Convex Partitioning of Large-Scale Sensor Networks in Complex Fields: Algorithms and Applications

GUANG TAN, SIAT, Chinese Academy of Sciences
HONGBO JIANG, Huazhong University of Science and Technology
JUN LIU, Huazhong University of Science and Technology
ANNE-MARIE KERMARREC, INRIA/IRISA

When a sensor network grows large, or when its topology becomes complex (e.g., containing many holes), network algorithms designed with a smaller or simpler setting in mind may be rendered rather inefficient. We propose to address this problem using a divide and conquer approach: the network is divided into convex pieces by a distributed *convex partitioning* protocol, using connectivity information only. A convex network partition exhibits some desirable properties that allow traditional algorithms to work to their full advantage. Based on this, we can achieve relatively high performance for an algorithm by combining algorithmic actions within individual partitions. We consider two important applications: virtual-coordinate-based geographic routing and connectivity-based localization. The former benefits from convex partition's friendliness to network embedding, which is crucial to generating accurate virtual coordinates for the nodes, while the latter leverages the fact that shortest paths are largely straight for node pairs within a convex partition. Experimental results show that the convex partition approach can significantly improve the performance of both applications in comparison with state-of-the-art solutions.

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

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Sensor networks, convex partition, geographic routing, localization

ACM Reference Format:
Guang Tan, Hongbo Jiang, Jun Liu, and Anne-Marie Kermarrec. 2014. Convex partitioning of large-scale sensor networks in complex fields: Algorithms and applications. ACM Trans. Sensor Netw. 10, 3, Article 41 (April 2014), 23 pages.
DOI: http://dx.doi.org/10.1145/2594772

G. Tan's work was supported by the NSFC under Grant 61103243, Youth Innovation Promotion Association, Chinese Academy of Sciences, the Ministry of Science and Technology 863 Key Project No. 2011AA010500, by the Scientific Research Foundation for the Returned Overseas Chinese Scholars (State Education Ministry), and Shenzhen Overseas High-level Talents Innovation and Entrepreneurship Funds KQC201109050097A. H. Jiang's work was supported in part by the National Natural Science Foundation of China under Grant 61073147, Grant 61173120, Grant 61103243, Grant 61202460, Grant 61271226, and Grant 61272410; by the National Natural Science Foundation of China and Microsoft Research Asia under Grant 60933012; by the Fundamental Research Funds for the Central Universities under Grant 2012QN078; by the CHUTIAN scholar Project of Hubei Province; by the Scientific Research Foundation for the Returned Overseas Chinese Scholars (State Education Ministry); by the National Natural Science Foundation of Hubei Province under Grant 2011CDB044; by the Fok Ying Tung Education Foundation under Grant 132036; by the Hong Kong Scholars Program under Grant XJ2012019; and by the Program for New Century Excellent Talents in University under Grant NCET-10-408 (State Education Ministry).
Authors' addresses: G. Tan, SIAT, Chinese Academy of Sciences, China; H. Jiang (corresponding author) and J. Liu, Department of Electronics and Information Engineering, Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan, China 430074; A. Kermarrec, INRIA/IRISA, Rennes France; Correspondence email: hongbojiang2004@gmail.com.

DOI: http://dx.doi.org/10.1145/2594772

1. INTRODUCTION

A sensor network's topology usually reflects the surrounding physical environment. As the network grows large, more environmental features will have to be covered, thus the network topology can become quite complex. This complexity can bring significant challenges to some existing algorithms. For example, geographic routing algorithms greedily forward packets by default, with the implicit assumption that the sought route should approximately follow a straight line. This assumption well holds in a regular network field, which leads to excellent routing performance [Karp and Kung 2000]. However, in reality, a sensor field may be irregular, possibly containing obstacles/holes of arbitrary shape. The holes or concave corners present in the network can seriously disrupt the straight-line course of greedy forwarding, thus leading to very poor performance [Kuhn et al. 2003, 2002].

Another application that often assumes straight-line shortest paths is connectivity based localization. In this problem, the physical distance between two nodes needs to be estimated with their shortest path hop count. When the network topology is highly irregular, the shortest path length may be bent, thus the measured path length can grossly overestimate the Euclidean distance between the end nodes. It has been shown that this issue can cause unacceptable errors to localization algorithms [Li and Liu 2007; Lederer et al. 2008; Wang et al. 2009].

In view of the negative impact of network topology irregularity, researchers have proposed numerous methods to improve the considered algorithm's performance or extend the algorithm's applicability. These efforts are mostly application specific, and normally add to the solution's complexity significantly.

In this article, we consider an alternative approach to tackling the topology complexity problem. Rather than modifying the core of particular algorithms to adapt them to complex environments, we divide the network into convex pieces. Because of its geometric simplicity, a convex network piece presents a friendly environment for many algorithms that involve geometric elements (e.g., Euclidean distance). In a convex partition, those algorithms can easily work to their advantage. Based on this, we introduce some extra, usually simple, application specific design to combine algorithmic actions in individual partitions.

The division of networks is accomplished by a distributed *convex partitioning* protocol, which assumes only the network's connectivity information. That is, the protocol only needs every node to know which neighboring nodes are connected to itself and the IDs of those neighbors. The output of the protocol is that every node is assigned a partition ID, indicating the partition it belongs.

We show how convex partition benefits the two applications mentioned earlier. For virtual-coordinate-based geographic routing, we show through theoretic analysis and experiments why representative algorithms, in particular NoGeo [Rao et al. 2003] and BVR [Fonseca et al. 2005], perform poorly in topologically complex networks. We then design a new routing protocol, called *CONVEX*, using our convex partitions. Experiments show that our approach significantly improves the routing performance, or achieves the same performance with much reduced overheads, in comparison with representative geometric routing algorithms.

For connectivity-based localization, we show that convex partitioning of a topologically complex network enables a straightforward way to localize network nodes. The localization quality is comparable with the state of the art, with reduced message cost.

The remainder of this article is structured as follows. Section 2 introduces related work; Section 3 describes the convex partition protocol in detail; Section 4 presents a convex partition based geographic routing algorithm and examines its performance;

Section 5 presents a localization scheme that uses convex partitions, and Section 6 concludes the article.

2. RELATED WORK

We briefly review the related work of sensor network partition, and then provide some background of the two applications we will discuss.

2.1. Network and Polygonal Partitioning

There have been only a few pieces of prior work that considered partitioning of sensor networks. An often used technique is Voronoi partitioning. In Fang et al. [2005], an algorithm called *GLIDER* helps to divide the network into *tiles* – regions where the node placement is relatively regular so that local greedy methods can work well. The partitioning relies on the selection of a set of landmarks, which are assumed to be done manually or at random. The former approach may be too cumbersome for a large network, while the latter exhibits several drawbacks as will be shown later. GLIDER provides much inspiration to our work, though it does not focus on convexity, which makes the generated tiles unsuitable for global network localization. Voronoi partitioning is also used in Lederer et al. [2008] and Wang et al. [2009] for localization. Zhu et al. [2008] propose to segment an irregular sensor field into nice-shaped pieces. This scheme does not have a notion of convexity.

In the context of computational geometry and computer vision, convex decomposition of a polygon has been well studied [Sack and Urrutia 2000; Ren et al. 2011; Lien and Amato 2004; Lu et al. 2012]. It is shown [Lingas 1982] that computing a minimum number of convex components for a polygon with holes is NP-hard. While a minimum number of convex partitions is desirable in our context, our primary goal in this article, is a simple and practical protocol that can be implemented in a distributed way.

2.2. Virtual-Coordinate-Based Geographic Routing

Virtual-coordinate-based geographic routing does not require every node be equipped with a GPS device, so significantly expands the use of geographic routing. Rao et al. [2003] first propose the NoGeo family of virtual coordinate assignment algorithms. In NoGeo, perimeter (boundary) nodes use a triangulation algorithm to compute their coordinates. These coordinates are projected onto a virtual circle, and other nodes then determine their virtual coordinates through an iterative relaxation procedure. NoGeo shows a very high greedy forwarding success rate (GFSR) – the success probability of a route being found using solely greedy forwarding – in a regular field, and is also robust to field irregularity to some degree. NoGeo represents an iterative approach for generating the virtual coordinates, respecting the connectivity relations between nodes. In NoGeo routing, the message is forwarded greedily and upon reaching a dead-end, an expanding ring search technique is used to find a node that can make progress.

