Detecting the Boundary of Sensor Networks from Limited Cyclic Information

Carlos Lara-Alvarez, 1 Juan J. Flores, 2 and Chieh-Chih Wang 3

1 Centro de Investigacion en Matematicas (CIMAT), Unidad Zacatecas, 98068 Zacatecas, ZAC, Mexico
2 Division de Estudios de Posgrado, Facultad de Ingenieria Electrica, Universidad Michoacana, 58390 Morelia, MICH, Mexico
3 Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan

Correspondence should be addressed to Carlos Lara-Alvarez; carlos.lara@cimat.mx

Received 8 September 2014; Revised 15 January 2015; Accepted 21 January 2015

1. Introduction

Sensor networks can be seen as a large collection of nodes that reason about the state of the world, that is, their environment. Distributed sensing improves the signal-to-noise ratio because sensors are closer to the phenomena. Obtaining the boundary of a sensor network is important in many engineering applications. Boundaries are useful to detect the breakdown of a network, insufficient coverage or connectivity, abnormal functioning sensors, virtual coordinates for routing, and so forth. Boundary recognition is a challenging issue in wireless sensor networks [1]. The distinction between inner and boundary nodes of the network can provide valuable knowledge to a broad spectrum of algorithms. In practice, sensors are deployed randomly in a given area. Even if the initial deployment is known, the topology may vary because of sensors failure. Furthermore, many applications require dynamic sensors.

Ad hoc networks are often modeled as unit disk graphs (UDG), where each point can reach other points in a given radius \( \lambda \). Accurately defining the boundary of a UDG without a general position system (GPS) is a challenge that should not be underestimated.

The problem we address in this paper is to find the boundaries of a set of points by using a limited range-cyclic order detector. This sensor, denoted by lrcod, can detect nearby objects and enumerate them by their cyclic order; neither distance nor the angular position of each object is provided. Boundaries are important in many applications such as detecting the breakdown of networks, insufficient coverage or connectivity, abnormal functioning sensors, and virtual coordinates for routing. We studied the information space of the lrcod sensors and established their capabilities to find inner and outer boundaries. Our proposed approach uses local information to recognize points on the boundary. To discover the complete boundary we define the Right Hand Without Crossings (RHWoC) rule. We also provide a correctness proof of this rule. The experimental evaluation confirms the effectiveness to find the boundary of large sensor networks.
with respect to its current position. In contrast to [6], which considers an infinite range sensor model, this paper studies a sensor that can only detect objects at a maximum range of \( \lambda \). A limited sensing model is more convenient for real applications; for example, laser scans have a maximum range, walls in indoor environments can occlude some objects, and so forth.

The rest of the paper is organized as follows: Section 2 introduces some basic definitions used along this paper. Section 3 describes the assumed network model and defines the problem. Section 4 describes how to detect a single point on a boundary and using it to recover the complete boundary. Section 5 describes the simulations and results obtained from the proposed algorithm, showing that it works under different conditions. Section 6 describes the related work. Finally, Section 7 concludes this work.

2. Basic Definitions

\text{lrcod} is a sensor that gives the counterclockwise cyclic permutations of other sensors or landmark labels. This sensor is able to see landmarks only within a given distance; for a given sensor the parameter \( \lambda \) is the maximum range at which the cyclic order detector can sense a landmark. Henceforth, \text{lrcod}(x) is a function that returns the readings of the sensor; those readings are permutations of the set of sensor identifiers as seen from the point \( x \in \mathbb{R}^2 \). Figure 1 illustrates an example of the sequences of landmarks detected from two positions \text{lrcod}(v_1) = [v_5, v_3, v_1, v_6] and \text{lrcod}(v_2) = [v_4, v_5, v_2].

