of the independent set are called *independent nodes*. Since the distance between each pair of independent nodes is at least d (in any d-QUDE), the independent set requires a certain *minimum* area on the plane to place the subgraph D. The embedding of a chordless cycle on the plane results in a polygon with a limited *maximum* area that depends on the length of the chordless cycle. As defined in [9], the number $fit_d(k)$ is the maximum number of independent nodes that can be placed inside of a chordless cycle C of length $|V_C| = k$ for a d-QUDE. The *independent set property* (ISP) for a d-QUDE with $d \geq \frac{\sqrt{2}}{2}$ holds for a chordless cycle C and a connected vertex-induced subgraph D if $|I_C(D)| > fit_d(|V_C|)$. This means that there is not enough space in the chordless cycle to place all independent nodes of D inside of it. Additionally, Lemma 4.1 guarantees that the embedding of D lies *completely* outside of the embedding of C if the ISP holds.

4.2. Patterns in UDG

Here we propose our generic pattern rules for the UDG model. We present a formal definition and then explain the individual conditions including the terms *extended independent set property* and *critical intersection*.

We define a *weak pattern* $P(S,1) = C_0 \cup \cdots \cup C_{n-1}$, $2 \leq n \leq 5$ as a vertex-induced subgraph of G composed of chordless cycles C_i such that $\forall i, j = 0 \,..\, n-1$, $i \neq j$ the following conditions hold:

1. $S \in V_{C_i}$
2. $|V_{C_i \cap C_j \cap N_1(S)}| = \begin{cases} 1, & j \neq (i \pm 1) \bmod n \\ 2, & j = (i \pm 1) \bmod n \wedge n > 2 \\ 3, & j = (i \pm 1) \bmod n \wedge n = 2 \end{cases}$
3. For $C_i, (P(S,1) \setminus N_1(C_i))$ the *extended independent set property* holds
4. For $C_i, \bigcup_{j \neq i} C_j$ there exists no *critical intersection*

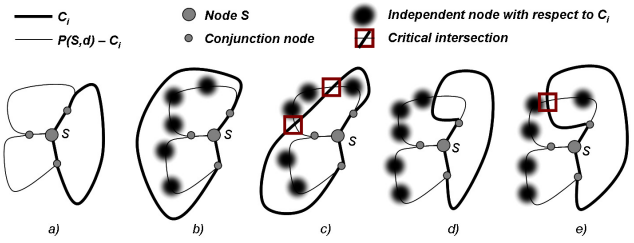


Figure 3: Combinations of three chordless cycles

To be able to reason about patterns, we have to define a few more terms, which are also illustrated in the example pattern in Fig. 4. We call $V_{P(S,1)} \cap V_{N_1(S)}$ the set of *anchors*. The number of anchors is equal to the number of cycles the pattern comprises, which we call the *pattern cardinality*. If a pattern consists of at least three cycles, then each pair $C_i, C_{(i+1) \bmod n}$ shares one anchor. We call this anchor the *common anchor* CA_i. If the pattern is composed of exactly two chordless cycles, both anchors are shared by both cycles and we deterministically define (e.g., by node ID) CA_0 and CA_1. Now consider the vertex-induced subgraph $N_1(C_i) \cap N_1(C_{(i+1) \bmod n}) \cap (C_i \cup C_{(i+1) \bmod n}) \setminus S$. This subgraph is generally not connected. We define the *conjunction* J_i between two cycles $C_i, C_{(i+1) \bmod n}$ to be the connected component that includes CA_i. Finally, we define the *outer cycle* of a pattern as the connected vertex-induced subgraph with the set of edges $E_{P(S,1)} \setminus E_{\bigcup J_i}$.

To motivate the need for the last two pattern conditions, we show different possible embeddings of the bold chordless cycle C_i in Fig. 3 for a combination of three cycles. There are three possible relations of one cycle to the others: it may contain them (b), it may intersect them (c,d,e) and it may lie on a different side of S (a,d,e). In cases b) and c) C_i is called *reflected*. If S lies outside of the construction, then either one cycle contains all others or there is a *critical intersection* of at least two chordless cycles. Informally, a critical intersection occurs when the independent set of a vertex-induced subgraph is partitioned by the intersection with a chordless cycle. In Fig. 3 the example b) is rejected (is not sufficient to be a pattern) because one cycle contains all others and examples c) and e) are rejected because of a critical intersection. The intersection in example d) is not critical and, therefore, both a) and d) are valid patterns.

We use the *extended independent set property* (eISP) to check if a cycle C_i can contain all other chordless cycles in a d-QUDE with $d \geq \frac{\sqrt{2}}{2}$. We examine the vertex-induced subgraph $\bar{C}_i = N_h(S) \setminus N_1(C_i)$, which generally consists of multiple connected components. We compute the size of a maximum independent set for each connected component that contains at least one node of $P(S,d) \setminus N_1(C_i)$. If the sum of these sizes is greater than $fit_d(|V_{C_i}|)$, then C_i cannot contain all other chordless cycles. Note that using the standard ISP would only allow us to calculate a maximum independent set for $P(S,d) \setminus N_1(C_i)$ (without considering

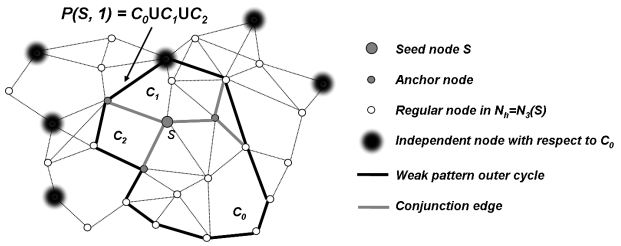


Figure 4: Extended independent set property

the whole neighborhood $N_h(S)$, the so-called *extension*). The eISP is especially important for complex patterns with long cycles since the number of independent nodes that can fit in a chordless cycle grows faster than the cycle length (see Table 1).

