SURROUNDING NODES IN COORDINATE-FREE NETWORKS ⋆

R. Ghrist[1], D. Lipsky[1], S. Poduri[2], and G. Sukhatme[2]

[1]Department of Mathematics, University of Illinois, Urbana, IL 61801, USA
[2]Department of Computer Science, University of Southern California, Los Angeles, CA, USA

Summary. Consider a network of nodes in the plane whose locations are unknown but which establish communication links based on proximity. We solve the following problems: given a node in the network, (1) determine if a given cycle surrounds the node; and (2) find some cycle that surrounds the node. The only localization capabilities assumed are unique IDs with binary proximity measure, and, in some cases, cyclic orientation of neighbors. We give complete algorithms for finding and verifying surrounding cycles when cyclic orientation data is available. We also provide an efficient but non-complete algorithm in the case where angular data is not available.

1 Introduction

It is increasingly important to analyze networked collections of sensors, robots, communication devices, or other local agents which coordinate to solve global problems. A similar problem arose in mathematics a century ago — how to extract global properties of a space built from local, combinatorially defined pieces, or *simplices*. It is not a coincidence that the techniques developed to solve such mathematical problems (algebraic topology) provide perspectives and tools applicable to this latest incarnation of the problem.

This paper considers a network version of a simple classical problem in algebraic/differential topology. Given a point x_0 in the plane $\mathbb{R}^2$ and a simple closed curve, determine whether or not the curve surrounds the point — that is, whether the *winding number* of the curve about x_0 is nonzero (Fig. 1[left]). In the topological setting, this problem is very easily solved using simple topological methods [9]. In a network-theoretic version of the problem, x_0 is a node in a network graph Γ whose vertices represent non-localized sensors in the plane and whose edges encode proximity; and $\mathcal{L}$ is an abstract cycle in this graph (Fig. 1[right]).

⋆ Supported by DARPA HR0011-05-1-0008 and NSF PECASE DMS-0337713 [RG,DL] and by NSF CCR-0120778, CNS-0520305, and IIS-0133947 [SP,GS].

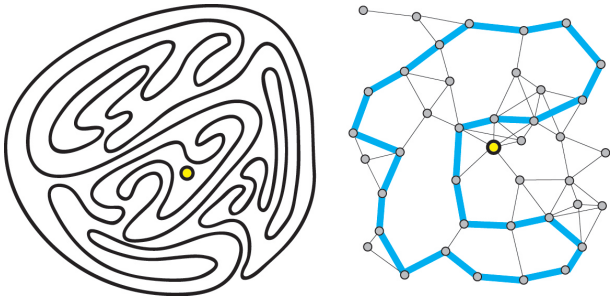

Fig. 1. [left] Is the node inside the curve or outside? For a network without localization [right], this can be challenging.

Several such problems about winding numbers have very natural motivations, and are especially challenging when the nodes are not localized. Consider as an example, a networked collection of sensors (e.g., accelerometers or acoustic sensors) which are distributed in a 2-d domain. Given a certain node x_0 which registers an important reading (an alarm), one problem relevant to security applications is to determine whether the detection has occurred within a region of particular importance whose perimeter is defined by a cycle in the network. Similarly, it an alarm goes off at a node, one might wish to find a small subcollection $\mathcal{L}$ of sensors whose sensing domains are guaranteed to 'surround' the node x_0 in the plane: thus, the embedding of the cycle $\mathcal{L}$ in the plane is a curve which surround x_0.

If one has sufficient data to localize nodes, then all such problems about winding numbers are trivial to solve and computationally efficient solutions exist in the computer graphics literature under various simplifying assumptions. The assumption of localized nodes is natural for any number of systems involving stationary nodes placed intentionally, e.g., video cameras. However, in the case of nodes which are distributed in an unpredictable and non-uniform manner, or in which the nodes are mobile, then localized nodes are no longer a priori natural. Robotics, in particular, presents a natural setting in which mobile devices communicating via an ad hoc wireless network can provide localization challenges.

1.1 Related Work

There is a substantial and growing literature on geometric properties of ad hoc networks in which localization is weakened or not assumed at all. The recent work on routing without localization initiated by [13] uses a heat-flow to determine virtual coordinates for a non-localized network for applications to weighted routing problems. In many cases [13, 7] a set of known landmarks is used to estimate system geometry. All these methods are effective, but, with a few exceptions [10] non-rigorous. Recent work of Fekete et al. [8] gives a dis-

tributed algorithm for rigorous topology exploration, boundary detection, and surrounding cycles: the algorithm is complete when the nodes are sufficiently dense.

There is a large body of work on coarse distance estimation in ad hoc networks augmented with angular data in the form of the angle of separation between a node's neighbors. This arises in the the paper [6], which uses a network graph along with exact angular measures of neighbors to detect holes in the physical network and perform routing. The work detailed in [12] gives criteria for ensuring coverage in a sensor network using bounds on separation angles among neighbors.

Very recently, algebraic topology has been recognized as a novel tool for problems in sensor and ad hoc networks. The papers [3, 4, 5] present algebraic topological criteria for coverage in sensor networks. Recent unpublished results of Y. Baryshnikov relating to hole-detection in networks with randomly distributed nodes uses Betti numbers to perform boundary detection.

As a problem in computational geometry, networks with no localization and proximity measurements arise in the literature on *unit disc graphs*: abstract graphs whose vertices correspond to a set of nodes in the plane and whose edges are determined by nodes within unit distance. Clearly, not all graphs are realizable as a unit disc graph. Recognizing whether a graph is a realizable unit disc graph is NP-complete [1]. It follows that finding some embedding of an abstract unit disc graph into the plane for which the graph is the unit disc proximity network is also NP hard. Even finding an 'approximate' embedding which realizes a unit disc graph up to local errors is NP hard [11]. But, using angular data, [2] gives an algorithm for finding a realization of a spanner of the unit disc graph, which enables one to compute virtual coordinates and approximate some locations.