A set of landmarks is in \( \mathbb{R}^2 \). Let \( G = (V, E) \) be a connected graph where \( V \) is the set of landmark labels and \( E = \{ \{x, y\} \mid x, y \in V, \|x - y\| \leq \lambda \} \), where \( \| \cdot \| \) denotes the Euclidean distance between two points. An edge between two landmarks \( x, y \in V \) means that \text{lrcod} can detect the landmark \( y \) from landmark \( x \) and vice versa. We assume that graph \( G \) is connected; therefore every landmark is detectable from at least one landmark.

**Circular Sector.** Given two landmarks \( v_i \) and \( v_j \) seen from \( v_s \), we define \( \Lambda'(v_s; v_i, v_j) \) as the points within the counterclockwise circular sector \( v_i v_j v_s \). Let the output of the sensor be of the form \text{lrcod}(v_s) = [A, v_s, B, v_s, C], in which \( A, B, \) and \( C \) are subsequences of \text{lrcod}(v_s) separated by the labels corresponding to landmarks \( v_i \) and \( v_j \); then \( \Lambda'(v_s; v_i, v_j) = B \) and \( \Lambda'(v_s; v_j, v_i) = C + A \), where + is the concatenation of two sequences.

The sensor \( v_s \), shown in Figure 2 reads \text{lrcod}(v_s) = [v_5, v_3, v_5, v_1, v_5, v_2]. Landmarks \( v_5, v_3 \) define two sectors \( \Lambda'(v_5; v_3, v_5) = \{v_3\} \) and \( \Lambda'(v_3; v_5, v_5) = \{v_5, v_2\} \).

**Walk.** A walk in a graph \( G \) is a sequence \( W := v_0 e_0 v_1 \cdots v_{\ell-1} e_{\ell-1} v_\ell \), whose components are alternately vertices and edges of \( G \) (not necessarily distinct), such that \( v_0 \) and \( v_0 v_1 \) are the ends of edge \( e_0 \). We say that \( W \) connects the initial point \( v_0 \) to the final point \( v_\ell \) and refer to \( W \) as a \( v_0 v_{\ell-1} \)-walk or simply \( v_0 W v_{\ell-1} \). A walk in a graph is closed if its initial and terminal vertices are identical. In case of no confusion we use the succession of vertices \( W = v_0 v_1 v_2 \cdots v_{\ell-1} v_\ell \) to denote the walk \( W = v_0 e_0 v_1 \cdots v_{\ell-1} e_{\ell-1} v_\ell \). The vertices of a closed walk also define a polygon \( P \).

**Polygon.** A polygon \( P \) is a finite sequence \( P = v_0 v_1 \cdots v_{p-1} \) of points in the plane. The set of edges of a polygon is \( E(P) = \{e_i \mid i = 0, \ldots, p-1\} \), where \( e_i \) is defined as an edge connecting \( v_i \) and \( v_{(i+1) \mod p} \).

**Simple Polygon.** A simple polygon is a polygon composed of nonrepeating vertices where none of its edges \( e_i \) and \( e_j \), for \( i \neq j \), intersect at points different from their extremes.

The Jordan curve theorem states that every simple closed curve divides the plane into an interior region bounded by the curve and an unbounded region containing all exterior points.

**Weakly Simple Polygon.** A polygon \( P \) is weakly simple if, for any \( \varepsilon > 0 \), there is a simple closed curve \( Q \) such that \( d_{\mathcal{F}}(P, Q) < \varepsilon \), where \( d_{\mathcal{F}} \) is the Fréchet distance [8] between two polygons \( P \) and \( Q \).
In other words, a polygon is weakly simple if it can be made simple by an arbitrarily small perturbation of the entire curve. Informally, a weakly simple polygon is a polygon in which sides can touch each other but cannot cross. Any weakly simple polygon \( P \) partitions \( \mathbb{R}^2 \setminus P \) into two regions: interior and exterior.

\( \lambda \)-Polygon. A \( \lambda \)-polygon \( P \) is a weakly simple polygon where \( \|e_i\|\leq\lambda \) for all \( e_i \in E(P) \).