We illustrate how the eISP works in Fig. 4 for a weak pattern $P(S,1) = C_0 \cup C_1 \cup C_2$. The ISP does not hold for $C_0, P(S,1) \setminus N_1(C_0)$ because $P(S,1) \setminus N_1(C_0)$ contains only one independent node with respect to C_0 and there is enough space in a cycle of length $|V_{C_0}| = 9$ (see $\mathrm{fit}_1(9)$ in Table. 1) to place this node inside for some UDE. However, the eISP holds since a connected component in $\bar{C}_0$ that includes this independent node contains a maximum independent set of size $6 > \mathrm{fit}_1(9) = 2$.

If S lies outside of the construction but the eISP holds, then the embeddings of at least two cycles must intersect. Since the eISP holds, both cycles must contain at least one independent node with respect to each other. We show examples of such constructions consisting of two cycles in Fig. 5. We distinguish between *vertex-based* (a) and *edge-based* (b) intersections. In order to detect a *critical intersection*, we require that either the cycles share at least one node (a) or that at least one of the dotted edges (1–4) in Fig. 5b) exists. Lemma 4.1 shows that at least one edge exists for a d-QUDE with $d \geq \frac{\sqrt{2}}{2}$. We show that at least two edges exist for a UDE with the following Lemma 4.2.

Lemma 4.2. *Let x, y, w, v be four different nodes in V, where $xy \in E$, $wv \in E$ and $xw \in E$. Assume the straight-line UDE of xy and wv intersect. Then at least one of the edges in $F = \{xv, yw\}$ is also in E.*

Proof. Assume $p(x) \neq p(y) \neq p(w) \neq p(v)$; otherwise the proof of the lemma is trivial. Let $a = |p(x) - p(y)| \leq 1$. Consider two circles of common radius d with their centers at $p(x)$ and $p(y)$ respectively. The distance between the xy segment and the intersection points of these circles is $\frac{h}{2} = \sqrt{d^2 - \frac{1}{4}a^2}$. As F and E are disjoint, $p(w)$ must lie outside of the circle with the center

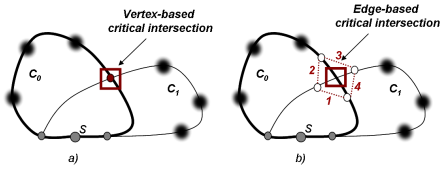


Figure 5: Types of critical intersections

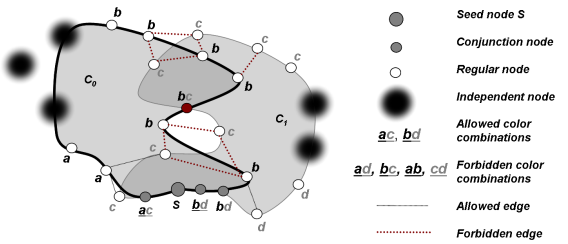


Figure 6: Coloring test

at $p(y)$ and $p(v)$ has to lie outside of the circle with the center at $p(x)$. Because of the intersecting edge embedding, for $d \geq 1$, $|p(w) - p(v)| > \sqrt{(\frac{h}{2})^2 + (d - \frac{a}{2})^2} = \sqrt{2d^2 - ad} \geq 1$, which contradicts that $wv \in E$. □

We detect critical intersections by coloring the chordless cycles as shown in Fig. 6 (colors a–d). We have to execute the following procedure for every chordless cycle C_i and $\bigcup_{j \neq i} C_j$. We color C_i starting at its two anchors with two different colors. Additionally, we switch to a new color each time we encounter an independent node with respect to $\bigcup_{j \neq i} C_j$. We color the other cycles of $P(S,d)$ the same way starting at their anchor nodes and change the color after each independent node with respect to C_i. S is then the only uncolored node. The vertices of the conjunctions between $C_i, C_{(i \pm 1) \bmod n}$ define the *allowed* color combinations. Every node and every edge connecting nodes of C_i and $\bigcup_{j \neq i} C_j$ is inspected. If the color combination is not allowed, then we speak of a critical intersection.

Lemma 4.3. *If node $S \in V_G$ is the seed of a weak pattern $P(S,1) \subseteq G$, then S is an inner node for any UDE of G.*

Proof. Assume a UDE such that S lies outside of $P(S,1)$. Then either at least one chordless cycle of $P(S,1)$ is reflected or at least one conjunction intersects the outer cycle. Assume C_i is a reflected cycle in $P(S,1)$. Let us color $C_i, \bigcup_{j \neq i} C_j$. According to the eISP C_i cannot contain all other chordless cycles in $P(S,1)$. Therefore, C_i intersects $C_j \subset P(S,1) (j \neq i)$. However, as no coloring conflicts are found, the intersection between C_i and C_j must have a color combination allowed by the two conjunctions of C_i. Therefore, no independent node is located between node S and the nodes that belong to the intersection. So, all independent nodes of $P(S,1) \setminus C_i$ lie inside of C_i, which violates the third pattern condition.