1.2 Innovations

The perspective that guides our techniques is that of topology, more specifically, winding numbers [9]. Recall that the winding number of a planar cycle $\mathcal{L}$ about a point $x \in \mathbb{R}^2$ is, roughly speaking, the number of times the cycle wraps around the point. It can be computed in a number of ways: analytically, via integrating a tangent vector about $\mathcal{L}$; topologically, via computing the homology class of $\mathcal{L}$ in the complement of $x_0 \in \mathbb{R}^2$; or combinatorially, via computing the intersection number of $\mathcal{L}$ with a ray based at x in $\mathbb{R}^2$. We develop an approach to computing winding numbers which is adapted to networks and differs from all three above.

In §3, we solve a **separation** problem: to compute whether a given node x_0 is surrounded by the image of a given cycle $\mathcal{L}$. In §4, we solve an **isolation** problem of determining whether a given node x_0 is surrounded by some cycle $\mathcal{L}$ and constructing an explicit cycle. The algorithms we present can be implemented in systems as a distributed local computation, in which nodes

are assumed to have a limited amount of memory and simple processing abilities. We present results of simulations in §7. In §5 we deal with managing uncertainty in cyclic orientation data, and in §6 we present an algorithm for systems which possess no cyclic orientation data whatsoever. In both these cases, as in [8], non-complete algorithms exist which provide certificates.

2 Problem Formulation

2.1 Assumptions

Throughout this paper, we consider networks which satisfy some or all of the following assumptions.

- **P** (Planar) Nodes with unique labels lie in the Euclidean plane $\mathbb{R}^2$.
- **N** (Network) Nodes form the vertices of a connected unit disc network graph Γ of sufficiently large diameter.
- **O** (Ordering type) Each node can determine the clockwise cyclic ordering of its neighbors in the plane.

There are no coordinates, no node localization, and no assumptions about node density or distribution other than sufficient extent. Assumptions **P** and **N** will be in force for the remainder of this paper. Assumption **O** will sometimes not be imposed.

Assumption **P** and **N** imply that nodes can broadcast their unique IDs and these can be detected by any neighboring nodes within unit distance. This creates a network graph whose vertices correspond to the labeled nodes and whose edges correspond to communication links. There is no metric information encoded in an edge beyond the coarse datum that the distance between the nodes in the plane is no more than one.

Assumption **O** means that each node can perform a clockwise "sweep" of its neighborhood and determine the order in which neighbors appear. More specifically, there is a cyclic total ordering $\lhd$ on the neighbors of a node x_0 which defines the counterclockwise (CCW) order in which they appear. There is no "compass" and thus ordering is known only up to a cyclic permutation. There is also no angular data assigned to the ordering: an oriented pair of neighbors may form an arbitrary (nonzero) angle with x_0 without changing the angular ordering. This type of coarse angular data is not too uncommon in robotics contexts. Cyclic orientation data is natural in, e.g., primitive landmark vision systems, radar networks, and robots with gap sensors.

Definition: Let x_0 be a node and $\{x_i\}_1^3$ be a triple of distinct neighbors of x_0. Define the index

$$\mathcal{I}_{x_0}(x_1; x_2, x_3) := \begin{cases} +1 : x_1 \lhd x_2 \lhd x_3 \lhd x_1 \\ -1 : x_1 \rhd x_2 \rhd x_3 \rhd x_1 \end{cases} \tag{1}$$

Geometric interpretation: The pair of rays in $\mathbb{R}^2$ from x_0 passing through x_2 and x_3 join at x_0 to form an bent line that divides the plane. Orient this bent line using the ordering $(x_2 \to x_0 \to x_3)$. The index $\mathcal{I}_{x_0}(x_1; x_2, x_3) = -1$ iff x_1 lies to the left of this line, and $\mathcal{I}_{x_0}(x_1; x_2, x_3) = +1$ iff x_1 lies to the right of it.

The limit in which two neighbors have angle zero with x_0 leads naturally to the problem of uncertainty in angular ordering. When the angle between neighbors is so small as to interfere with the orientation type, the data is weaker and the problems more subtle. For the present, we assume that nodes are in a 'general position' so as to possess a positive lower bound on angles. This is, of course, completely unrealistic in practice. Later, in §6 we consider this more carefully and allow for nodes to be unable to distinguish the angular ordering of certain neighbors. For most (but not all) networks, there is a surprisingly large tolerance for angular ordering blindness.

The input data for the problem is the network graph Γ. When Assumption **O** is in place, the graph has vertices augmented with the cyclic ordering type of its immediate neighbors.

2.2 Problem statements

Definition: The projection map $\Gamma \mapsto \overline{\Gamma} \subset \mathbb{R}^2$ maps vertices of Γ to the position of the corresponding node in the Euclidean plane and edges of Γ to the line segment connecting the nodes. These line segments all have length bounded above by one. We solve two problems concerning winding numbers of cycles:

> **Separation:** Given a cycle $\mathcal{L}$ in the network graph Γ and a node $x_0 \in V(\Gamma)$ which is disjoint from the nodes of $\mathcal{L}$, determine whether the projected cycle $\overline{\mathcal{L}}$ surrounds x_0.

The image of the cycle $\overline{\mathcal{L}}$ in the plane is a closed piecewise-linear curve. If the curve is *simple* (that is, non-self-intersecting), then the Jordan Curve Theorem implies that the cycle separates the plane into two connected components, only one of which is bounded. We will restrict attention to simple cycles, using network criteria to satisfy this condition (Corollary 1).

Our second problem is a constructive version of the previous.

> **Isolation:** Given a node $x_0 \in \mathcal{X}$, find a cycle $\mathcal{L}$ in the network graph Γ whose projection $\overline{\mathcal{L}}$ surrounds x_0 or determine that no such cycle exists.

For reasons of robustness with respect to error, we desire a cycle $\mathcal{L}$ which is not too close to x_0. An interesting generalization of this problem relevant to security applications is to construct a sequence of concentric cycles which isolate the target node x_0 and form 'moats' in $\mathbb{R}^2$ to ring x_0.