GSpring [Leong et al. 2007] is another iterative virtual coordinate assignment scheme that uses a modified spring relaxation algorithm to incrementally increase the convexity of voids in the network. Arad and Shavitt [2006], present a Node Elevation Ad-hoc Routing (NEAR) protocol to address the concave voids problem. Both GSpring and NEAR try to increase the convexity of voids in a best-effort way, thus do not guarantee complete elimination of local minima areas in geographic routing.

The Beacon Vector Routing (BVR) protocol, by Fonseca et al. [2005], represents another approach of virtual coordinate generation. In BVR, nodes' coordinates are assigned in reference to a set of preselected landmark nodes (beacons). Every node is assigned a distance vector $\langle d_0, d_1, \ldots, d_m \rangle$, where d_i is its hop distance to the ith landmark. Routing is done by minimizing a distance function of these coordinates. When encountering a dead-end, the algorithm performs a scoped flooding to guarantee

delivery. Very similar techniques include Cao and Abdelzaher [2004], Zhao et al. [2005], and Caruso et al. [2005], with slight differences in the definition of distance function and dead-end overcoming strategies. The GLIDER protocol [Fang et al. 2005] also uses landmarks but with a different global data structure (rather than a global shortest-path tree rooted at each landmark). The problem of selecting a good set of landmarks is addressed in Nguyeny et al. [2008].

Instead of assigning virtual coordinates based only on hop distances, GEM [Newsome and Song 2003] builds a polar coordinates system, in which every node is assigned an *angle* range in addition to its hop distance to a root node. The presence of the angle range helps to deliver message precisely to its destination, but on the other hand incurs significant overheads under network reconfiguration. In Bruck et al. [2005], a medial axis graph reflecting the global field topology is first established for global routing, then a naming scheme similar to GEM is used in local routing.

2.3. Connectivity-Based and Anchor-Free Localization

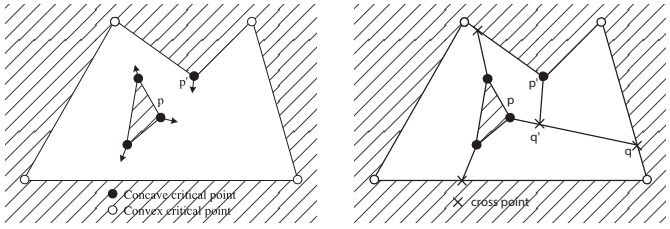
Connectivity-based and anchor-free localization aims to produce a relative coordinate system for a network without using ranging techniques (for measuring physical distance) or anchor nodes with known positions (to serve as a reference). The localized nodes should form a shape similar to the original, with difference only in size and orientation. This kind of localization makes very relaxed assumptions about network configuration, thus is appealing to system depolyment.

A well-known approach to connectivity-based and anchor-free localization is based on Multidimensional Scaling (MDS) techniques [Shang et al. 2003]. This method takes an inter-node hop distance matrix as input, and generates a set of relative coordinates for each nodes in a centralized way. In the presence of holes, it is shown to perform poorly [Lederer et al. 2008]. Shang and Ruml [2004] addresses the hole problem by creating a local map at each node and then merging them to create a global map. It is unclear how this scheme works for large and complex networks since they examined only examples with simple layouts.

Li and Liu [2007] propose a method that uses the geometric characteristics of hole corners to correct the distorted length estimate of a path passing by a hole. The idea is to create a new shortest path by creating a virtual hole around each hole corner. The difference between the original and new shortest paths allows one to obtain the real Euclidean distance between two nodes. Lederer et al. [2008] develop a novel localization method with a emphasis on how to avoid flip ambiguities. In their subsequent work [Wang et al. 2009], they propose an incremental Delaunay refinement method that does not rely on network boundary assumption. The new design is shown to be more robust to different network environments. More recent works on localization include Liu et al. [2012] and Jiang et al. [2012].

3. THE CONVEX PARTITION PROTOCOL

We assume that the network is deployed on a 2D plane; each node p has a unique ID, denoted by $ID(p)$. The network may contain large holes caused by obstacles. There exists a boundary detection algorithm (e.g., Fekete et al. [2004], Funke [2005], Kroller et al. [2006] and Wang et al. [2006]) that detects the nodes on the network boundaries, including outer boundary and hole boundaries. Each boundary has a unique ID and is represented by a directed cycle of nodes. Each node on such a boundary is aware of its upstream and downstream boundary neighbors. A sequence of nodes on a boundary cycle is called a *boundary path*. Considering the uneven node distribution, our aim is to partition the field into approximately convex polygonal subareas, and to let every node know which partition(s) it belongs to.



(a) Drawing bisectors (b) The final partitions

Fig. 1. Convex partition of a polygonal environment. The sensor field is shown as nonshaded area. Both critical points and cross points are vertices of the final partitions.

Definition 3.1 (*r-hop Neighborhood*). The *r-hop neighborhood* of node p, denoted by $N_r(p)$, is the set of nodes at most r hops away from p. The *r-neighborhood size* of node p is $|N_r(p)|$.

Definition 3.2 (*r-hop Interior Node*). An *r-hop interior node* is a node whose nearest boundary node is at least r-hops away.

Let D_r be the average r-hop neighborhood size of all r-hop interior nodes in a network. D_r can be thought of as the area of a full disk of radius r in the continuous domain.

Definition 3.3 (*r-hop Criticality*). The *r-hop criticality* of a boundary node p, $C_r(p) = \frac{|N_r(p)|}{D_r/2}$.

Criticality reflects the shape of a boundary node's neighborhood area. If a node p is located at the middle of a straight boundary line, then its neighborhood area will be approximately a half disk, so $C_r(p)$ will be close to 1. If $C_r(p)$ deviates from 1 significantly, then p is likely to be near the corner of a hole.

Definition 3.4 (*Concave/Convex Critical Point*). Let $\delta_1 > 0$ and $\delta_2 < 1$ be two predefined system parameters. A boundary node p is a *concave critical point* if $C_r(p) > 1 + \delta_1$, or a *convex critical point* if $C_r(p) < 1 - \delta_2$.

Intuitively, if we approximate the field boundaries using polygons, then the critical points are expected to correspond to the polygons' vertices. A concave point is a node where the inward angle (the angle spanning across the sensing area) is greater than π, and a convex point is a node where the inward angle is smaller than π; see Figure 1 for an illustration.

The protocol has three phases:

(1) identifying the critical points;
(2) bisector-induced convex partitioning;
(3) partitions recognition.

All these operations are performed in a distributed way. We assume that there is a base station that initiates each phase by sending a command to the network; after that it is no longer involved and sufficient time is allowed to elapse before the next phase begins. We first describe the protocol in a form without much message optimization. Reducing message complexity will be discussed later.

3.1. Identifying Critical Points

Critical nodes are determined based on their r_0-hop criticality, where r_0 is a small constant system parameter (e.g., 3). r_0 is chosen such that there will be sufficient r_0-

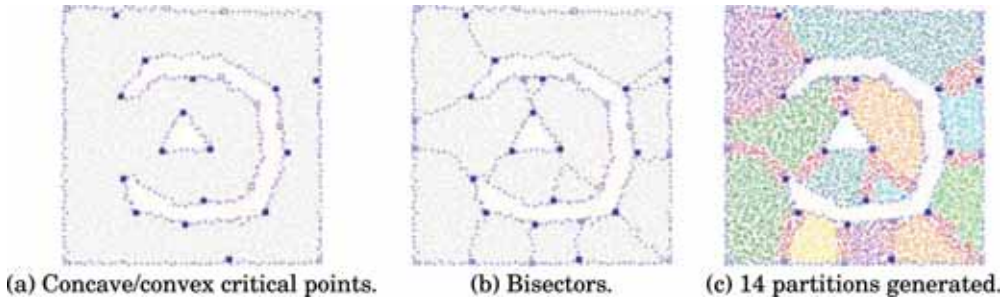


Fig. 2. Illustration for convex partition. The network has 4431 nodes with average node degree 10.22. Concave critical points, convex critical point, and cross points are shown in solid squares, squares and crossed squares, respectively. Boundary/bisector nodes are shown in blue; belt nodes are in red. $r_0 = 5, \delta_1 = 0.2, \delta_2 = 0.1$.

interior nodes present in the network. When no confusion will be caused, we often refer to a node's r_0-hop criticality as its criticality.