From Lemma 4.2 follows that a conjunction J_i can only intersect one edge of the outer cycle which lies in $N_1(J_i)$ and connects $C_i \setminus J_i, C_{(i+1) \bmod n} \setminus J_i$. No two conjunctions can intersect the same edge and from the fourth pattern condition follows that no two conjunctions can intersect each other. If the seed lies outside of the pattern and no cycle is reflected, then there is one cycle that contains all others. □

4.3. Patterns for d-QUDG

After having described patterns for UDG in the last section, we now present our extended approach that supports

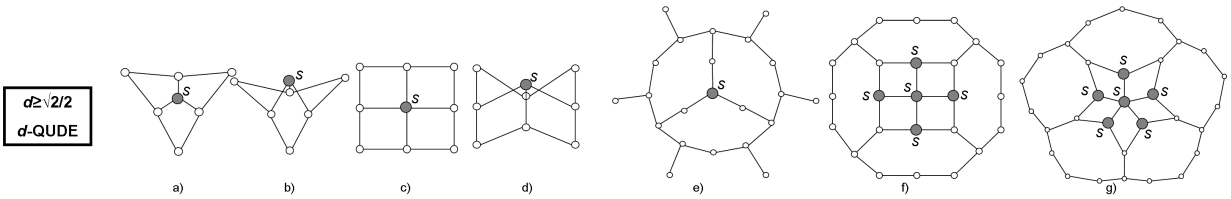


Figure 7: a-d) Insufficient constructions for d-QUDG, $d < 1$; e-g) Patterns for d-QUDG (f,g) from [9])

d-QUDG for $d \geq \frac{\sqrt{2}}{2}$. The simple patterns for UDG presented in Fig. 2 are not sufficient for d-QUDG with $d < 1$. We show counter examples in Fig. 7b) and d) for the UDG-only patterns from Fig. 7a) and c). We also show patterns for d-QUDG with $d \geq \frac{\sqrt{2}}{2}$ in Fig. 7e-g). The examples f) and g) are taken from [9] and illustrate an interesting difference to our approach. In [9] these very complex patterns are used to detect a group of guaranteed inner nodes (highlighted in the figure), whereas our approach is able to detect each of these inner nodes individually. By using simpler as well as more complex patterns our approach is more general and more powerful. All of these patterns are covered by the following extended pattern definition.

We define a *strong pattern* $P^*(S,d) = C_0 \cup \cdots \cup C_{n-1}$, $2 \leq n \leq 8$ as a vertex-induced subgraph of G composed of chordless cycles C_i such that $\forall i, j = 0 \,..\, n-1$:

1-4) $P^*(S,d)$ fulfills the conditions of the weak pattern definition for the given value of d

5) One of the following conditions holds for each conjunction J_i between the pair of chordless cycles $C_i, C_{(i+1)\bmod n}$:
   a) $|V_{J_i \setminus S}| \geq 2$
   b) $|V_{J_i \setminus S}| = 1$ and an edge exists between $N_1(J_i \setminus S) \cap C_i$ and $N_1(J_i \setminus S) \cap C_{(i+1)\bmod n}$
   c) $|V_{J_i \setminus S}| = 1$ and a weak pattern exists for $V_{J_i \setminus S}$ that includes $C_i, C_{(i+1)\bmod n}$

The reason why a weak pattern $P(S,d)$ does not work in a d-QUDE for $d < 1$ is that Lemma 4.1 does not preclude the possibility that a chordless cycle is self-intersecting. This is only guaranteed for UDE by Lemma 4.2.

Lemma 4.4. *If node $S \in V_G$ is the seed of a strong pattern $P^*(S,d) \subseteq G$, then S is an inner node for any d-QUDE of G with $d \geq \frac{\sqrt{2}}{2}$.*

Proof. The proof for strong patterns exactly follows the proof for weak patterns. There is only one additional point to show: Although conjunctions may intersect the outer cycle of a strong pattern $P^*(S,d)$, the seed S is still guaranteed to lie inside of it. Assume S lies outside of the construction. Let $C_i \subset P^*(S,d)$ be a chordless cycle. It follows from the fifth condition of a strong pattern that $|V_{C_i}| > 4$. The conjunction J_i between $C_i, C_{(i+1) \bmod n}$ can intersect edges of the outer cycle of the strong pattern in $C_i \cap N_2(J_i)$ and in $C_{(i+1) \bmod n} \cap N_2(J_i)$. However, these vertex-induced subgraphs do not overlap for any pair of conjunctions, which follows from the fifth condition of a strong pattern and Lemma 4.1. Since no independent node with respect to C_i is in $N_1(J_i)$, this intersection is not critical. Therefore, there is a chordless cycle that contains all others. This violates the third pattern condition and contradicts the assumption. □

4.4. Pattern Properties

There are several important properties of weak and strong patterns that can be used to derive further spatial information of a sensor network and to optimize the pattern recognition algorithm.

Distance guarantees: Both weak and strong patterns guarantee that the seed lies inside of the pattern. However, a strong pattern additionally ensures that the seed has no direct connection (edge) to the outer cycle of the pattern. Therefore, the seed is at least $\sqrt{d^2 - \frac{1}{4}}$ away from every edge of this outer cycle. Thus, strong patterns can additionally provide *distance guarantees* for their seeds.

Nesting levels: We show in Section 5.2 how to accumulate guaranteed distances in order to designate nesting levels for sensor nodes.

Inclusion property: If $P(S,d)$ then $P(S,q), \forall q \geq d$. This holds as long as there exists a valid q-QUDE for $P(S,q)$. This property is important for applying the pattern concept in real-world deployments where the exact value of d is not known.

Maximum pattern cardinality: The size of a maximum independent set of $N_1(S)$ for a node S is limited by $\left\lfloor \frac{\pi}{\arcsin \frac{d}{2}} \right\rfloor$ (5 for a UDG, 8 for the d-QUDG model with $d = \frac{\sqrt{2}}{2}$). This limits the maximum number of chordless cycles in a pattern and restricts the search depth.