3 Separation

3.1 Restrictions

We consider the separation problem for a network satisfying Assumptions **P**, **N**, and **O**. Namely, given a node x_0 and an oriented cycle $\mathcal{L} = (x_i)_1^N$ of cyclically connected nodes distinct from x_0, determine whether the projected cycle $\overline{\mathcal{L}}$ surrounds the node x_0. Let d denote the "hop" distance function on Γ. We assume that $d(x_0, \mathcal{L}) > 1$, meaning that the shortest path from x_0 to a vertex of $\mathcal{L}$ in Γ requires more than one "hop" or edge.

Both Assumption **O** and the bound on $d(x_0, \mathcal{L})$ are necessary. A critical example is illustrated in Fig. 2, which gives two labeled graphs in $\mathbb{R}^2$ with isomorphic network graphs and identical cyclic orientation data. The target node x_0 lies on opposite sides of the cycles illustrated. It is much easier to construct examples of unit disc graphs which can be realized in ways which change winding numbers of cycles.

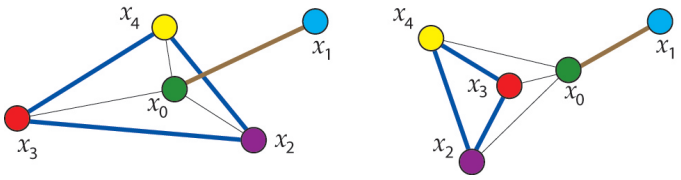

Fig. 2. Two examples of embedded network graphs with identical network and cyclic orientation data.

The problem makes the most sense when the projected cycle $\overline{\mathcal{L}}$ is a simple closed curve in $\mathbb{R}^2$. The easiest way to guarantee such a cycle is to choose a cycle which is 'minimal' with respect to communication between nodes.

Definition: For any subgraph $\Delta \subset \Gamma$, let $\langle \Delta \rangle$ denote the maximal subgraph of Γ spanned by the vertices of Δ. Say that Δ is **chord-free** if $\langle \Delta \rangle = \Delta$. The simplest criterion for a cycle $\mathcal{L}$ to have a simple projection to the plane is that $\langle \mathcal{L} \rangle = \mathcal{L}$. The following lemma is both trivial and well-known [8, 3].

Lemma 1. *If the projections of two edges of a unit disc graph Γ intersect in $\mathbb{R}^2$, then these span a subgraph of Γ containing a cycle of three edges.*

Corollary 1. *Any path (or cycle) $\mathcal{P}$ in a unit disc graph Γ satisfying $\langle \mathcal{P} \rangle = \mathcal{P}$ has image $\overline{\mathcal{P}}$ a non-intersecting (closed) curve in $\mathbb{R}^2$.*

3.2 Algorithm

In differential topology, the way one decides whether a loop in the plane encloses a point is to choose a path from the point which terminates sufficiently

far from the starting point as to be definitely outside the loop. For a 'generic' choice of such a path, the path and the loop intersect transversally (i.e., without tangencies), and the number of intersection points counted mod 2 is zero if and only if the loop does not surround the point [9].

The obvious generalization of this strategy is to choose any path $\mathcal{P}$ in Γ from x_0 to a terminal point which is sufficiently far away from $\mathcal{L}$ to guarantee that it is outside the cycle in $\mathbb{R}^2$, and then court intersections. However, this counting is not always easy or even possible. The inspiration for our method is nearly opposite to that coming from differential topology. Instead of trying to force intersections to be a discrete set of points, one thinks of manipulating the path so as to *maximize* the amount of intersection with the cycle with the result of having a single connected component in the intersection. Then, one could compute whether the endpoints of $\mathcal{P}$ lie on the same side of $\overline{\mathcal{L}}$. This last step is what we do, using the orientation data in a crucial manner.

Fix an orientation for the cycle $\mathcal{L}$ in Γ and order the nodes (ℓ_i) of $\mathcal{L}$ cyclically. Choose a node x_∞ sufficiently far from $\mathcal{L}$ in Γ. Generate chord-free paths $\mathcal{P}_0$ and $\mathcal{P}_\infty$ from x_0 and x_∞ respectively to points on $\mathcal{L}$. The projection of these paths to $\mathbb{R}^2$ are not self-intersecting, and can only intersect $\overline{\mathcal{L}}$ at most once at the last segment of the path.

The crucial step is to determine whether the paths $\overline{\mathcal{P}}_0$ and $\overline{\mathcal{P}}_\infty$ lie on the same side of $\overline{\mathcal{L}}$ or different sides. In the simplest configuration, the final point on a path is connected to $\mathcal{L}$ in Γ by only one edge, as in Fig. 3[left]. The angular orientation data then suffices to determine on which side of $\overline{\mathcal{L}}$ the path lies. However, the situation may be more complicated, as in Fig. 3[right]. A more subtle manipulation of orientation data is required in this case. Details are presented in Algorithm IndexCheck.

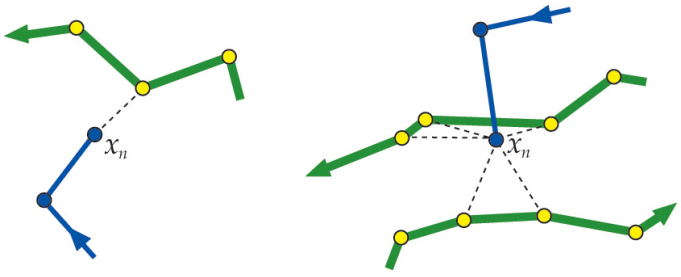

Fig. 3. Determining whether an oriented path with terminal node x_n approaches the projected oriented cycle $\overline{\mathcal{L}}$ from the left or from the right can be simple [left] or complicated [right] depending on the number of communication links between x_n and $\mathcal{L}$.