The observation of critical nodes is that

(1) if p is a concave critical node, then its criticality should be larger than 1, and also be a local maximum among its r_0-hop neighbors on the same boundary;
(2) if p is a convex critical node, then its criticality should be smaller than 1 and also be a local minimum among p's r_0-hop neighbors on the same boundary;
(3) if a boundary node p is situated in the middle of a sufficiently long (e.g., $>2r_0$) straight-line boundary, then its criticality should be close to 1.

Therefore, to determine whether p is a critical node it needs to calculate $|N_{r_0}(p)|$ and to obtain an estimate of D_{r_0}.

Calculating $|N_{r_0}(p)|$ is simple: the node p does a scoped flooding to its r_0-hop neighborhood, and collects the replies aggregated along the reverse paths. To estimate D_{r_0} a number of interior nodes are first sampled probabilistically across the network. Specifically, each nonboundary node p performs with a small probability an expanding ring search for a nearest boundary node. The search is limited to r_0-hop distance of p. If p finds that its nearest boundary node is on or beyond its r_0-hop ring, it marks itself as an r_0-interior node, and calculates $|N_{r_0}(p)|$. Suppose these tasks take no more than t time, then after t plus the beginning time of this phase, all interior nodes flood their r_0-hop neighborhood size to the network. Because of this synchronization every node only needs to forward one message while being able to receive an r_0-hop neighborhood size announcement. This way, every node will get an estimate of D_{r_0} from its nearby areas.

After p has collected the value $|N_{r_0}(p)|$ and D_{r_0}, it can calculate $C_{r_0}(p)$ and determine whether it is a critical point. Let δ_1 and δ_2 be two system parameters.

(1) If $C_{r_0}(p) > 1 + \delta_1$ and $C_{r_0}(p)$ is a local maximum among p's r_0-hop neighbors on the same boundary, then p is a concave critical point;
(2) if $C_{r_0}(p) < 1 - \delta_2$ and $C_{r_0}(p)$ is a local minimum among p's r_0-hop neighbors on the same boundary, then p is a convex critical point;
(3) otherwise, p is not a critical point.

The parameters δ_1 and δ_2 determine how sensitive this process is to noise. In our experiments, values between $[0.1, 0.2]$ prove to be a good choice for practical purposes. Figure 2(a) shows the result of identifying critical points in a graph. We can see that the identified critical points roughly capture the shape of the holes. Although there are a few false positives, their only effect is to slightly increase the number of

partitions generated. This leads to a small increase of the amount of state information at each node. Our experiments will show that this factor does not affect the significant advantage of our protocol over existing schemes in terms of routing performance and maintenance overhead.

3.2. Bisector-Induced Convex Partition: Principle

To give a clear idea of our convex partition protocol, we first describe it in a continuous domain, which helps us concentrate on the principle; the implementation is deferred to the next section. The sensor field is assumed to be a *polygonal environment*, where the field outer boundary and inner holes are all simple polygons. The partitioning result is a set of convex polygons (partitions).

The key operation of the partition protocol is to draw a bisector of the inward angle at each concave critical point. In a distributed environment, drawing a line can be thought of as moving a point continuously from some origin across the plane in a known direction (Figure 1(a)). We assume such a direction is already determined. Now each concave critical point p sends a point q out of it. On its journey, the moving point q will often hit a line, which is either an existing field/hole boundary edge, or a fully or partially drawn bisector from another origin, where it stops and the line segment $\overline{pq}$ becomes an edge of some new partition. At this point, q becomes a *cross point*. When traveling, q may occasionally "collide with" another moving point q' from some other origin p'. They compare the IDs of their origins – the point from the origin with a larger ID continues to move, while the other stops and creates a new partition edge. The point that keeps on moving will ultimately hit some boundary edge and create another partition edge (Figure 1(b)).

THEOREM 3.5. *All the partitions generated by this protocol are convex polygons.*

PROOF. Assume that there exists a concave partition, whose inward angle at its vertex p is greater than π. Then, p must not be a critical point, because the bisector from p divides the inward angle at p in half, leaving no angular space greater than π around it. p cannot be a cross point either, because the partition protocol requires one of the two bisectors that intersect at p to extend to some edge of the field/hole boundary. Therefore, there cannot be such a vertex p with an inward angle greater than π, and so all partitions must be convex polygons. □

3.3. Bisector-Induced Convex Partition: Implementation

Analogous to drawing a bisector on the 2D plane, in a discrete network we need to identify a path, referred to as a *bisector path*, from a concave critical point p; see Figure 2(b) for an example. The observation is that the bisector nodes should have approximately the same distance from the two adjacent boundary paths of p. More generally, let p_0 and p_1 be two nodes on p's boundary that are r_0 hops away from p in the upstream and downstream direction, respectively, then a node on the bisector path should be approximately equidistant from p_0 and p_1. In our implementation, we do a further simplification: if we can find the endpoint q of such a bisector path, then the whole path is approximated by the shortest path between q and p. Here the endpoint q should be a boundary node, and following its analogy in the continuous domain, is also called a *cross point* after the establishment of this shortest path.

We search for the bisector endpoint q for p in the following way. First, p finds the two nodes p_0 and p_1 as specified previously. It then commands each of p_0 and p_1 to perform a flooding operation. As a flooded message reaches an intermediate node v, v records the parent node from which it receives the message and the number of hops the message has travelled so far. v forwards the message to its neighbors only if v is not a boundary node. If v is on the field/hole boundary, and has received the messages from both p_0 and

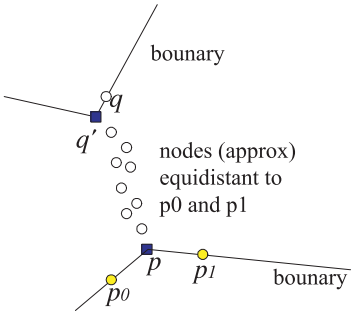


Fig. 3. Avoiding over fragmentation.

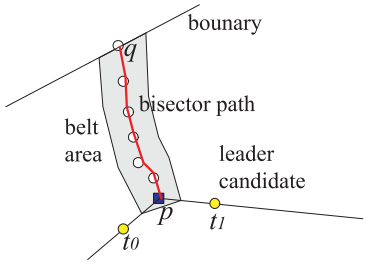


Fig. 4. Selection of leader candidates.

p_1, it compares its hop distances from p_0 and p_1. If the distances are the same or differ by only a small constant (in our implementation one), v sends a reply immediately to p via the parent links; the message will record the IDs of all traversed nodes. In the case of receiving multiple replies, p selects one from a node q that has the smallest average distance to p_0 and p_1. It then sends a confirmation message to the selected node q via the reverse path recorded in q's reply. This path thus is the desired bisector path. While the confirmation message travels, the nodes on the path (not including p and q) mark themselves as *bisector nodes* and also boundary nodes, and record the two endpoints p and q. Note that the bisector nodes are also a type of boundary nodes.

Several strategies are used to avoid generating an overly fragmented field. First, when a boundary node q that is (approximately) equidistant to p_0 and p_1 is found, q does a local two-hop broadcast to search for existing critical point or candidate bisector endpoints nearby. If another boundary node q' is found to be a critical point (see Figure 3), or with a smaller average distance to p_0 and p_1, or with a smaller difference of distance to p_0 and p_1, then q gives up the chance of establishing a bisector path to p. Second, when two existing critical points happen to find each other to be the desired bisector endpoint, the bisector path will be determined by the critical point with a larger ID, but not by both endpoints which often generate incongruent shortest paths.

Due to the asynchronous discovery of bisector paths, some of those paths may cross each other, which is undesirable. To avoid this a concave critical point p sends messages over its outgoing bisector path (p, q) periodically for a certain period of time (until the end of this phase.) While the message travels, the host node x checks whether its one-hop neighborhood contains a bisector node x' from another bisector path (p', q'). If so, it compares the $ID(p) + ID(q)$ and $ID(p') + ID(q')$. If the former is larger, or is equal to the latter but $ID(p) > ID(p')$, then the bisector path between p and q is split: the node x' is taken as p's new bisector endpoint, while the segment of path between q and x (excluding q and x) will be notified to give up its role as part of bisector path; that is, all the nodes on that segment are no longer bisector nodes.

After this phase, the sensor field is divided into a number of partitions, each being an approximately convex polygon. A polygon P has a number of critical and cross points as its vertices, and also a number of edges, called *partition edges*, each being a path between two critical/cross points. We let each partition vertex remember its adjacent partition vertices on the partition graph, and also the length in hops of each partition edge incident on it. Figure 2(c) shows an example of the partition result on a network.