Discreteness: In UDE both weak and strong patterns can be used. Weak patterns only guarantee that the seed lies inside of the construction whereas strong patterns additionally provide a guaranteed minimum distance of $\frac{\sqrt{3}}{2}$ from the outer cycle of the pattern. However, if we consider d-QUDE with $d \in [\frac{\sqrt{2}}{2}, 1)$, only strong patterns work. This *discrete* behavior of patterns results from Lemma 4.1 and Lemma 4.2: According to Lemma 4.1, there are only two possible relations between two edges $xy, vw \in E$ for a d-QUDG with $d \in [\frac{\sqrt{2}}{2}, 1)$. If $xv, xw, yv, yw \notin E$, then x and y are at least $\sqrt{d^2 - \frac{1}{4}}$ away from vw. If *at least one* of the edges xv, xw, yv, yw exists, then there might be an embedding where xy, vw intersect. There is one more possibility in UDE: if *exactly one* of the edges xv, xw, yv, yw exists, xy, vw cannot intersect according to Lemma 4.2, but x or y can be arbitrarily close to vw.

Soundness: Our approach is *sound* as the patterns covered guarantee that the corresponding seed nodes lie inside of the network for any d-QUDE with $d \geq \frac{\sqrt{2}}{2}$.

Incompleteness: Although being able to describe a family of simple as well as arbitrarily complex patterns, this concept does not cover the whole set of patterns (e.g., Fig. 2h)). Therefore, our approach is *incomplete*. However, we show in the evaluation section that our pattern rules are powerful and general enough to recognize a large number of guaranteed inner nodes for dense as well as for sparse topologies.

5. Boundary Recognition Algorithm

Having laid out the mathematical foundations and necessary conditions for patterns, we now present our algorithm for boundary recognition and its complexity analysis.

The goal of the *boundary recognition algorithm* is to find weak and strong patterns in the network. It also assigns a level to every node that indicates its distance to the boundary. If a node is guaranteed to lie inside of the network, level 1 is assigned (UDG only). If the node is additionally guaranteed to be at least $dist = \sqrt{d^2 - \frac{1}{4}}$ away from the boundary, then level 2 is assigned. All other nodes are assumed to belong to a boundary and receive level 0.

Our algorithm is parameterized with d, which specifies the d-QUDG model. The range of d is limited to $\frac{\sqrt{2}}{2} \leq d \leq 1$, which is a hard bound resulting from Lemma 4.1. The parameter h specifies the h-hop neighborhood used for finding patterns. We limit the maximum length of chordless cycles to $2h + 1$. This implies that the minimum chordless cycle length to define the generalized set of holes is $min_C = 2h + 2$. In order to reduce the time, space and message overhead for dense networks, h should be chosen based on the average node degree in the respective network. We provide guidelines for selecting reasonable values of h in the evaluation.

After the boundary recognition algorithm is finished, it is executed again using only those nodes that have been identified as seeds of strong patterns. We execute the algorithm repeatedly, each time reducing the examined graph and incrementing the levels, as long as strong patterns are found.

5.1. Boundary Recognition

The boundary recognition algorithm executes the following steps for each node S in order to find a weak (UDE only) or strong pattern by checking the pattern conditions.

1) Gather the h-hop connectivity graph N_h.
2) Find all chordless cycles of maximum length $2h+1$ in N_h that include S (condition 1).
3) Construct valid combinations of chordless cycles (condition 2) and perform:
   a) The extended independent set test (condition 3).
   b) The critical intersection test (condition 4).
   c) The strong pattern test (condition 5).

We now explain each step in detail and analyze their respective time, space and message complexities.

1) *Gather the h-hop connectivity graph.* Every node broadcasts a message containing its ID with a time to live (TTL) of h. When a node receives such a message from another node, it appends its ID, decrements the TTL and forwards the message. The message complexity of this step for one node is $O(h^2 n_{max})$ where n_{max} is the maximum node degree in N_h. Constructing the connectivity graph of N_h runs in $O(h^4 n_{max}^3)$ time and requires $O(h^2 n_{max}^2)$ space.

2) *Find chordless cycles.* In this step, we perform a depth first search starting at S to find all chordless cycles in the discovered connectivity graph $N_h(S)$. The maximum search depth is limited by the maximum length of a cycle $2h + 1$. This step runs in $O(n_{max}^{2h+1})$ time. We have to store the chordless cycles in memory to be able to construct later on the chordless cycle combinations needed to find patterns. However, it is difficult to establish a tight upper bound for the number of chordless cycles. We have to estimate it by the number of paths of length $\leq 2h + 1$ that start at S. This number is clearly greater than the number of chordless cycles. Therefore, the definitely over-estimated space complexity is $O(n_{max}^{2h+1})$. Additionally, the number of chordless cycles determines how many combinations of cycles are possible and how often the tests for the remaining pattern conditions have to be performed. Therefore, the number of chordless cycles is the determining factor for the time and space complexity of the complete algorithm. To reduce the number used in later tests, we define a *similarity metric* for chordless cycles and avoid storing similar ones. If one cycle cannot be used to construct a pattern, it is unlikely that a very similar cycle can be used successfully. To take advantage of that, we developed *μ-filtering* for two chordless cycles C_i and C_j that both contain S. For a chordless cycle C, we define the *center* node $c_S(C)$ relative to S as the node in C that has the greatest hop distance to S. If the length of the cycle is even, we choose the center node deterministically (e.g., by node ID). Two cycles are considered similar, if they share the same anchors and the center node of one cycle is also part of the other. If two cycles are considered similar we only keep the longer one to increase the probability of constructing a strong pattern.