\( \lambda \)-Boundary. A \( \lambda \)-boundary or outer boundary \( P \) of a set of points \( V \) is a \( \lambda \)-polygon such that \( V \setminus P \) lies in the interior of \( P \).

Diagonal. A diagonal is a line that connects two nonconsecutive vertices of a polygon; an inner diagonal lies totally in the interior of the polygon.

Hole. A hole or inner boundary is an empty \( \lambda \)-polygon with vertices \( H \subset V \) such that every inner diagonal is larger than \( \lambda \). A hole with \( n = |H| \) vertices will be called \( n \)-hole.

3. Network Model

Throughout this paper, we consider sensor networks that satisfy the following assumptions.

(i) The system is composed of a set of sensors \( V \), located in a region \( R \subset \mathbb{R}^2 \); each sensor \( v \in V \) has a distinctive identity.

(ii) Every sensor \( v \in V \) is able to detect \( \text{lrcod}(v) \).

(iii) All wireless sensors have the same maximum transmission range \( \lambda \).

Assumptions (i) and (iii) involve that each sensor can gather information from other sensors within its transmission range. Therefore, we can build a network graph whose vertices correspond to the labeled nodes and whose edges correspond to communication links. Henceforth, we will indistinctly call them sensors, nodes, or landmarks.

Assumption (i) admits that sensors can move arbitrarily in \( R \) but they are static for a period of time \( \tau \) long enough to recover the boundaries. Furthermore, we assume that landmarks are in general position (i.e., no three landmarks are collinear). Assumption (iii) is usual in \( \text{ad hoc} \) networks modeled as unit disk graphs (UDG), where each point can reach other points in a given radius \( \lambda \).

Problem Definition. Given a network \( V \) of lrcod sensors, find the boundary polygon of \( V \) and its holes by using only the local information provided by the lrcod sensors.

The task we are solving is to find the perimeter of a set of points; this task can be done by simply following the sequence of extreme points on the convex hull. Classical algorithms such as the Graham Scan [9] can be used to find the convex hull (CH) of the point set.

Unfortunately, following it by using the limited range-cyclic order detector is in certain cases impossible, because the distance between two consecutive extreme points could be greater than the maximum range of the sensor. Hence, when following the CH of the set of lrcod sensors, the algorithm can reach a position where the sensor cannot see the next point in the CH, so it has to share information through other points inside the CH (points at sight) until the next point on the CH (not at sight before) is reached. Once it has done so, it can continue finding points on the CH. This approach is time consuming and sometimes unnecessary.

4. Proposed Algorithm

The proposed approach finds points on the boundary following the procedure described in Section 4.1. Those points can be used as seeds to find the complete sequence of points on the boundary, as described in Section 4.2.

4.1. Boundary Points. This subsection presents a series of definitions and lemmas that are the basis for discerning boundary from inner points. According to our definition of holes a \( \lambda \)-polygon \( P \) formed by less than four points cannot be a boundary; hence, three connected points on the vertices of a triangle cannot form a hole; therefore, a hole must have at least four landmarks.

\( \lambda \)-Point. A \( \lambda \)-point is a point \( v_j \) for which \( \exists v_i, v_k \in \text{lrcod}(v_j) \) such that \( \Lambda'(v_i; v_j; v_k) = \emptyset \) and \( v_i \notin \text{lrcod}(v_k) \). Points \( v_i, v_k \) are called adjacent boundary points of \( v_j \).

4.2. Detecting a Boundary from \( \lambda \)-Points. This subsection addresses how to use \( \lambda \)-points as seeds to find the boundary sequence.

Let \( h_j \) and \( h_{j+1} \) be two consecutive points on the convex hull and let \((h_j, h_{j+1})\) be the line segment joining them. Since the graph \( G \) is connected, there exists at least one \( h_j h_{j+1} \)-walk. We are looking for the walk such that the area comprised by \( W' := h_j W h_{j+1}(h_{j+1}, h_j) h_j \) does not contain any landmark \( v \in V \); we refer to the walk that satisfies this condition as a pocket.