3.4. Partition Recognition

The partition recognition relies on partition-wise flooding to probe the partition edges. Some leader node in a partition then collects the edge information and constructs

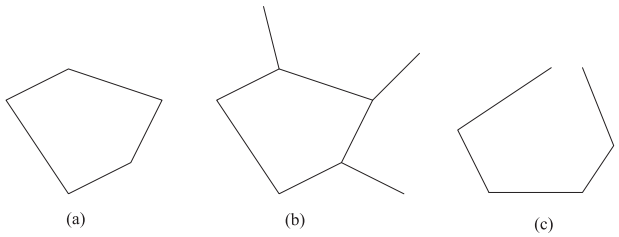


Fig. 5. The three cases in partition construction.

the partition polygon, and finally notifies the nodes of the partition(s) they belong to. The key challenge of this phase is to limit flooding to a certain partition without letting it penetrate to other parts of the network. Although the bisector path provides a natural border line between two partitions, without location information a node near a bisector path can by no means tell on which side it is situated. Our strategy is to construct a dumb belt area near a bisector path that prevents the flooding in a partition from entering another. In our implementation, all the one-hop neighbors of the bisector nodes, not including the bisector nodes themselves, constitute such a belt area. Such nodes are termed *belt nodes*. Recall that we have, so far, divided the nodes into four categories: non-bisector boundary nodes, bisector boundary nodes, belt nodes, and other nodes. We have all the boundary nodes remember their associated partition edge endpoints, and have all the belt nodes remember the endpoints of their bisector paths.

For a partition P, the partition-wide flood is initiated by some nodes near P's critical points. Such critical points only need to be concave critical points, since every partition must have at least one bisector path as one of its polygon edges, thus at least one concave critical point as its polygon vertex. This way no partition will be left out without receiving flood. A concave critical point searches in its neighborhood for two nearest boundary nodes t_0 and t_1 beyond the belt areas (see Figure 4), one in upstream direction and the other in downstream direction, on its field/hole boundary. If such a node exists, then p commands it to start a flood.

Suppose a node t_0 in a partition P performs a flood. The flooded message will be forwarded by all nodes except the belt nodes and bisector nodes. When the flood reaches a belt or bisector node, the message collects the information of those nodes's associated partition edges, including the endpoints' IDs and length in hops, and then travels back to t_0 via the reverse routes. t_0 then collects this information into an edge set $\mathcal{E}$, based on which it tries to construct a polygon. The collected edges may constitute a graph that contains, but is not exactly, a polygon, or may not even be able to form a polygon. There are three possible cases to consider (see Figure 5).

(1) The edges in $\mathcal{E}$ form exactly a cycle (Figure 5(a)). This is the ideal case and the cycle is directly taken as the polygon.
(2) The edges in $\mathcal{E}$ contains one and only one cycle (Figure 5(b)), with a number of additional noncyclic edges. The extra edges come from belt nodes near concave critical points that have many partition edges incident on them. We simply remove those edges and take the cycle as the polygon.
(3) The edges in $\mathcal{E}$ contains no cycle (Figure 5(c)). This can happen in a partition with a very short partition edge, whose belt area is largely covered by the belt areas of adjacent bisector paths. Nodes in such a belt area will thus be unreachable by the flood. This case will result in a number of nodes, referred to as *orphan nodes*, ending up without partition assignment. We address this issue later.

If t_0 has successfully constructed a partition P, it takes itself as the leader of the partition, and performs a new partition-wide flood to notify all the nodes in P of its ID. There can be more than one leaders in one partition; in such a case the leader t_{max} with the highest ID prevails, and the nodes in P only take t_{max} as their leader. For bisector nodes and belt nodes, they belong to two adjacent partitions simultaneously. For orphan nodes, we simply let them search for a nearest node with partition assignment and join the partition(s) of that node. At this point, the partition recognition phase is complete.

3.5. Message Complexity

Since the partitioning protocol is expected to primarily serve as a network preprocessing service, the requirement on completion time is often not stringent. We can therefore reduce message cost by introducing appropriate synchronization.

3.5.1. Boundary Nodes Identification Phase. This phase incurs $O(n)$ message overhead.

3.5.2. Critical Points Identification Phase. Message transmission is needed for two tasks: the r_0-hop interior nodes calculating and announcing D_{r_0}, and boundary nodes calculating their r_0-hop neighborhood sizes. For the first task, the sample interior nodes are selected probabilistically. The probability ρ can be easily chosen such that in expectation, the union of those sample nodes' neighborhoods will be no larger than the whole network size. For example, let $D_{r_0}^{max}$ be the maximum r_0-hop neighborhood size estimated according to the maximum node degree (which is assumed to be known in advance), then an option is $\rho = D_{r_0}^{max}/n$. This will result in $O(n)$ message transmission on average. The announcement process also generates $O(n)$ messages. For the second task, a node near a boundary can buffer the query messages from all its boundary neighbors and forward them at once. Since the query message is very short, a common message can accommodate many such queries, and thus the total number of messages transmitted is roughly r_0 times the total number of boundary nodes. This also requires $O(n)$ messages.

3.5.3. Bisector Path Identification Phase. Every concave critical point needs $O(n)$ messages to establish a bisector path. The number of concave critical points is less than the number of critical points n_{cri}, therefore the total number of message is $O(n \times n_{cri})$, which depends only on the complexity of the field's large topological features, rather than on the network size n, and is often small in real-world applications.

3.5.4. Partition Recognition Phase. Let n_{par} be the number of partitions generated. It is easy to see that $n_{par} \in O(n_{cri})$, therefore this phase incurs $O(n \times n_{cri})$ messages as well.

Putting things together, our partitioning protocol incurs a total of $O(n \times n_{cri})$ messages, where n_{cri} is the number of concave critical points. Since n_{cri} is very small compared with n, we believe this is a reasonable overhead in practice.

4. CONVEX PARTITION BASED GEOGRAPHIC ROUTING

In geographic routing, nodes are identified by their geographic coordinates and routing is done greedily: at each step, a node routes a message to the neighbor that is closest to the destination. When the message reaches a dead-end, that is, has no neighbor closer to the destination, the protocol uses a face-routing strategy to route out of the dead-end, and then resumes the greedy forwarding when appropriate. Such an approach is simple and extremely scalable as every node only needs to remember its immediate neighbors. In a regular field with a relatively high node density, this protocol performs nearly optimally [Karp and Kung 2000].

A main problem with geographic routing is that each node needs to know its geographic location, either directly through a GPS device or indirectly through a

localization service. Equipping each node with a GPS device may be too costly for some applications, while accurate localization service itself involves some technical challenges [Li and Liu 2007]. To address this issue, many protocols have been proposed that use *virtual coordinates* for geographic routing [Rao et al. 2003; Fonseca et al. 2005; Zhao et al. 2005; Cao and Abdelzaher 2004; Tsai et al. 2008; Caruso et al. 2005]. The virtual coordinates do not necessarily correspond to the actual physical locations, but often reflect the connectivity relations between nodes. Typically, the protocol forwards packet greedily using virtual coordinates, sometimes with assistance of some global data structure [Fonseca et al. 2005; Fang et al. 2005; Nguyeny et al. 2008]; when encountering a dead-end, the protocol uses a scoped flooding to guarantee delivery. This approach obviates the need for GPS devices (or localization services) and planarization algorithms, thus greatly improving its applicability. Moreover, those protocols have been shown to offer a high *greedy forwarding success rate* (GFSR), which is an important factor in routing performance.

Nevertheless, a question that remains unanswered is how well those protocols adapt to general sensing environments with diverse geometric features. The excellent performance of existing protocols are often shown through topologically simple settings, for example, an obstacle-free square, or a field with simple obstacles/holes, yet in real-world scenarios, it is common that the sensor field is irregular, possibly containing obstacles/holes of arbitrary shape. If those protocols could not maintain a high GFSR in such more general scenarios, then the expensive dead-end recovery process may bring the ultimate performance down to an unacceptable level.

Through experiments and theoretical analysis of two prominent protocols, NoGeo [Rao et al. 2003] and BVR [Fonseca et al. 2005], which represent the iterative approach and landmark approach for generating virtual coordinates, respectively, we show serious limitations of existing protocols in topologically nontrivial environments. We then design a new routing protocol, referred to as *CONVEX*, by combining convex partition with existing routing techniques. Simulations demonstrate dramatically improved performance of the new protocol over NoGeo and BVR.