Each chordless cycle has two anchor nodes and exactly one center. Therefore, the number of chordless cycles after μ-filtering can be estimated with $n_{cyc} = O(h^2 n_{max} \mathrm{C}(n_{max}, 2)) = O(h^2 n_{max}^3)$ where $\mathrm{C}(n, k)$ is the binomial coefficient. Consequently, the space complexity is $O(h^3 n_{max}^3)$. The search for chordless cycles includ-

ing μ-filtering runs in $O(n_{max}^{2h+4}h^2)$. As shown in the evaluation section, μ-filtering does not degrade the quality of the result and reduces both time and space complexity of the complete algorithm significantly.

3) *Construct valid combinations of chordless cycles.* In this step, we check the second pattern condition by constructing the valid combinations of chordless cycles using depth first search (DFS). We perform the remaining pattern tests as described in the next paragraphs. DFS runs in $O(n_{cyc}^{\mathrm{card}_{\max}})$ time and requires $O(n_{cyc})$ space where $\mathrm{card}_{\max}$ is the maximum pattern cardinality (5 for UDE, 8 for d-QUDE with $d = \frac{\sqrt{2}}{2}$). We show in the evaluation that by using reasonable parameter values it is possible to reduce the maximum number of chordless cycles to construct a pattern to 2 and 3 (instead of 5 and 8) for UDE and d-QUDE respectively without degrading the result.

3a) *Extended independent set test.* Since the problem of finding the maximum independent set is NP-hard, we use the following greedy algorithm to approximate this set. Consider a connected vertex-induced subgraph $D = N_h \setminus C_i$ for which the maximum independent set is computed. We search for a node v with a minimum node degree and choose this node as an element of the independent set. This step is repeated for the vertex-induced subgraph $D' = D \setminus N_1(v)$. Note that the independent set constructed this way is not necessarily maximal. However, since we use the eISP, this does not degrade the quality of the result. The time complexity of this step is $O(|V_D|^2\ log\ |V_D|) = O(h^2 n_{max}^2)$.

3b) *Critical intersection test.* As discussed above, we perform the critical intersection test by coloring the nodes of a potential pattern $P(S, d)$. The algorithm consists of the following steps for every chordless cycle C_i of $P(S, d)$:

1. Determine the *unrelated* nodes in C_i and in $P(S, d) \setminus C_i$ that have no edge to any node of the other set.
2. Color the nodes of C_i and $P(S, d) \setminus C_i$ as described in Section 4.2.
3. Determine the two color combinations allowed by the two conjunctions of C_i.
4. If vertex-based or edge-based illegal color combinations exist, then reject $P(S, d)$.

Determining the unrelated nodes in step 1 and coloring all nodes (step 2) both run in $O(h^2)$ time and require $O(h)$ additional space. The running time of checking for illegal color combinations (step 4) is $O(h^2)$ and no additional space is required.

3c) *Strong pattern test.* For checking whether the conjunctions fulfill the fifth strong pattern property, we only consider the first two possibilities of the fifth condition. This is done in $O(1)$.

We now present the overall complexities of the boundary recognition algorithm for a single node using μ-filtering and a limitation of the pattern cardinality to 3. These optimizations are also used in the evaluation. The resulting time complexity is $O(n_{max}^{2h+4}h^2)$, the space complexity is $O(n_{max}^3 h^3)$, and the message complexity is $O(n_{max}h^2)$, where h is the depth of the neighborhood N_h and n_{max} is the maximum node degree in N_h. These complexity values hold for $h \geq 4$.

5.2. Nesting Levels

After the boundary recognition algorithm is completed, we execute it again for all nodes of level 2. This corresponds to constructing a vertex-induced subgraph $G' \subset G$ composed of the nodes of level 2 and using G' as input to the pattern recognition algorithm. After the second run of the algorithm, the seeds of the weak patterns (UDE only) and of the strong patterns are assigned the levels 3 and 4 respectively. Consequently, nodes of level 4 are at least $2 \cdot dist$ away from the boundary. The boundary recognition algorithm is executed repeatedly as long as seeds of strong patterns (resulting in higher levels) are found. Since in each round the examined graph is smaller than in the previous round, the algorithm is guaranteed to terminate.