The right-hand rule (RH) is a heuristic rule that has been used extensively to deliver packages in \( \text{ad hoc} \) networks [10]. It has also been used to find the convex hull for an unlimited cyclic sensor [6]. The right-hand rule states that if the last two previously visited vertices of a walk are \( v_{j-1}, v_j \) then the next vertex is the first vertex that appears immediately by turning right from the edge \( (v_{j-1}, v_j) \). When the greedy multi-hop forwarding algorithm gets stuck, the right hand rule can be used to go around holes.

To complete a boundary starting from a border walk, we define the Right Hand Without Crossings (RHWoC) rule. Given a partial walk \( W' = [v_0 \cdots v_{j-1} e_{j-1} v_j] \) such that its last edge is \( (v_{j-1}, v_j) \), the RHWoC rule selects the first landmark \( v_k \in \text{lrcod}(v_j) \) that follows \( v_{j-1} \) in the permutation such that the edge \( e_j := (v_j, v_k) \) does not intersect at a single point to any previous edge in the walk. To illustrate the RHWoC rule, Figure 4 shows an example to find the next point of the walk \( W = v_1 e_{12} v_2 e_{24} v_4 e_{46} v_6 \). When the point \( v_6 \) is analyzed, \( v_3 \) appears in the permutation immediately after point \( v_2 \) but it is not selected because the edge \( (v_6, v_3) \) intersects the boundary; then the correct point \( v_5 \) is selected.
4.3. Discovering Intersections. The RHWoC rule selects an edge that does not cross any previous edge in the border walk, but the border walk $W$ could be a long sequence of vertices and edges. The following lemma states that when two edges intersect at a single point, at least one vertex of an edge must see at least one vertex of the other edge.

Lemma 1. Let $e_{ij} = (v_i, v_j)$ and $e_{kn} = (v_k, v_n)$ be two edges, and let $I_{\text{rcod}} := I_{\text{rcod}}(v_k) \cup I_{\text{rcod}}(v_n)$ be the landmarks seen from $v_k$ or $v_n$. If $(v_i \notin I_{\text{rcod}}(v_k)) \land (v_j \notin I_{\text{rcod}}(v_n))$ then $e_{ij}$ does not intersect $e_{kn}$.

Proof. When $e_{ij}$ and $e_{kn}$ intersect at a single point $q$, there exist two points $v_a \in \{v_i, v_j\}$ and $v_b \in \{v_k, v_n\}$ such that $\|v_aq\| \leq \lambda/2$ and $\|v_bq\| \leq \lambda/2$. From the law of cosines it can be shown that $\|v_aq\| \leq \lambda$; in other words, $v_b \in I_{\text{rcod}}(v_a)$. □

We are interested in determining if two different edges intersect in a single point. When discovering crossing edges, the endpoints of two edges could be the same; hence, there is only one rather than two distinct edges. This condition is easily detectable and can be used to discover connectivity issues of the network such as bridges and articulation points [11].

Lemma 2 states how to use the information provided by $I_{\text{rcod}}$ to detect that two edges in a walk intersect in a single point.

Lemma 2. Let $v_m e_{hj} v_k e_{ij} v_{k} e_{jk} \cdots v_m$ be a border walk, and let $v_n \in I_{\text{rcod}}(v_m)$ be a point seen from $v_m$; edge $e_n := (v_m, v_{n})$ intersects edge $e_{ij}$ in a single point when $(v_n \in A_j(v_k, v_j)) \land (v_n \in A_i(v_j, v_h))$ and $I_{\text{rcod}}(v_n)$ is arranged as $[v_i, A, v_k, B, v_j, C]$ with $A$, $B$, and $C$ subsequences of $I_{\text{rcod}}(v_n)$.