4.1. Understanding Existing Algorithms

In this section, we analyze two representative algorithms: NoGeo and BVR. We examine their behavior in networks with relatively complex topologies, and provide new insights that shed light on the fundamental characteristics of other virtual coordinate routing algorithms. A key metric used in the performance evaluation is *transmission stretch* [Fonseca et al. 2005]. A protocol's transmission stretch, for a given pair of source and destination nodes, is defined as the ratio of the total number of transmitted messages involved in the routing process to the shortest path hop count between the two nodes.

4.1.1. NoGeo. NoGeo's success relies on its coordinate generation based on connectivity information, rather than on physical location information, which is often misleading in greedy forwarding. For example, using true location information, two nodes physically nearby but actually far apart in connectivity (due to, e.g., a long wall between them blocking their direct communication) may wrongly draw in traffic destined to each other. The coordinates generated by NoGeo better reflect the real connectivity of the network, therefore produce a higher GFSR, as demonstrated in Rao et al. [2003] through several simple obstacle settings.

What is not shown in Rao et al. [2003] is to what a degree NoGeo is adaptable to irregularity of the field. If in a topologically nontrivial field NoGeo is unable to maintain a high GFSR, then its ultimate routing performance may degrade considerably, since it has to frequently resort to an expensive *expanding ring* search for overcoming

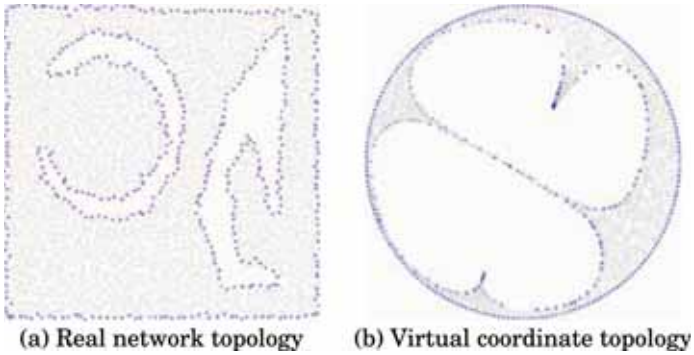


Fig. 6. A network topology and its image under NoGeo. There are 4259 nodes with average degree 10.96. The perimeter nodes are shown in blue (darker color) and ordinary nodes in grey.

dead-ends. The expanding ring is essentially a scoped flooding technique which is commonly used for location-free dead-end recovery [Fonseca et al. 2005; Fang et al. 2005; Nguyeny et al. 2008; Zhao et al. 2005], and is communication costly.

Figure 6 shows a field with two back-to-back C-shaped holes. Execution of the algorithm for 20,000 randomly chosen source-destination pairs in Figure 6(b) shows a surprising result: the average GFSR of NoGeo is only 47.5%, in stark contrast with its nearly 100% success rate in the same network without the holes. The greedy forwarding failures mainly occur on the line between the two virtual holes and in the concave parts of the holes. More experiments show that NoGeo does try to make the original holes more convex, in a way as if a balloon was being inflated. However, if the original holes have a long perimeter, especially if there are more than one such holes, the generated virtual holes may not fully expand into their convex shapes, and may "crowd" together, creating big "traps" for greedy traffic. As a result of the low GFSR, the average/maximum transmission stretch is as high as 11.50/75.97. In comparison, a location-dependent GPSR [Karp and Kung 2000] algorithm achieves a much better average/maximum transmission ratio of 2.32/21.8, despite its even lower GFSR 41.2%.

In conclusion, NoGeo adapts poorly to general field topology, and the field irregularity problem must be addressed in order to obtain reasonable performance in such settings.

4.1.2. BVR. The BVR approach is conceptually simple and relatively easy to implement. The main concern is the number of landmarks needed to guarantee good routing performance. A general trend is that the more landmarks, the better performance, while the higher the overhead for landmark maintenance and addressing. Experiments in Fonseca et al. [2005] show that to achieve a 95% GFSR, the number of landmarks needed remains below 2% of the network size. The simulated field topology is a square, or a square with a number of small-sized obstacles.

We create a more complex network, shown in Figure 7, to see how BVR is adaptable to complex topologies. We set the proportion of randomly selected landmarks to be 2%, and run the algorithm for 20,000 randomly chosen source-destination pairs. The GFSR turns out to be only 83.8%, with an average/maximum transmission stretch of 1.46/111.8. This is clearly much worse than its performance in a regular field. Increasing the proportion of landmarks to 3% and 4% yields a GFSR of 88.65% and 90.85%, respectively, both below the 95% target set in Fonseca et al. [2005].

One reason for the unsatisfactory performance of BVR is the uneven distribution of landmarks. With the disturbance from the irregular hole boundaries, achieving an even distribution is nontrivial [Nguyeny et al. 2008]. As a result, many nodes, especially those near the field/hole boundaries, may end up being far away from their nearest

Fig. 7. A network with 3996 nodes, avg degree 9.7.

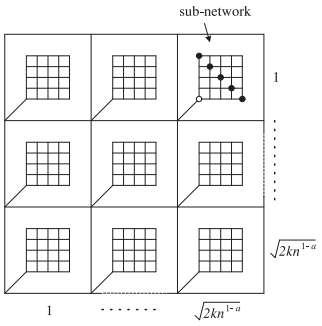


Fig. 8. Lower bound graph for BVR's transmission stretch.

landmarks, thus incurring a high flooding cost when greedy routing fails. To account for this factor, a very high number of landmarks would be needed to ensure a small flooding scope.

Even if the distribution problem could be perfectly solved, a fundamental question yet to be answered is: how many landmarks are needed to ensure a low transmission stretch? The following theorem provides the answer.

THEOREM 4.1. *In an n-node network with* $O(n^{1-a}), 0 < a \leq 1$ *landmarks, BVR has* $\Omega(n^a)$ *transmission stretch.*

PROOF. Assume the number of landmarks is no more than $kn^{1-a}, k > 0$. We construct a networks as shown in Figure 8. The network consists of a $\sqrt{2kn^{1-a}} \times \sqrt{2kn^{1-a}}$ backbone grid network, forming $2kn^{1-a}$ cells. Within each cell, a subnetwork of size $\Omega(n^a)$ is connected to the bottom left corner of the cell. The subnetwork itself is an $\Omega(\sqrt{n^a}) \times \Omega(\sqrt{n^a})$ grid.

Since the number of landmarks is less than the number of subnetworks, there must exist some subnetwork that does not contain a landmark. Without loss of generality, assume such a subnetwork is in the top right cell, and assume the source node is at the bottom left node of the subnetwork, shown as a circle in the figure. Consider the nodes on the diagonal of the chosen subnetwork, shown in filled circles in the figure. These diagonal nodes all have equal hop distance to the bottom left node of the subnetwork, which is the only point connecting the subnetwork to outside. Since all landmarks are outside the subnetwork, these nodes will have the same distance to each landmark, thus their coordinates will be exactly the same. Now pick two random and nonneighboring nodes on the diagonal as a source/destination pair. The source node cannot reach the destination so has to go to the nearest landmark, which will perform a scoped flooding. Such a flood will cover all the nodes below the diagonal, including the diagonal itself, traversing a total of $\Omega(n^a)$ nodes. Since it is easy to pick the source/destination nodes such that their shortest path length is a constant, the transmission stretch is therefore $\Omega(n^a)$. This proves the theorem. □

Remark 4.2. The same lower bound example also applies to HopID [Zhao et al. 2005], LCR [Cao and Abdelzaher 2004; Caruso et al. 2005] for showing their limitations. It holds even when every node knows its c-hop neighborhood, where c is a constant.

Theorem 4.1 indicates that BVR needs $O(n)$ landmarks to ensure a constant transmission stretch. Worse still, the constant factor behind the O symbol seems to be fairly high in our setting, posing a great challenge to BVR's applicability. In Figure 7, for example, we need 3% landmarks only to achieve a GFSR of 88.65%, with a maximum

stretch as high as 50.0. As the number of landmarks grows, the message and state information overheads increase linearly. At the same time, the necessity of a distance vector in greedy routing seems to be diminishing. Observe that at the point of 3% landmarks, the network may be organized such that every landmark is responsible for a cluster of approximately $100/3 \approx 33$ nodes, which is only the average size of a two-hop neighborhood. Establishing a shortest path tree at every landmark plus a simple local routing scheme (e.g., using two-hop neighborhoods or a local naming scheme [Newsome and Song 2003]) would solve the routing problem.