Algorithm 1 $\mathcal{I} = \mathsf{IndexVertexLoop}(x, \mathcal{L}, \Gamma)$

Require: $\Gamma = (V, E)$ is a graph satisfying **P**, **N**, and **O**
Require: $x \in V(\Gamma)$, $\mathcal{L} = (\ell_i)$ is an oriented cycle of Γ, $\langle \mathcal{L} \rangle = \mathcal{L}$, and $d(x, \mathcal{L}) > 1$.
choose a path $\mathcal{P} = (x_i)_0^n$ in G with $x_0 = x$, $\langle \mathcal{P} \rangle = \mathcal{P}$, and $d(x_i, \mathcal{L}) = 1$ iff $i = n$.
if at x_n, for some j, either $\ell_j \lhd x_{n-1} \lhd \ell_{j+1}$ or $\ell_{j+1} \lhd x_{n-1} \lhd \ell_j$ and no other ℓ_i separates x_{n-1} from these neighbors in $\lhd$ **then**
  return $\mathcal{I} \Leftarrow \mathcal{I}_{\ell_j}(\ell_{j-1}; \ell_{j+1}, x_n) \cdot \mathcal{I}_{\ell_{j+1}}(\ell_{j+2}; x_n, \ell_j) \cdot \mathcal{I}_{\ell_{x_n}}(x_{n-1}; \ell_j, \ell_{j+1})$
else
  if for some j, $d(x_n, \ell_j) = d(x_n, \ell_{j+1}) = 1$ **then**
    return $\mathcal{I} \Leftarrow \mathcal{I}_{\ell_j}(\ell_{j-1}; \ell_{j+1}, x_n) \cdot \mathcal{I}_{\ell_{j+1}}(\ell_{j+2}; x_n, \ell_j) \cdot \mathcal{I}_{\ell_{x_n}}(x_{n-1}; \ell_j, \ell_{j+1})$
  else
    choose any ℓ_j with $d(\ell_j, x_n) = 1$.
    return $\mathcal{I} \Leftarrow \mathcal{I}_{\ell_j}(x_n; \ell_{j-1}, \ell_{j+1})$
  end if
end if

Algorithm 2 $\mathcal{I} = \mathsf{IndexCheck}(x_0, \mathcal{L}, \Gamma)$

Require: $\Gamma = (V, E)$ is a graph satisfying **P**, **N**, and **O**
Require: $x_0 \in V(\Gamma)$, $\mathcal{L}$ is an oriented cycle of Γ, $\langle \mathcal{L} \rangle = \mathcal{L}$, and $d(x_0, \mathcal{L}) > 1$
choose $x_\infty \in V(\Gamma)$ with $d(x_\infty, \mathcal{L}) > 2|\mathcal{L}|^2/\pi^2$
return $\mathcal{I} \Leftarrow \mathsf{IndexVertexLoop}(x_0, \mathcal{L}, \Gamma) - \mathsf{IndexVertexLoop}(x_\infty, \mathcal{L}, \Gamma)$

3.3 Proofs

Lemma 2. *If $\mathcal{L}$ is a cycle in Γ with $\langle \mathcal{L} \rangle = \mathcal{L}$ and $x \in V(\Gamma)$ with $d(x, \mathcal{L}) > 2|\mathcal{L}|^2/\pi^2$, then x is not in the region of $\mathbb{R}^2$ bounded by $\overline{\mathcal{L}}$.*

Proof. The Isoperimetric Inequality says that the area A enclosed by the simple closed curve $\overline{\mathcal{L}}$ in $\mathbb{R}^2$ is bounded above by $1/(4\pi)$ times the square of the perimeter of $\overline{\mathcal{L}}$. This perimeter is bounded above by $|\mathcal{L}|$. Let $\mathcal{P}$ be a chord-free path. By placing a ball of radius $\frac{1}{2}$ about every other vertex of $\mathcal{P}$, one obtains disjoint balls of total area $\frac{1}{8}\pi|\mathcal{P}|$. Such a path $\mathcal{P}$ from x to $\mathcal{L}$ of length at least $2|\mathcal{L}|^2/\pi^2$ violates the area constraint: the endpoint is thus not surrounded by $\overline{\mathcal{L}}$.

Better constants can be estimated with a longer proof; however, the lower bound must be quadratic in $|\mathcal{L}|$, since a chord-free path in the interior of $\overline{\mathcal{L}}$ can fill up the area bound by $\overline{\mathcal{L}}$, which is quadratic in the perimeter. The following lemmas are critical for turning local cyclic orientation data into global cyclic orientation data. These can be proved by direct enumeration, but an approach which is both more elegant and more easily generalized is to use simple algebraic topology (homology theory).

Lemma 3. *Consider a graph Δ having one oriented cycle (x_1, x_2, x_3), with each inner node x_i connected to an outer node y_i. If this graph has planar*

image $\overline{\Delta}$ *as in Fig. 4, then the cyclic orientation of the outer nodes* $(y_i)_1^3$ *with respect to any point in their convex hull is equal to*

$$\prod_{i=1}^{3} \mathcal{I}_{x_i}(y_i; x_{i-1}, x_{i+1}) \tag{2}$$

under the convention -1*=CW and* $+1$*=CCW, and where the subscript indices are cyclic (computed mod 3).*

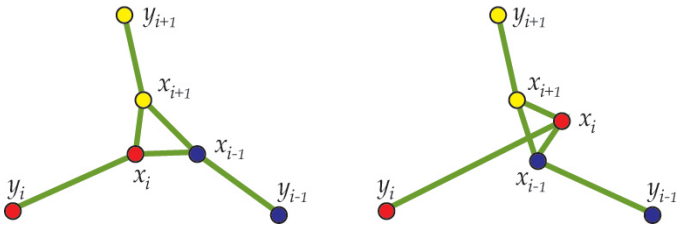

Fig. 4. The cyclic orientation of the outer nodes can be derived from the indices of the inner nodes in the above cases by computing the product of indices.