Proof. Figure 5(a) illustrates this lemma for $v_m = v_k$. Every point satisfying $(v_n \in A_j(v_k, v_j)) \land (v_n \in A_i(v_j, v_h))$ is in the red shadow. When $I_{\text{rcod}}(v_j) = [v_i, A, v_k, B, v_j, C]$, meaning that $v_i$ is in the circular sector (of a large circle) $v_j v_k v_i$, then $e_{kn}$ intersects $e_{ij}$. □

Corollary 3. Let $v_j$ be a $\lambda$-point with adjacent boundary points $v_i, v_k$. If $v_j \notin A_j(v_i, v_k)$, $e_{kn} := (v_k, v_n)$ does not intersect $e_{ij}$. In other words, a walk of four consecutive boundary points can be found from two consecutive $\lambda$-points.

Proof. Corollary 3 follows directly from Lemma 2. In Figure 5(b), the edge $(v_k, v_j)$ does not cross any edge when $v_n \notin A_j(v_k, v_j)$. Point $v_k$ is also a $\lambda$-point because $A_j(v_j, v_k)$ is empty. □

Observation 1. It is always possible to define intersections for a walk with four or more vertices by using Lemma 2.

Although it is useful to find inner diagonals of a hole, $\lambda$-points are better to find points on the boundary when two consecutive points on the convex hull cannot see each other. Sometimes small holes do not have detectable $\lambda$-points; for instance, the 4-hole shown in Figure 3(a) can be detected because $\lambda_j(v_k, v_j)$ is empty, but $\lambda_j(v_k, v_j) = \{\}$ in Figure 3(b). Those undetectable cases never occur for points on the convex hull.

Observation 2. If there is only one single $\lambda$-point (a walk with three points instead of four) then Corollary 3 can be used. We can define some other cases such as that shown in Figure 5(c) where $v_5 \in A_j(v_4, v_2)$ and $I_{\text{rcod}}(v_5)$ reads $[v_5, A, v_k, B, v_j, C]$. Unfortunately, the limited range of the sensor prevents us from using this case for all situations because sometimes $v_i \notin I_{\text{rcod}}(v_n)$. We say that the intersection is undetermined for the cases where we cannot determine if an edge intersects or not with another edge of a walk. In such cases, the boundary could be completed by using other $\lambda$-points.
Data: A finite set of connected points \( V \subset \mathbb{R}^2 \).
Result: A list \( \mathcal{L} \) of the boundary and detectable holes of \( V \).
(1) Exchange sensor information between neighbors;
(2) \( \mathcal{L} := \{ \} \);
(3) \textbf{foreach} \( v \in V \) \textbf{do}
(4) \textbf{if} \( v \) is a \( \lambda \)-point with adjacent points \( v_a, v_b \) \textbf{then}
(5) \( \mathcal{W} := v_a(v_a, v)(v, v_b) \);
(6) \textbf{if} \( \mathcal{W} \) is not used in \( \mathcal{L} \) and \( v \) is a \( \lambda \)-point with adjacent points \( v_c, v \) \textbf{then}
(7) \( \mathcal{W} := \text{RHWoC}(v_a, v_b, v_c) \);
(8) \( \mathcal{L} := \mathcal{L} \cup \mathcal{W} \);
(9) \textbf{end}
(10) \textbf{end}
(11) \textbf{return} \( \mathcal{L} \)

\textbf{Algorithm 2: Function RHWoC} \((v_{i-3}, v_{i-2}, v_{i-1}, v_i)\).