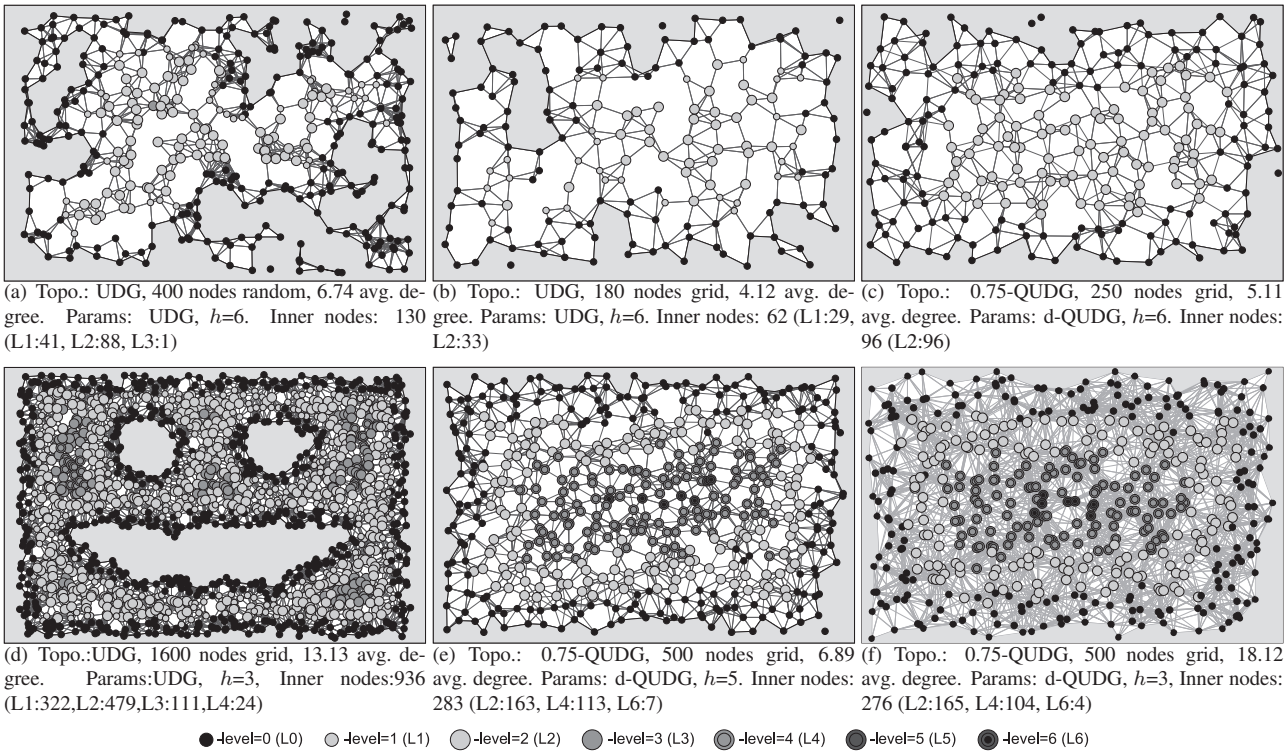
6. Evaluation

The primary goal of the evaluation is to show that our approach is general enough to support both sparse and dense deployments. We also investigate the empirical time and space costs of the different steps of our algorithm and conclude with guidelines on how to select the most appropriate parameter values.

We concentrate on two types of topologies: random and grid-based. We use *random* topologies (uniform distribution) to show that our approach does not rely on any assumptions concerning the regularity of node positions, which are required by some previous approaches. We use *grid-based* topologies to create more realistic scenarios: the nodes are placed randomly inside of circles arranged in a regular grid (the radius is equal to the grid spacing). This type of topology better matches real-world deployments where the goal usually is a more or less regular coverage of the sensing area.

The probability that a link exists between two nodes is calculated by $\frac{1-s}{1-d}$ for a distance s between two nodes with $d \leq s \leq 1$. While our approach supports asymmetric links as long as the connectivity graph is a valid d-QUDG, we only experiment with symmetric links in our topologies.

We vary the network wide average node degree between 2 and 20, which is reasonable considering the results discussed in [2] where values between 3 and 9 are suggested as reasonable for networks composed of 50 to 500 nodes.



(a) Topo.: UDG, 400 nodes random, 6.74 avg. degree. Params: UDG, h=6. Inner nodes: 130 (L1:41, L2:88, L3:1)

(b) Topo.: UDG, 180 nodes grid, 4.12 avg. degree. Params: UDG, h=6. Inner nodes: 62 (L1:29, L2:33)

(c) Topo.: 0.75-QUDG, 250 nodes grid, 5.11 avg. degree. Params: d-QUDG, h=6. Inner nodes: 96 (L2:96)

(d) Topo.:UDG, 1600 nodes grid, 13.13 avg. degree. Params:UDG, h=3, Inner nodes:936 (L1:322,L2:479,L3:111,L4:24)

(e) Topo.: 0.75-QUDG, 500 nodes grid, 6.89 avg. degree. Params: d-QUDG, h=5. Inner nodes: 283 (L2:163, L4:113, L6:7)

(f) Topo.: 0.75-QUDG, 500 nodes grid, 18.12 avg. degree. Params: d-QUDG, h=3, Inner nodes: 276 (L2:165, L4:104, L6:4)

Figure 8: Qualitative evaluation

We use the values shown in Table 1 for the fit_d-function. Since determining these values corresponds to solving the packing problem, we approximate the values assuming a hexagonal and square packing of nodes for UDE and d-QUDE respectively. Hexagonal and square packings provide packing densities of $\frac{\pi}{2\sqrt{3}}$ and $\frac{\pi}{3\sqrt{2}}$. It is known that a hexagonal packing is the most dense packing for the infinite face. However, by applying both packings to estimate the maximum number of circles that can be located inside of a polygon, we cannot guarantee that the values in the table are not underestimated. Nevertheless, we have chosen the values liberally high because we consider the whole N_h neighborhood for the extended independent set test. Using the eISP instead of the ISP increases the number of detected patterns considerably.