Lemma 4. *Consider a graph* Y *having one central node* x_0 *attached to an ordered triple of non-colinear points* $(y_i)_1^3$*. Then, for any* i*,* $\mathcal{I}_{x_0}(y_i; y_{i-1}, y_{i+1})$ *is equal to the cyclic orientation of the ordered triple* $(y_j)_1^3$ *in* $\mathbb{R}^2$ *about* x_0*, under the convention* -1*=CW and* $+1$*=CCW.*

Theorem 1. *In any network satisfying Assumptions* **P**, **N**, *and* **O**, *let* $\mathcal{L}$ *be a cycle satisfying* $\langle \mathcal{L} \rangle = \mathcal{L}$ *and* x_0 *a node with* $d(x_0, \mathcal{L}) > 1$*. Algorithm* IndexCheck *returns* $\mathcal{I} = 0$ *iff the winding number of* $\overline{\mathcal{L}}$ *about the node* $x_0 \in \mathbb{R}^2$ *vanishes.*

Proof. Corollary 1 implies that $\overline{\mathcal{L}}$ is embedded in $\mathbb{R}^2$. This simple closed curve separates the plane in two connected components, thanks to the Jordan Curve Theorem. Fixing an orientation on $\mathcal{L}$ induces an (unknown) orientation on $\overline{\mathcal{L}}$.

Choose a chord-free path $\mathcal{P}_0 = \{x_i\}_0^n$ from x_0 to x_n with $d(x_i, \mathcal{L}) = 1$ iff $i = n$. Via Corollary 1, the image of this path, $\overline{\mathcal{P}}_0$, is simple and the restriction of this path to the subpath between nodes x_0 and x_{n-1} lies entirely on one side of $\overline{\mathcal{L}}$ in $\mathbb{R}^2$. The edge from x_{n-1} to x_n may or may not cross $\overline{\mathcal{L}}$.

If there do not exist consecutive cycle nodes ℓ_j, ℓ_{j+1} incident to x_n, then choose any ℓ_j incident to x_n. In this case, the 'Y' graph connecting ℓ_j to x_n, ℓ_{j-1}, and ℓ_{j+1} has no additional connections between outer nodes, and the index $\mathcal{I}_{\ell_j}(x_n; \ell_{j-1}, \ell_{j+1})$ shows on which side of $\overline{\mathcal{L}}$ the node x_n (hence x_0) lies.

If, however, consecutive cycle neighbors exist, one argues that the subgraph Δ consisting of the cycle $(x_n, \ell_j, \ell_{j+1})$ and the connections of these

inner nodes to respective outer nodes $(x_{n-1}, \ell_{j-1}, \ell_{j+2})$ has image $\overline{\Delta}$ as in Fig. 4. A more complicated embedding cannot appear thanks to repeated application of Lemma 1. Thanks to Lemma 3, the product of the three indices $\mathcal{I}_{\ell_j}(\ell_{j-1}; \ell_{j+1}, x_n)$, $\mathcal{I}_{\ell_{j+1}}(\ell_{j+2}; x_n, \ell_j)$, and $\mathcal{I}_{\ell_{x_n}}(x_{n-1}; \ell_j, \ell_{j+1})$ gives the cyclic orientation of the ordered triple $(x_{n-1}, \ell_{j+2}, \ell_{j-1})$ of outer nodes. Via Lemma 4, this tells whether x_{n-1} (and thus x_0) lies to the 'left' or to the 'right' of the embedded segment $(\ell_i)_{j-1}^{j+2}$ of $\overline{\mathcal{L}}$. However, it is possible that $\overline{\mathcal{L}}$ doubles back and crosses the segment between x_{n-1} and x_n, as in Fig. 3[right]. In this case, one needs to be sure to use the subgraph Δ generated by consecutive nodes $(\ell_i)_{j-1}^{j+2}$ where, from the vantage of x_n, $\ell_j \lhd x_{n-1} \lhd \ell_{j+1}$ or $\ell_{j+1} \lhd x_{n-1} \lhd \ell_j$, and no other ℓ_i separates.

There is an ambiguity in $\mathcal{I}$ resulting from the fact that we do not know if the orientation on $\overline{\mathcal{L}}$ is clockwise or counterclockwise; thus we do not know which sign for $\mathcal{I}$ (i.e., the 'left' or the 'right' side of $\overline{\mathcal{L}}$) corresponds to the bounded component of $\mathbb{R}^2 - \overline{\mathcal{L}}$. To determine this, choose a node x_∞ with $d(x_\infty, \mathcal{L}) > 2|\mathcal{L}|^2/\pi^2$. From Lemma 2, x_∞ lies within the unbounded component of $\mathbb{R}^2 - \overline{\mathcal{L}}$. That one can choose such a node and a chord-free path $\mathcal{P}_\infty$ from $\mathcal{L}$ to x_∞ is possible thanks to Assumption **N**. Computing the index of x_∞ with respect to $\mathcal{L}$ and comparing it to that of x_0 as in IndexCheck determines whether x_0 and x_∞ are on the same or different sides of $\overline{\mathcal{L}}$.

4 Isolation

We consider the isolation problem for a network satisfying Assumptions **P**, **N**, and **O**. Given a node x_0, determine whether there exists a cycle $\mathcal{L}$ which surrounds x_0 and construct one if it exists. We restrict the location of the cycle we search for by specifying a lower R_α and upper R_ω bound on the hop distance to x_0. We search for surrounding cycles within the subgraph whose vertices satisfy both bounds.

4.1 Algorithm

The first algorithm we give to solve this problem is similar is spirit to the BoundHole algorithm of [6], in that it relies on angular ordering to perform a depth-first search with constraints. The algorithm of [6] was intended to find holes in a network at a known boundary (or 'stuck') point assuming known exact pairwise angles between neighbors.

To solve the Isolation problem, we choose a chord-free path $\mathcal{P}$ in Γ from x_0 to some terminal point x_∞ which is more than R_ω hops from x_0. Truncate the graph Γ to Γ', the subgraph generated by nodes within R_α and R_ω hops of x_0. The path $\mathcal{P}$ restricts to a path $\mathcal{P}' = \{p_i\}_1^N$ in Γ'.

Beginning with the first node p_1 of $\mathcal{P}'$, construct a path $\mathcal{L}$ by performing a depth-first search of Γ' with the following conditions (Algorithm SweepCycle).