(1) \( e_j := (v_{i-1}, v_i) \);
(2) \( v_{j-1} := v_{i-1} \);
(3) \( v_j := v_i \);
(4) \( \mathcal{W} := v_{i-3} v_{i-2} v_{i-1} v_i \);
(5) \textbf{repeat}
(6) \( \mathcal{L} := \text{lrcod}(v_j) \);
(7) \textbf{Circular-rotate} \( \mathcal{L} \) until \( v_{j-1} \) is the last element of \( \mathcal{L} \);
(8) \textbf{foreach} \( v_j \in \mathcal{L} \) \textbf{do}
(9) \textbf{if} \( (k = j - 1) \) or \( (v_j, v_k) \) does not intersect any edge of \( \mathcal{W} \) \textbf{then}
(10) \( e_j := (v_j, v_k) \);
(11) \( v_{j-1} := v_j \);
(12) \( v_j := v_k \);
(13) \( \mathcal{W} := \mathcal{W} e_j v_j \);
(14) \textbf{break};
(15) \textbf{end}
(16) \textbf{end}
(17) \textbf{until} \( e_j \notin \mathcal{W} \);
(18) \textbf{return} \( \mathcal{W} \)

4.4. Algorithm Overview. To find the boundaries of a set of points \( V \) we propose Algorithms 1 and 2. In Algorithm 1 each sensor first obtains local information (line 1), then it applies the definition of Section 4.1 to recognize two consecutive \( \lambda \)-points (lines 4–6). If the \( \lambda \)-point and its corresponding adjacent points are not already used, then a boundary (or a hole) has been detected and the function RHWoC is called (line 7). Function alg::closed recovers the complete border until it finds an edge that already is on the boundary. When no more borders can be found then the list \( \mathcal{L} \) is returned (line 12).

Correctness of Function RHWoC. Let \( \mathcal{W} \) be the outer area reachable with the lrcod from vertices of \( \mathcal{W} \). At each step, the next landmark \( v_k \) in the walk is selected according to the RHWoC rule. The rule’s precondition asserts that \( v_1, \ldots, v_{j-1} \) are \( \lambda \)-convex hole points; then \( \mathcal{W} \) does not contain any landmark. For the following steps, RHWoC rule ensures that \( S_j = \Lambda^i(v_j v_{j-1}, v_k) \cap \mathcal{W} \) does not contain any landmark. As \( \bigcup_{v_j \in \mathcal{W}} S_j = \mathcal{W} \) and the walk never crosses itself, the Jordan theorem asserts that \( \mathcal{W} \) is a boundary and the set of landmarks \( V \setminus \mathcal{W} \) are in the same side of \( \mathcal{W} \). Finally, based on our connectivity assumption, the walk reaches a previously detected edge and the cycle of lines 5–17 terminates correctly.

5. Simulations
The proposed algorithm has been implemented for randomly generated sets in a quadrangle of \( 30 \times 30 \). The number of nodes and the value of \( \lambda \) are user-defined parameters. Examples of networks for different values of \( n \) are illustrated in Figure 6. The proposed algorithm finds correctly the outer boundaries of networks which are relatively sparse (Figures 6(a) and 6(b)) and dense (Figures 6(c) and 6(d)). Figures 6(a) and 6(b) also show the detectable inner boundaries.
Figure 5: Discovering intersections. Using sectors and landmark ordering, one can determine when two edges intersect at a single point.

Figure 6: Boundaries obtained from a set of $N$ uniform distributed random points within a quadrangle of $30 \times 30$. 
6. Related Work

A well-known technique to find the boundary of a set of points is the $\alpha$-shape [12]. Alpha shapes are closely related to alpha complexes, subcomplexes of the Delaunay triangulation of the point set.

Some approaches to detect boundaries in networks are (a) distribution of nodes in the nonhole regions [13], (b) length of the shortest path between two nodes to detect cycles [14, 15], and (c) patterns that the nodes produce [1].

In robotics, landmarks are useful for navigation, place recognition, localization, and mapping. The proposed approach deals with a more challenging setting compared with [6], as it is necessary to detect the cyclic angular order of landmarks in an unlimited range can accomplish simple tasks, such as navigation and patrolling. One of the advantages of this approach is that the memory of the robot does not save the complete state of the environment. To compensate, it considers an infinite range in the landmark order detector.