We conclude that BVR's performance is heavily affected by field topology, and if we can afford using a certain fraction of nodes as landmarks, such a fraction can be made reasonably low only if we can solve the field irregularity problem.

4.2. The Routing Protocol

With the partitions determined and recognized, each node can be assigned one or two virtual coordinates in the form $(PartitionID, x, y)$, where $PartitionID$ is the ID of its partition leader, and x and y are local coordinates. These coordinates are managed by some location service [Rao et al. 2003; Fonseca et al. 2005].

We employ the NoGeo method [Rao et al. 2003] to generate local coordinates. Recall that in the partition recognition process, the leader of a partition P has already acquired the full knowledge of P's vertices and edges. It projects the partition polygon onto a virtual circle [Rao et al. 2003] and assigns a coordinate for each of the partition vertices. These assignments are sent to each partition vertex through a separate partition-wide flood. Once a vertex p receives the assignments, it sends a packet to a neighboring partition vertex on P that has a smaller ID than itself. The packet carries two pieces of information: the coordinates of the endpoints of the partition edge it is traversing, and the hop length of that partition edge. This way, all nodes on that edge can calculate its own virtual coordinate on the circle. Eventually all boundary nodes of P will determine their coordinates on the circle, while other nodes in the partition is assigned the same initial coordinate $(0, 0)$. After this, the iterative coordinate calculation begins, as described in Rao et al. [2003]. After a certain number of iterations, every node in P learns its local virtual coordinate. Because the boundary nodes (or perimeter nodes in NoGeo's term) are already known before the iterative process, the heavy-traffic part of perimeter coordinate estimation in NoGeo is avoided in our protocol.

The leaders of partitions serve as global landmarks that help with inter-partition routing. Each of them performs a network-wide flooding to establish a shortest-path tree that covers the whole network. When a source node s wants to route to a destination node t, it first checks if s and t shares a common partition; if so, s performs an intra-partition greedy routing in the same way as NoGeo does; otherwise, s follows the shortest path to t's partition leader until reaching a node that shares a common partition with t, where it begins intrapartition routing. Upon reaching a dead-end, a node uses expanding ring search to find a node closer to the destination than its current position. The pseudocode of the CONVEX routing algorithm is given in Algorithm 1.

Different from other landmark-based protocols, in our protocol the number of landmarks is independent of the network size, but relies on the complexity of large topological features (e.g., the number and shape of holes), which is usually much smaller than network size.

In our implementation, partition leaders are situated on partition boundaries. If the application traffic is roughly uniform over the field, then moving those leaders to their partition centers can improve average interpartition routing performance. This can be done as follows. Suppose a partition P has a leader p. The node p routes a message to the partition center $(0, 0)$. Upon reaching the first node whose radio range covers $(0, 0)$, or encountering the first dead-end, the message's host node becomes the new leader

ALGORITHM 1: The forwarding routine in CONVEX.

input : current node s, detination t, and packet P

if *s and t are in different partitions* **then**
  Send P to the next node on the shortest path between s and the leader of t's partition;
else
  if *t is a neighbor of s* **then**
    Terminate the algorithm;
  else
    Among s's neighbors, find the node s' that is closest to t;
    if $|s't| < |st|$ **then**
      Send P to s';
    else
      Perform expanding ring search to find a node w such that $|wt| < |st|$;
      Deliver P to w;
    end
  end
end

of the partition P. The new leader then floods a new message to the whole network to establish a new shortest-path tree, which replaces the old tree rooted at the old leader, and also to notify all the nodes in P to update their leader information. The message overhead of this optimization is $O(n_{par} \times n)$, where n_{par} is the number of partitions in the network.

4.3. Performance Evaluation

We conduct simulations on various network topologies. The sensor network is deployed in a 1000m $\times$ 1000m square field with irregular holes. Figure 9 shows four example field layouts. The number of sensors are around 4000 with average node degree around 10 [Arora et al. 2005]. Sensor nodes are placed in a perturbed grid distribution [Li and Liu 2007; Bruck et al. 2005]. Each sensor has a communication range of 60 meters, modelled as a uniform disk. By default, $r_0 = 5, \delta_1 = 0.2, \delta_2 = 0.1$. A total of 20,000 source-destination pairs are randomly chosen for test. Two performance metrics are used: greedy forwarding success rate (GFSR) and transmission stretch.

We compare the performance of NoGeo, BVR, CONVEX, and a real location based routing algorithm, GPSR [Karp and Kung 2000]. For handling dead-ends in NoGeo and CONVEX, we use an exponentially expanding ring to search for a closer node to the destination. For BVR, we evaluate its performance under two cases: (1) it has the same number of landmarks as needed by CONVEX; (2) it uses 2% of the nodes as landmarks. The 2% percentage is observed to be sufficient for achieving at least a 95% GPSR in Fonseca et al. [2005]; we will examine whether this holds in more complex network settings. In both cases, the landmark nodes are selected randomly. We also consider the effect of two-hop neighborhoods.

4.3.1. Greedy Forwarding Success Rate. Figure 10 shows the GFSR of the various schemes, all using one-hop neighborhood. In most cases, NoGeo has a GFSR below 50%. For BVR with the same number of landmarks as CONVEX, the GFSR is consistently below 75%. Increasing the number of landmarks to 2% of the network size (around 80) brings the GFSR up to 90.35% in the best case, yet is still worse than CONVEX with much fewer landmarks. For example, in the 2-C topology, CONVEX achieves a GFSR of 94.4%, using only 11 landmarks, in contrast with BVR's 90.35% with 80 landmarks.

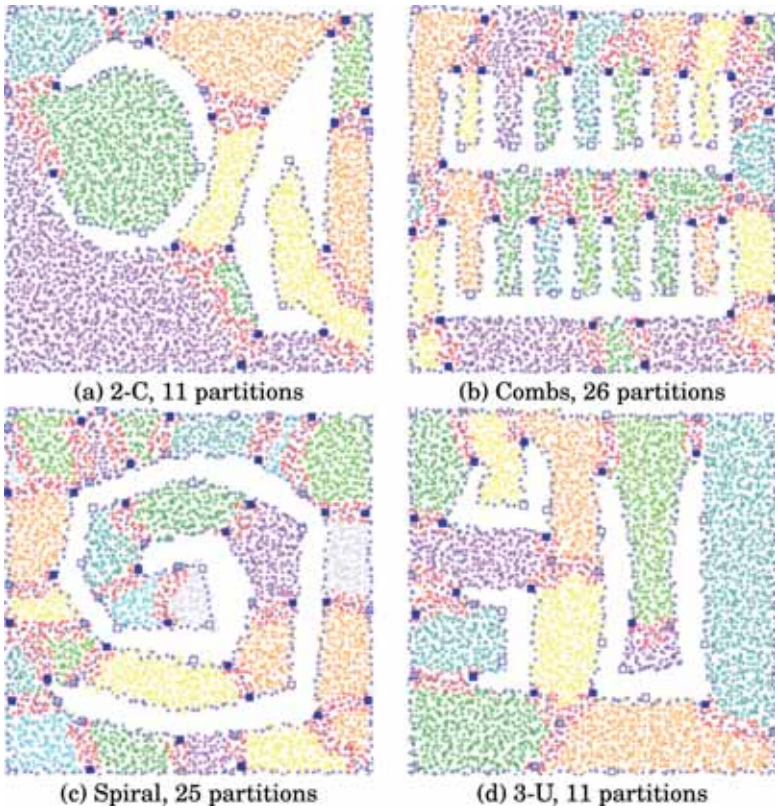


Fig. 9. Four 1000m × 1000m sensor fields after partitioning.

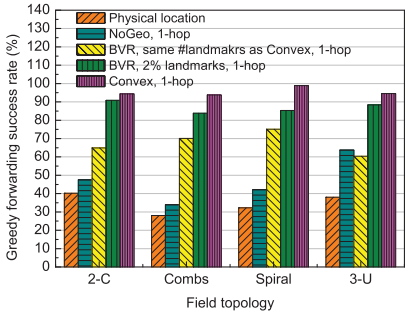


Fig. 10. Greedy routing success rate with one-hop neighborhood.