1. The depth-first search augments the path $\mathcal{L}$ by choosing the next available node of Γ' which is clockwise (CW) from the prior edge of $\mathcal{L}$.
2. The search backtracks whenever there is no available CW node, or when the path $\mathcal{L}$ has endpoint in the 1-hop neighborhood of $\mathcal{P}$ approaching from the 'right' side.
3. If the search backtracks all the way to the starting point of $\mathcal{L}$, this starting point is changed to be the next point on $\mathcal{P}$.
4. If the path $\mathcal{L}$ has endpoint in the 1-hop neighborhood of $\mathcal{P}$ on the 'left' side, then $\mathcal{L}$ is completed to a cycle in Γ' by connecting the ends along $\mathcal{P}$.

The justification for this algorithm is that any cycle in Γ' whose image in $\mathbb{R}^2$ has winding number ± 1 about x_0 must intersect $\overline{\mathcal{P}}'$ (since the path is chord-free and thus embedded). Thus, constructing any path which approaches $\mathcal{P}'$ from each side once is automatically a path which encircles x_0.

Algorithm 3 $\mathcal{L} = \mathsf{SweepCycle}(x_0, R_\alpha, R_\omega, \Gamma)$

Require: Γ satisfies **P**, **N**, and **O**
Require: $R_\omega > R_\alpha > 1$
let $\mathcal{P}$ be a chord-free path from x_0 to x_∞ with $d(x_0, x_\infty) > R_\omega$
truncate $\mathcal{P} \to \mathcal{P}' = (p_i)$, $\Gamma \to \Gamma'$ with distance to x_0 between R_α and R_ω
split 1-hop neighborhood of $\mathcal{P}'$ into two sides $\mathcal{P}'_+$, $\mathcal{P}'_-$ using orientation data
while (ℓ_j) has endpoint not in $\mathcal{P}'_-$ **do**
  while (ℓ_j) does not have endpoint in $\mathcal{P}'_+$ and interior point not in $\mathcal{P}'_+$ **do**
    augment (ℓ_j) via CW depth-first search of $\Gamma' - (\mathcal{P}' \cup \mathcal{P}'_+)$
  end while
  $\ell_1 \Leftarrow p_i$, the next available node of $\mathcal{P}'$
end while
return $\mathcal{L} \Leftarrow \emptyset$ if search is exhausted, else return $\mathcal{L} = (\ell_j)$ union segment of $\mathcal{P}'$ connecting ends of (ℓ_j)

4.2 Proofs

Theorem 2. *For any system satisfying Assumptions* **P**, **N**, *and* **O**, *Algorithm* SweepCycle *returns a cycle* $\mathcal{L}$ *in* Γ' *whose image* $\overline{\mathcal{L}}$ *encloses* x_0 *in* $\mathbb{R}^2$ *if and only if such a cycle exists.*

Proof. The image of the truncated graph Γ' lies in a topological annulus $A \subset \mathbb{R}^2$, whose boundary components are connected by the embedded path $\overline{\mathcal{P}}$. Any simple closed curve in A which consists of a segment of $\overline{\mathcal{P}}$ and a segment in $A - \overline{\mathcal{P}}$ which approaches $\overline{\mathcal{P}}$ from both sides surrounds x_0 in $\mathbb{R}^2$ (this is proved using, e.g., homology theory). Thus, if Algorithm SweepCycle returns a non-empty cycle $\mathcal{L}$, then its image $\overline{\mathcal{L}}$ surrounds x_0.

The Jordan Curve Theorem applied to A implies that any simple closed curve in A which surrounds x_0 must intersect $\overline{\mathcal{P}}$. If $\mathcal{L}$ is any cycle of Γ'

whose image surrounds x_0, then there are at least two nodes of $\mathcal{L}$ within the 1-hop neighborhood of $\mathcal{P}'$, thanks to Lemma 1. Choose two such nodes on opposite sides of $\mathcal{P}'$ and such that no further nodes of $\mathcal{L}$ are within the 1-hop neighborhood of $\mathcal{P}'$ In a search of Γ', Algorithm SweepCycle will eventually hit one of these two nodes. The depth first search the algorithm performs cannot exhaust Γ' without sweeping through $\mathcal{L}$.

5 Angles and uncertainty

As a step toward removing angular orientation data, we modify Assumption **O** to account for uncertainty in angular orientation types.

U (Ordering type with Uncertainty) Each node can determine the clockwise cyclic ordering of any neighbors separated by angles of at least α_0.

That is, any triple of neighbors can be cyclically ordered if none of the three pairwise angles is below the threshold α_0. We do not assume that the cyclic ordering measurement always fails whenever an angle is below α_0, but rather that the reading either returns a true angular reading or an empty (i.e., uncertain) reading.

The surprising fact is that for α_0 as large as $\pi/3$, it is often possible to rigorously determine winding numbers. Some choices of Γ and $\mathcal{L}$ present too much uncertainty, but criteria for knowing when you can compute winding numbers are possible. We outline this procedure in the setting of the Separation Problem as an example.

Consider a network satisfying Assumptions **P**, **N**, and **U**, with α_0. Let $\mathcal{L}$ be a cycle satisfying $\langle \mathcal{L} \rangle = \mathcal{L}$ and x_0 a node with $d(x_0, \mathcal{L}) > 1$. In the simplest case where x_n is not within 1-hop of a consecutive pair of cycle nodes, choose any isolated incident cycle node ℓ_j of $\mathcal{L}$. That $\mathcal{L}$ is chord-free implies all three angles at ℓ_i to x_n, ℓ_{i-1}, and ℓ_{i+1} are greater than $\pi/3$ and thus α_0. Therefore $\mathcal{I}$ is well-defined here and yields winding information.

In the more complicated case where there is a subgraph of the form in Fig. 4, then some of the indices at the three inner nodes may be undefined. Since the angles of a triangle sum to π, at least one angle in the interior triangle is no less that $\pi/3$. We consider cases based on how many of these three interior nodes admit a well-defined cyclic orientation of neighbors.

Case 1: If all three indices exist, we are obviously done.