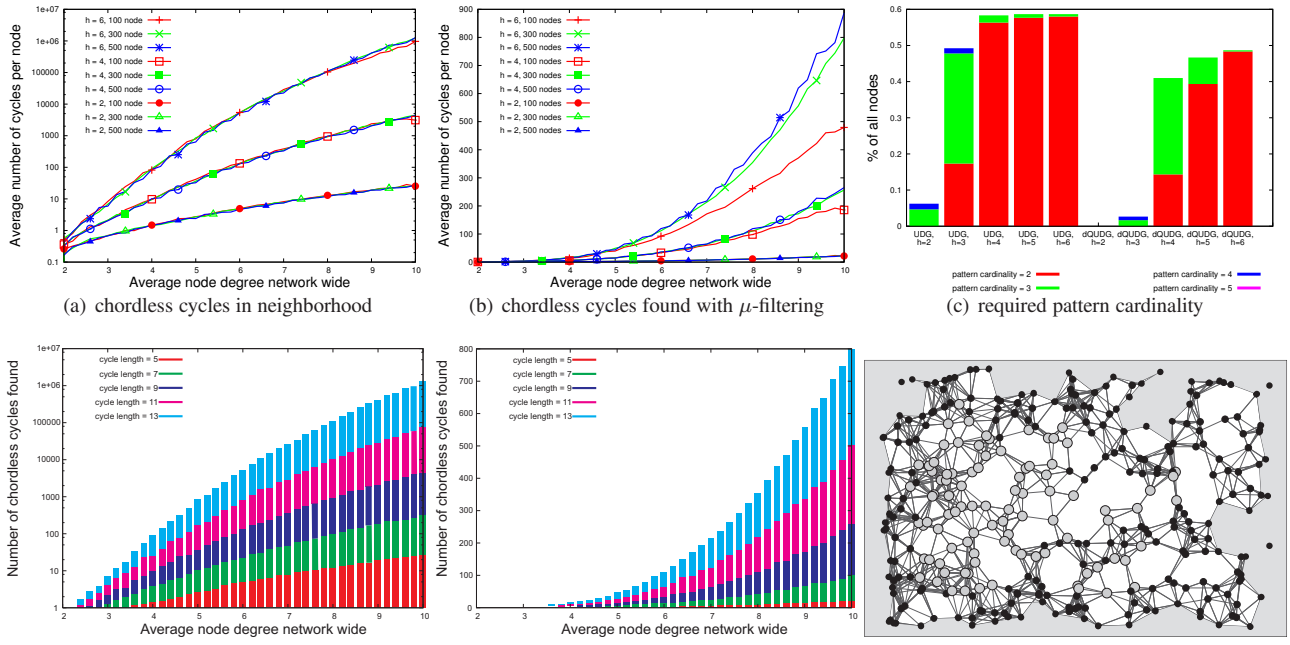
6.1. Qualitative Evaluation

Our algorithm approximates the generalized boundary of a network. Therefore, the nodes that belong to the geometric boundary of an embedding are included in the set of detected boundary nodes but cannot be distinguished from the other nodes in the set without location information. We are not aware of any way to reliably determine all nodes that belong to the generalized boundary. Therefore, the only reasonable way to evaluate the quality of our approach is to visually inspect the result. We examined a large number of results covering a wide range of parameters.

n	4	5	6	7	8	9	10	11	12	13
$fit_1(n)$	0	0	0	1	1	2	3	4	5	7
$fit_{\uparrow 1}(n)$	0	0	1	1	2	3	4	5	7	8
$fit_{\uparrow \sqrt{2}/2}(n)$	1	1	2	3	5	7	8	10	13	17

Table 1: ISP: values of fit_d for different d

In Fig. 8 we present several examples of network topologies (UDG and d-QUDG) processed by our algorithm with different node distributions and a wide range of different average node degrees that represent a wide spectrum of possible network deployments. Since the evaluation of previous approaches uses the UDG model, we first show several UDG deployments. Fig. 8a) is an example of a sparse random deployment of a UDG (average node degree 6.74) which requires the nodes to have information of their 6-hop neighborhood. A more regular grid-based UDG deployments shown in b) with an even lower average node degree of 4.12 also requires the knowledge of the 6-hop neighborhood. In Fig. 8d) we present a dense topology with artificially created holes. The high average node degree allows to achieve good results considering only the 3-hop neighborhood when looking for patterns. We continue the evaluation of our approach using the more realistic d-QUDG model with $d \geq \frac{\sqrt{2}}{2}$. Notice that only even levels are meaningful as nesting levels for the d-QUDG model. We show a sparse grid-based 0.75-QUDG network deployment with an average node degree of 5.11 in Fig. 8c). A slightly higher value



(a) chordless cycles in neighborhood (b) chordless cycles found with μ-filtering (c) required pattern cardinality

(d) chordless cycles in neighborhood (e) chordless cycles found with μ-filtering (f) Topo.: 0.75-UDG, 330 nodes rnd., 7.67 avg. degree. Params: 0.75-UDG, h=5, Inner nodes: 87 (L2:87)

Figure 9: Evaluation results and parameter selection guidelines

of the average node degree allows to decrease the amount of required neighborhood knowledge (Fig. 8e,f). Since no holes exist in the deployment in these example, the nodes with the highest nesting level of 6 approximate the geometric center of the network. We present a very dense topology in f) that allows us to use only the minimum possible neighborhood knowledge (3-hops) for searching strong patterns in d-QUDE. For UDE it is even possible to only search the 2-hop neighborhood if nesting levels are not required.

As described above, the approach with the lowest requirements on node density [13] only provides good results for node densities of at least 10–16 for UDG. Our evaluation shows that our approach works well in sparse networks even with an average node degree of only 4.

6.2. The Cost of Pattern Recognition

Fig. 9a, b) show the average number of chordless cycles found for a large number of topologies. Since boundary nodes find fewer chordless cycles than inner nodes, the ratio between boundary nodes and inner nodes determined by the network size directly influences the number of cycles. For that reason, we chose three different network sizes with 100, 300 and 500 nodes and constructed 20 grid-based topologies for each size. We varied the transmission range in each topology in order to obtain different average network wide node degrees and computed the average number of chordless cycles for different values of $h \in \{2, 4, 6\}$. In Fig. 9a) we show the average number of chordless cycles found by a node (note the logarithmic scale). The average number of chordless cycles after μ-filtering is presented in Fig. 9b) (linear scale). We evaluated the efficiency of μ-filtering by computing the ratio between the number of cycles left after μ-filtering and the total number of cycles. Our analysis shows that the efficiency of μ-filtering increases with the average node degree and with higher values of h. Therefore, μ-filtering decreases the space and time requirements of the complete pattern recognition algorithm significantly.