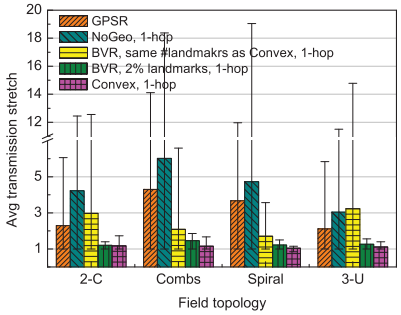
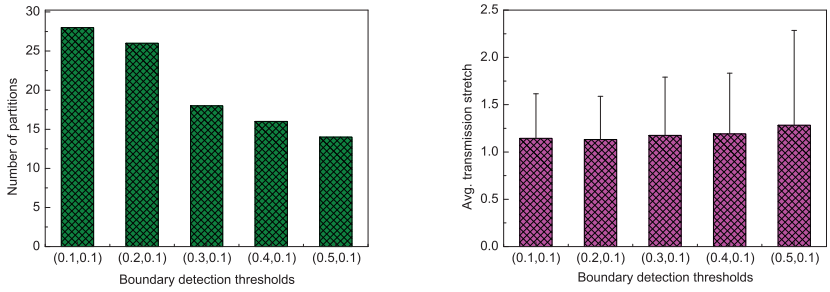


Fig. 11. Average transmission stretch (with 5th and 95th percentiles).

Using two-hop neighborhood brings considerable improvement to all schemes (results not shown here). Throughout the experiments, the GFSR of NoGeo remains below 80%, and BVR never exceeds 85% with the same number of landmarks as CONVEX, while CONVEX has a GFSR of 99.35% at the lowest.

4.3.2. Transmission Stretch. Figure 11 shows the average transmission stretch of the five schemes, along with the 5th and 95th percentiles. As expected, the CONVEX protocol outperforms all other schemes in all cases. Its average stretch is consistently



(a) (b)

Fig. 12. Impact of partitioning parameters (δ_1, δ_2). (a) Number of partitions vs. (δ_1, δ_2). (b) Stretch performance vs. (δ_1, δ_2).

Fig. 13. Impact of non-uniform node distribution, where the upper half of the topology has node density as half as that of the lower half. (a) Topology and partitioning result. (b) Topology with links.

below 1.2. The BVR scheme with 2% of landmarks performs the second best with a small difference; however, one should notice that in this case BVR uses 3–7 times as many landmarks as used by CONVEX, implying a significantly higher overhead. These results demonstrate the great advantage of CONVEX over other schemes.

4.3.3. Impact of Partitioning Parameters. The convex partitioning algorithm is affected by two parameters (δ_1, δ_2). In particular, the parameter δ_1 determines the number of concave critical nodes detected. Since convex partitions are generated by bisector paths from those nodes, fewer concave nodes will lead to fewer partitions being generated. We vary δ_1 and fix $\delta_2 = 0.1$, and run the partitioning and routing algorithm. Figure 12 shows the number of partitions and stretch results as a function of (δ_1, δ_2) under the Combs topology. It can be seen that a higher δ_1 results in fewer (and less convex) partitions being produced, because a higher δ_1 means it takes a sharper corner node to be a concave critical node. One can also see that the routing performance degrades slightly with an increasing δ_1. When a partition becomes less convex, the NoGeo algorithm will produce coordinates that are less friendly to greedy routing, thus resulting in increased path stretch. Overall, the CONVEX algorithm appears to be fairly robust to the partitioning parameters.

4.3.4. Impact of Nonuniform Node Distribution. Figure 13 shows a variant of the Combs topology where nodes in the upper half has node density as half as that of the lower half. The CONVEX algorithm partitions the network into 27 near-convex pieces. The geo-routing algorithm produces an average stretch 1.2. Generally, the nonuniformity

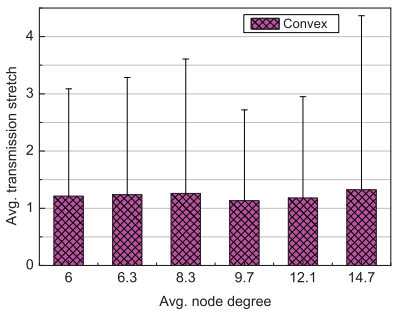


Fig. 14. Impact of node density, measured by average node degree.

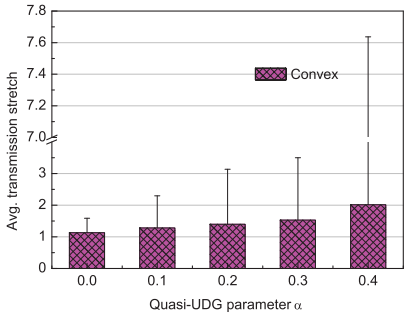


Fig. 15. Impact of Non-UDG radio model with parameter α.

of node distribution adds to the irregularity of connectivity, which introduces increased disturbance to the CONVEX algorithm. The main negative effect, however, is a few more partitions being generated. Since an additional partition only results in one more routing table entry on an average node, we believe the extra cost is acceptable. Therefore, our algorithm is not significantly impacted by nonuniformity in node distribution.

4.3.5. Impact of Node Density. We adjust the communication range to produce various node densities under the Combs topology. The lowest density we experimented was 6, below which the network became disconnected. Figure 14 shows that the routing stretch remains relatively stable across difference densities.

4.3.6. Impact of Non-UDG Radio Model. Figure 15 presents the experimental results under the Combs topology for Quasi-UDG radio model with parameter $0 \leq \alpha < 1$. Under this model, a link exists between two nodes u and v with probability 1 when $|uv| \leq 1 - \alpha$, and with probability $0 < p < 1$ when $1 - \alpha < |uv| \leq 1 + \alpha$; the link does not exist when $|uv| > 1 + \alpha$. We vary α from 0 to 0.4, and adjust p so that the average node degree in different networks remains nearly the same. The generated number of partitions is 26, 23, 22, 23, 28 for these α values, respectively. We can see that the performance degrades gracefully with an increasing α, due to the increased disturbance from the connectivity irregularity to the greedy forwarding algorithm. Note that an $\alpha = 0.4$ represents a rather irregular connectivity pattern, at which point the average stretch is still within 2.

4.3.7. Impact of Node Placement Randomness. We mainly use a perturbed grid model for node placement, which reflects the result of planned deployment with operational errors. In this model, a node's deviation from a planned grid point is governed by a zero-mean Guassian distribution with an adjustable variance, which defines the *perturbation degree*. A perturbation degree of one corresponds to a variance of the size of the grid interval. Figure 16 shows the routing performance with increasing perturbation degree. The numbers of partitions for perturbation degrees from 0.1 to 0.6 are 20, 23, 26, 29, 28, 22, respectively. We can see that the performance is relatively stable for the variations.

4.3.8. More Complex Topology. We also experimented with a realistic and more complex topology to check whether our algorithm can adapt to more complex cases. In Figure 17(a), a total of 5500 sensors are assumed to be deployed on the street area, containing many turns and corners that challenge the partitioning algorithm. Using the default parameters, our algorithm partitions the network into 55 near-convex parts. Running the geo-routing algorithm produces an average stretch 1.75, with a 95-percentile of 4.86. The results are fairly good, because the NoGeo's rubberband

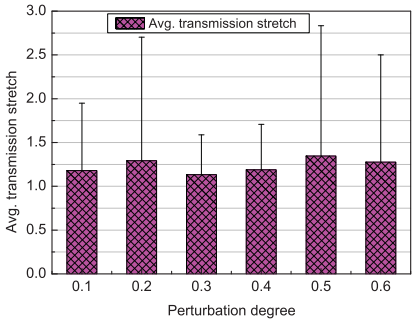


Fig. 16. Impact of node placement randomness, as determined by the variance of the Guassian distribution used in the perturbed-grid node placement.

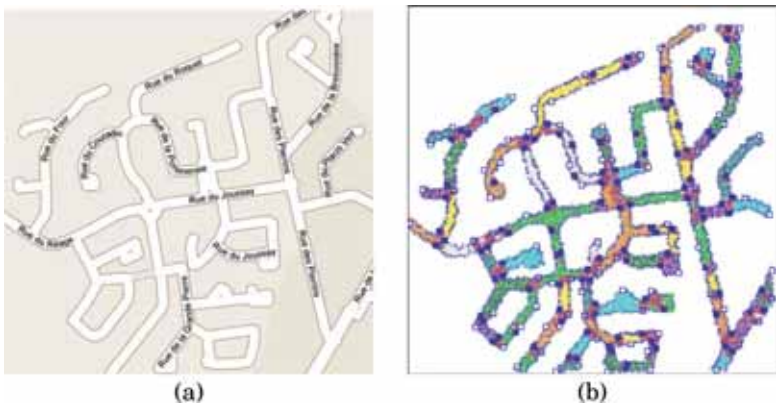


Fig. 17. A realistic and complex network topology. (a) The map of a residential area of many streets. (b) Partitioning result.

algorithm can tolerate some nonconvexity of the partitions, so the generated virtual coordinates can still support efficient geo-routing.