Case 2: If only one index exists, we claim that this single index is equal to the full index of $\mathcal{P}$ with respect to $\mathcal{L}$. This breaks into two cases, according to Fig. 4. In the case on the left, each vertex has the same index, and choosing any one yields the same as their product. In the case on the right, if only one index exists, a brief argument involving plane geometry shows that the angle out of which the bisector of the inner triangle emanates is the largest of the three angles; thus, if only one index is well-defined, it is this one. This node always has index equal to the product of the three inner node indices.

Case 3: In the case where only two indices exist, they either have the same sign or different sign. If they have the same sign, then, using the 'largest angle' result of Case 2, we know that Δ has no self-intersections in $\mathbb{R}^2$. Thus, the index of this path with respect to $\mathcal{L}$ is equal to the index of either of the two well-defined nodes.

Case 4: If two indices only are defined and the two computed indices differ, then we are certainly in the case where Δ is not embedded in the plane. However, if it is not possible to compute the third index and it is not possible to determine which of the two nodes has the larger subtended angle, then there is no information by which the index of this path can be determined. One may attempt to modify either $\mathcal{P}$ or $\mathcal{L}$ locally to remove the ambiguity, but there is no guarantee that this is possible for every Γ.

6 Isolation without cyclic orientation data

For systems which do not satisfy Assumption **O**, no solutions are possible which apply to arbitrary networks: it is easy to generate examples of very sparse graphs which can be embedded in the plane as a unit disc graph in multiple ways. However, there are non-complete algorithms which, upon successful termination, return rigorous winding number information. The "flower" graphs of [8] provide one example of rigorous containment certificate. We briefly present a different approach which uses a modification of Assumption **P** as follows.

P' There is a simply-connected domain $\mathcal{D} \subset \mathbb{R}^2$ which partitions the nodes by membership in $\mathcal{D}$ and yields an 'interior' graph Γ^o of all nodes in $\mathcal{D}$ which satisfies $\overline{\Gamma^o} \subset \mathcal{D}$.

It is not necessary to know the precise geometry of $\mathcal{D}$ (cf. [3]). Algorithm TriPath performs the following operations. From x_0, choose three chord-free paths $\{\mathcal{P}_i\}_1^3$ from x_0 to the 'exterior' graph $\Gamma - \Gamma^o$ such that the 1-hop neighborhood of each $\mathcal{P}_i$ is disjoint from $\mathcal{P}_{i-1} \cup \mathcal{P}_{i+1}$ outside an R_α-hop neighborhood of x_0. (The existence of such paths is of course not guaranteed for all networks.) The algorithm searches within Γ^o for a sequence of arcs connecting $\mathcal{P}_i$ to $\mathcal{P}_{i+1}$ avoiding $\mathcal{P}_{i-1}$ for each i. Chaining these arcs together yields a cycle in Γ^o: see Fig. 6[left].

A proof analogous to that of Theorem 2 implies that this cycle surrounds x_0 in $\mathbb{R}^2$. Assumption **P'** implies that there is a topological annulus $A \subset \mathbb{R}^2$ whose outer boundary is $\partial\mathcal{D}$ and whose inner boundary is a simple closed curve surrounding x_0. The three paths $\{\overline{\mathcal{P}}_i\}$ intersect A in three pairwise-disjoint arcs, each connecting the inner boundary of A to the outer boundary of A. A simple homological argument reveals that any loop in A whose intersections with the $\overline{\mathcal{P}}_i$ are cyclically ordered must surround the inner hole of A and thus surround x_0.

We repeat that should Algorithm TriPath fail to construct a surrounding cycle, it does not indicate the non-existence of such a cycle.

Algorithm 4 $\mathcal{L} = \mathsf{TriPath}(x_0, R_\alpha, \Gamma, \Gamma^o)$

Require: Γ satisfies **P'** and **N**
Require: $x_0 \in \Gamma^o$
1: search for chord-free paths $\{\mathcal{P}_i\}_1^3$ in Γ from x_0 to $\Gamma - \Gamma^o$ with $d(\mathcal{P}_i, \mathcal{P}_{j\neq i}) > 1$ outside an R_α neighborhood of x_0.
2: search for paths $\{\mathcal{L}_i\}_1^3$ in Γ^o from $\mathcal{P}_i$ to $\mathcal{P}_{i+1}$ with $d(\mathcal{L}_i, \mathcal{P}_{i-1}) > 1$.
3: **if** either search fails **then**
4: $\quad \mathcal{L} \Leftarrow \emptyset$
5: **else**
6: $\quad \mathcal{L} \Leftarrow$cycle in $\cup_i \mathcal{P}_i \cup_i \mathcal{L}_i$
7: **end if**
8: return $\mathcal{L}$

7 Simulations

Algorithm IndexCheck is easily implemented, However, as it produces no output except for a winding number that is easily seen when the graph is illustrated, we waste no space illustrating test runs of this algorithm.

Algorithm SweepCycle has been implemented [in C] for randomly generated sets of nodes with node density, input node, and the radius R_α as user-defined parameters. Examples of networks which are relatively dense and sparse are illustrated in Fig. 5. This algorithm inherits the time- and space-complexity of a depth-first search on the truncated graph.

Algorithm Tripath has likewise been implemented [in Java] with randomly generated sets of nodes. Fig. 6[left] displays a typical output, with the three paths $(\mathcal{P}_i)_1^3$ in bold and the cycle $\mathcal{L}$ marked. Depending on the exact form of Γ, it may be impossible to find three such paths $(\mathcal{P}_i)_1^3$ which are properly separated. Fig. 6[right] illustrates such an example, and reinforces the result that this algorithm does not always find a surrounding cycle. This algorithm is distributed and local: both the operation of finding $(\mathcal{P}_i)$ and the connecting segments between them are distributed in this software.