The next step of the pattern recognition algorithm is to search for a valid pattern by combining the chordless cycles. This search stops as soon as a strong pattern is found. This prevents inner nodes from searching through all possible combinations. The number of chordless cycle combinations that form a pattern in UDE and d-QUDE is limited by the maximum pattern cardinality. However, it is generally cheaper to check for a potential pattern consisting of fewer cycles. More importantly, the number of combinations grows exponentially with the maximum considered pattern cardinality. It is interesting to evaluate the benefit of a greater pattern cardinality. For the network in Fig. 8d) we calculated the number of nodes for which a pattern exists that is composed of $n_{combi} + 1$ chordless cycles, while no pattern exists that involves only n_{combi} chordless cycles. In Fig. 9c) we show the percentage of nodes that recognized a pattern consisting of a certain minimum number of cycles

for values of h between 2 and 6. On the one hand, there is a certain value of $h = h'$, such that almost all inner nodes find a pattern and there is no benefit in further increasing h. On the other hand, for $h = h'$ there is a pattern consisting of only 2 (UDE) or 3 (d-QUDE) cycles for nearly all inner nodes. The saturation of the number of detected inner nodes with increasing values of h motivates the need for the following guidelines for parameter selection depending on the density of the sensor network.

6.3. Parameter Selection and Adaptation

The main parameter of the system is h which defines the depth of the neighborhood that has to be considered as well as the minimum size of a hole $2h + 2$. Obviously, h should be chosen based on the density of the network. We investigated the influence of the density on the average number of chordless cycles (per node) with a given length. We chose the grid-based topology (to reduce variance) shown in Fig. 8d) and varied the transmission range to obtain different average node degrees. In Fig. 9d) and e) the average number of cycles with a given length is plotted against the average node degree. Values without (logarithmic scale) and with μ-filtering (linear scale) are shown.

Since the variance of the average node degree is relatively small in grid-based topologies, it is possible to use the same value of h for the whole network. In random topologies there usually are areas with a very high average node degree and areas with a very low average node degree in the same network. We show an example of such a random network topology in Fig. 9f). The same value of h was selected for the complete network. Our algorithm finds nearly all inner nodes in the left part where the value of h is sufficient for the average node degree of this area. However, the sparser area in the right part of the figure requires a greater value of h. This motivates the need for adapting the value of h to the local density of the region, which we plan to investigate as part of future work.

7. Conclusions

In this paper we have presented a solution for boundary recognition and the assignment of nesting levels in sensor networks without location information using only local knowledge. Our approach is based on the d-quasi unit disk graph model for radio propagation and supports any $d \geq \frac{\sqrt{2}}{2}$. We have presented a graph-oriented definition of boundary for this model which addresses issues found in prior work. We are able to guarantee that all nodes recognized as inner nodes lie inside of the network for any d-quasi unit disk embedding for a given d. We also provide additional discrete distance guarantees to the boundary called nesting levels. Our solution based on generic pattern rules is mathematically proven. We showed that our algorithm including optimizations runs in polynomial time and works equally well on dense and sparse topologies.

Note that $\frac{\sqrt{2}}{2}$ is a hard limit for d and cannot be reduced by any incremental work on our solution. Although our generic pattern rules do not cover the complete set of patterns, the mathematical properties and the quality of the results especially for sparse networks illustrate the power and wide applicability of our approach.

References

[1] J. Aspnes, D. Goldenberg, and Y. Yang. On the computational complexity of sensor network localization. *Proc. of the 1st Int. Workshop on Algorithmic Aspects of Wireless Sensor Networks*, 2004.

[2] D. M. Blough, M. Leoncini, G. Resta, and P. Santi. The k-neigh protocol for symmetric topology control in ad hoc networks. In *Proc. of the 4th Int. Symp. on Mobile Ad Hoc Networking & Computing*, 2003.

[3] H. Breu and D. G. Kirkpatrick. Unit disk graph recognition is NP-hard. *Computational Geometry: Theory and Applications*, 1998.

[4] J. Bruck, J. Gao, and A. A. Jiang. MAP: Medial axis based geometric routing in sensor networks. In *Proc. of the 11th Int. Conf. on Mobile Computing and Networking (MobiCom)*, 2005.

[5] S. P. Fekete and A. Kröller. Geometry-based reasoning for a large sensor network. In *Proc. of the 22nd Symp. on Computational Geometry*, 2006.

[6] S. P. Fekete, A. Kröller, D. Pfisterer, S. Fischer, and C. Buschmann. Neighborhood-based topology recognition in sensor networks. In *Proc. of the 1st Int. Workshop on Algorithmic Aspects of Wireless Sensor Networks*, 2004.

[7] S. Funke. Topological hole detection in wireless sensor networks and its applications. In *Proc. of the Joint Workshop on Foundations of Mobile Computing*, 2005.

[8] S. Funke and C. Klein. Hole detection or: "How much geometry hides in connectivity?". In *Proc. of the 22nd Symp. on Computational Geometry*, 2006.

[9] A. Kröller, S. P. Fekete, D. Pfisterer, and S. Fischer. Deterministic boundary recognition and topology extraction for large sensor networks. In *Proc. of the 17th Symp. on Discrete Algorithms*, 2006.

[10] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Unit disk graph approximation. In *Proc. of the Joint Workshop on Foundations of Mobile Computing*, 2004.

[11] A. Rao, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *Proc. of the 9th Int. Conf. on Mobile Computing and Networking*, 2003.

[12] S. Schmid and R. Wattenhofer. Algorithmic Models for Sensor Networks. In *Proc. of the 14th Int. Workshop on Parallel and Distributed Real-Time Systems*, 2006.

[13] Y. Wang, J. Gao, and J. S. Mitchell. Boundary recognition in sensor networks by topological methods. In *Proc. of the 12th Int. Conf. on Mobile Computing and Networking*, 2006.