7. Conclusions

The problem we address in this paper is to find the boundaries of a sensor network by using an extended range-cyclic order detector. By using local information we show how to detect $\lambda$-points; these points are on the boundary or on a hole. The Right Hand Without Crossings (RHWoC) rule can be used to discover other landmarks from an incomplete border walk. We provide a correctness proof of the RHWoC rule.

Some holes could not be detected by using local information. We also show the configurations where the boundary is detectable; the most remarkable case is when there are two consecutive $\lambda$-points. Therefore, the proposed technique is appropriate for large sensor networks. The experimental evaluation confirms the effectiveness to find the boundary of large sensor networks.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] O. Saukh, R. Sauter, M. Gauger, and P. J. Marrón, “On Boundary recognition without location information in wireless sensor networks,” ACM Transactions on Sensor Networks, vol. 6, no. 3, article 20, 2010.

[2] P. Ferrat, C. Gimkiewicz, S. Neukom, Y. Zha, A. Brenzikofer, and T. Baechler, “Ultra-miniature omni-directional camera for an autonomous flying micro-robot,” in Conference on Optical and Digital Image Processing, vol. 7000 of Proceedings of SPIE, 2008.

[3] B. Tavli, K. Bicakci, R. Zilan, and J. M. Barceló-Ordinas, “A survey of visual sensor network platforms,” Multimedia Tools and Applications, vol. 60, no. 3, pp. 689–726, 2012.

[4] J. Pugh, X. Raemy, C. Favre, R. Falconi, and A. Martinoli, “A fast onboard relative positioning module for multirobot systems,” IEEE/ASME Transactions on Mechatronics, vol. 14, no. 2, pp. 151–162, 2009.

[5] J. F. Roberts, T. Stirling, J.-C. Zufferey, and D. Floreano, “3-D relative positioning sensor for indoor flying robots,” Autonomous Robots, vol. 33, no. 1-2, pp. 5–20, 2012.

[6] B. Tovar, L. Freda, and S. M. Lavalle, “Learning combinatorial map information from permutations of landmarks,” International Journal of Robotics Research, vol. 30, no. 9, pp. 1143–1156, 2011.

[7] A. Bondy and U. S. R. Murty, Graph Theory, Graduate Texts in Mathematics, Springer, 2008.

[8] A. Ribó Mor, Realization and counting problems for planar structures: trees and linkages, polytopes and polyominoes [Ph.D. thesis], Freie Universität Berlin, Berlin, Germany, 2006.

[9] R. L. Graham and F. F. Yao, “Finding the convex hull of a simple polygon,” Journal of Algorithms, vol. 4, no. 4, pp. 324–331, 1983.

[10] R. B. Muhammad, “A distributed graph algorithm for geometric routing in ad hoc wireless networks,” Journal of Networks, vol. 2, no. 6, pp. 50–57, 2007.

[11] G. Ausiello, D. Firmani, and L. Laura, “Real-time monitoring of undirected networks: articulation points, bridges, and connected and biconnected components,” Networks, vol. 59, no. 3, pp. 275–288, 2012.

[12] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel, “On the shape of a set of points in the plane,” IEEE Transactions on Information Theory, vol. 29, no. 4, pp. 551–559, 1983.

[13] S. Funke and C. Klein, “Hole detection or: how much geometry hides in connectivity?,” in Proceedings of the 22nd Annual Symposium on Computational Geometry (SCG ‘06), pp. 377–385, ACM, New York, NY, USA, 2006.

[14] A. Kröller, S. P. Fekete, D. Pfisterer, and S. Fischer, “Deterministic boundary recognition and topology extraction for large sensor networks,” in Proceedings of the 17th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA ’06), pp. 1000–1009, ACM, New York, NY, USA, 2006.

[15] Y. Wang, J. Gao, and J. S. B. Mitchell, “Boundary recognition in sensor networks by topological methods,” in Proceedings of the 12th Annual International Conference on Mobile Computing and Networking (MobiCom ’06), pp. 122–133, ACM, New York, NY, USA, September 2006.