8 Concluding remarks

The challenge of localization in an unknown environment is significant across many areas of robotics and sensor networks, and has generated an impressive array of techniques and perspectives. This paper demonstrates that localization is not a prerequisite to solving problems about the geometry and topology

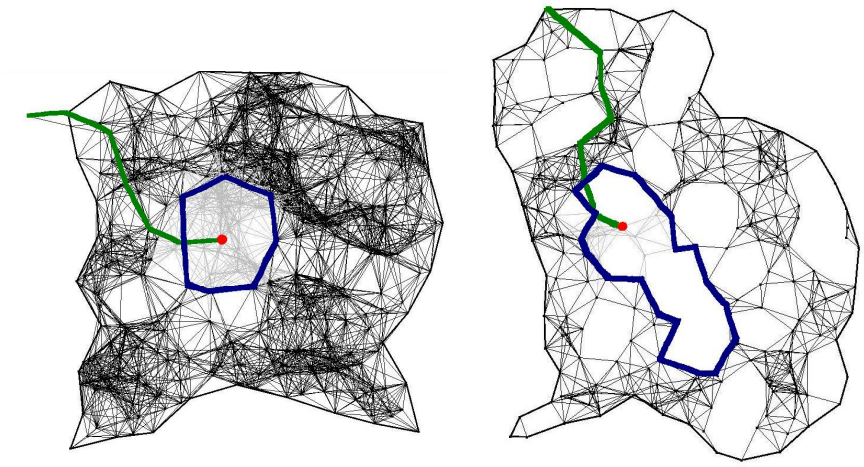

Fig. 5. Results of Algorithm SweepCycle on a dense [left] and sparse [right] randomly generated network. In both cases, the algorithm successfully produces a chord-free surrounding cycle outside of the 2-hop neighborhood of the encircled node.

of cycles in a planar network. For many systems, the unit disc graph possesses sufficient information to find separating cycles about a node. We also demonstrate that, in addition to the unit disc graph, an angular ordering of neighbors suffices to solve winding number problems for all possible networks. Absolute angles are not needed, and large uncertainty in the angular ordering data may be tolerated. As with many problems in manipulation, localization, mapping, etc., the amount of sensory information needed to solve the problem is sometimes far below what one would expect.

References

1. H. Breu and D. G. Kirkpatrick. "Unit Disk Graph Recognition is NP-hard," *Comput. Geom. Theory Appl.*, 9(1-2):3–24, 1998.
2. J. Bruck, J. Gao, and A. Jiang, "Localization and routing in sensor networks by local angle information," in Proc. of the Sixth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'05), 181-192, 2005.
3. V. de Silva and R. Ghrist, "Coordinate-free coverage in sensor networks with controlled boundaries via homology," submitted for publication.
4. V. de Silva and R. Ghrist, "Coverage in sensor networks via persistent homology," accepted for publication, *Alg. & Geom. Topology*, preprint.
5. V. de Silva, R. Ghrist, and A. Muhammad, "Blind swarms for coverage in 2-d," in Proc. Robotics: Systems & Science, 2005.

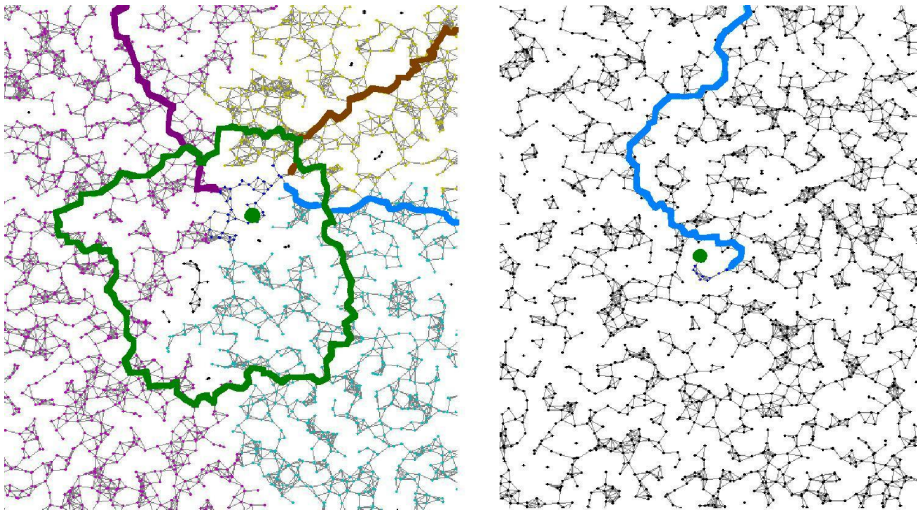

Fig. 6. Examples of Algorithm TriPath applied to a sparse randomly generated network: [left] a network for which TriPath is successful. Three disjoint paths exist to the 'exterior' of the domain pictured, and a loop which cyclically connects these paths must generate a surrounding cycle. [right] The algorithm fails to find a surrounding cycle about the highlighted node, even though a surrounding cycle exists. The algorithm cannot find three disjoint paths from the node to the boundary.

6. Q. Fang and J. Gao and L. Guibas, "Locating and Bypassing Routing Holes in Sensor Networks," in Proc. 23rd Conference of the IEEE Communications Society (InfoCom), 2004.
7. Q. Fang, J. Gao, L. J. Guibas, Vin de Silva, Li Zhang, "GLIDER: Gradient landmark-based distributed routing for sensor networks," in IEEE INFOCOM'05, March, 2005.
8. S. Fekete, A. Kröller, D. Pfisterer, and S. Fischer, "Deterministic boundary recognition and topology extraction for large sensor networks," in *Algorithmic Aspects of Large and Complex Networks*, 2006.
9. V. Guillemin and A. Pollack. *Differential Topology.* Prentice Hall, 1974.
10. A. Jadbabaie, "On geographic routing without location information," in Proc. IEEE Conf. on Decision and Control, 2004.
11. 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.
12. S. Poduri, S. Pattem, B. Krishnamachari, and G. Sukhatme, "A unifying framework for tunable topology control in sensor networks," USC CRES Technical Report *CRES-05-004*, 2005.
13. A. Rao, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic Routing without Location Information," in Proceedings of 9th Annual International Conference on Mobile Computing and Networking (Mobicom'03), September 2003.