5. CONVEX-PARTITION-BASED LOCALIZATION

Connectivity-based and anchor-free localization aims to assign every node a set of virtual coordinates such that the inter-node distances under these virtual coordinates are proportional to their ground truth. The final network layout should resemble the original network shape, with difference only in size and orientation.

5.1. The Basic Idea

Our basic idea of convex-partition-based localization is to first fix a framework for the network layout, and then to localize individual nodes by trilateration. Here convex partitions serve two purposes: (1) they allow us to recover the shape of a partition in a straightforward way; (2) they help us avoid flip ambiguities [Lederer et al. 2008] when merging the convex partitions to form a global picture.

Since a convex partition does not contain (high-level) network holes, the shortest path between any node pair within a partition should be approximately straight, so the hop distance measurement is reasonably accurate (see Figure 18(a)). This makes it very easy to fix the polygonal vertices one by one. To see the importance of convexity, consider the network layout shown in Figure 18(b), where a partition is not convex and

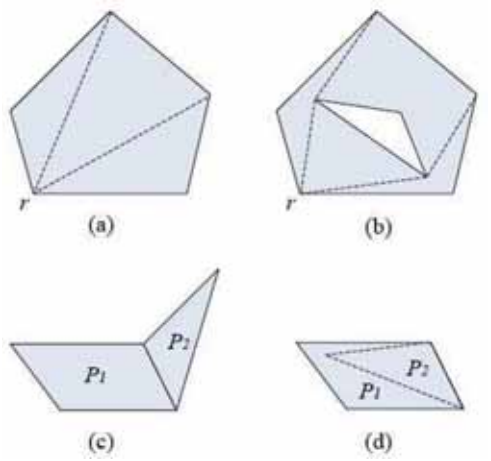


Fig. 18. (a) A convex partition's vertices can be localized in an easy way, for example by using the edge lengths and the distances between a single vertex to all other vertices. (b) In a non-convex partition, intervertex distance measurement is made more difficult by holes. (c) Correct embedding of two neighboring convex partitions. (d) Flip occurs in an incorrect embedding.

contains a hole. This hole will make it much harder to estimate intervertex distance and as a result, bring trouble to proper assignment of coordinates to the polygonal vertices.

The avoidance of flip ambiguities uses thhe fact that the convex partitions to be "glued" together are "solid" pieces, which, when embedded onto the plane, must keep their interiors disjoint (see Figure 18(c)). This means that given two convex partitions with fixed vertices sequences and sharing a common edge, there is a unique way to embed them, thereby avoiding flip ambiguity. The principle of flip ambiguity avoidance is the same as in Lederer et al. [2008] and Wang et al. [2009], where the authors use Delaunay triangles, instead of more general convex pieces, to recover the network layout.

5.2. The Localization Algorithm

Recall that the following facts hold after the partition completes:

—the base station knows the leader node of each partition;
—each partition leader knows its partition's polygonal vertices (nodes) and edge lengths;
—each node knows which partition(s) it belongs to.

In addition, each partition leader learns how to route to the base station through the base station's network-wide flooding (e.g., in the beginning of the CONVEX protocol). With these establishments, the localization algorithm consists of three steps. First, each partition leader localizes its partition's vertices. Second, the partition leaders report to the base station which then collocates individual partitions to form a global framework of the network. Third, nodes in each partition localize themselves by trilateration.

5.2.1. Localizing Individual Partitions. After receiving a flooded command from the base station, the leader l of each partition P performs the following process. First, l floods within P a message containing P's vertices sequence and edge lengths (in hops), instructing P's maximum-ID vertex p_0 to coordinate the localization. The node p_0 then floods within P and collects shortest path lengths between p_0 and each other vertex.

Let the vertices sequence of P be $p_0, p_1, \ldots, p_m$, and let $d(x, y)$ be the minimum hop count between the nodes x and y. Node p_0 first assigns $(0, 0)$ and $(0, d(p_0, p_1))$ to itself

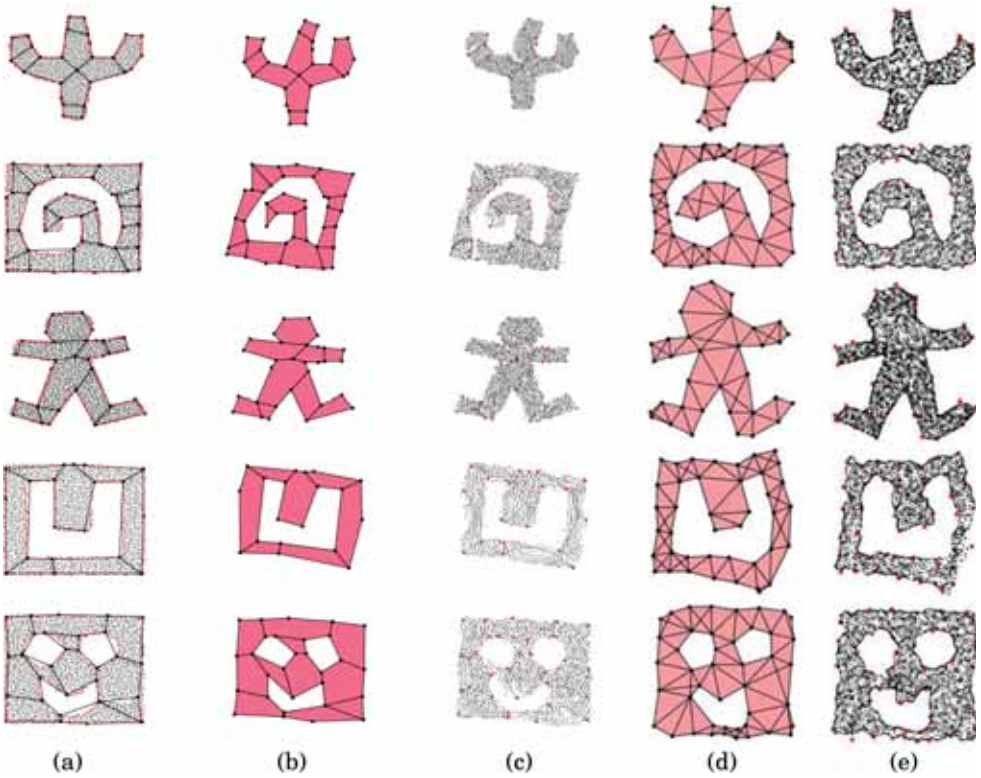


Fig. 19. From left to right: (a) initial sensor locations and extracted convex partitions; (b) embedding of convex partitions; (c) the localization results of our algorithm; (d) embedding of the combinatorial Delaunay complex in LWG08; (e) the localization results of LWG08. First row: Cactus, 1693 nodes with average degree 7.02. Second row: Spiral in a box, 2909 nodes with average degree 9.81. Third row: Ginger man, 2810 nodes with average degree 10.0. Fourth row: Square with a concave hole, 2165 nodes with average degree 10.8. Fifth row: Smiley face, 2793 nodes with average degree 10.1.

Table I. Average Location Error, Scaled by Communication Range

Topology	cactus	spiral	man	concave	face
Our algorithm	1.51	0.96	1.41	0.95	0.62
LWG08	2.39	1.11	1.94	1.88	0.91

and p_1, respectively. With the additional information of $d(p_0, p_2)$ and $d(p_1, p_2)$ it is then easy to assign coordinates to the vertex p_2, assuming without loss of generality a counter-clock traversal direction of the vertices sequence on the plane, that is, p_2 is located in the upper half of the plane.

Flip ambiguity arises when localizing p_3, which may appear in two possible ways: on the right-hand or left-hand side of the vector $\overrightarrow{p_0p_2}$. However, the fact that $\triangle p_0p_1p_2$ and $\triangle p_0p_2p_3$ do not overlap helps the algorithm avoid the wrong case, enabling the proper localization of the remaining vertex nodes $p_3, \ldots, p_m$.

5.2.2. Merging Partitions. After determining the shape of its partition, each partition leader reports to the base station the coordinates sequence of its partition. The base station then tries to merge these partitions to form a global framework.

The adjacency relationship between partitions induce a "partition graph", where each node of the tree corresponds to a partition, and each edge corresponds to an

adjacency relation. Starting from an arbitrary partition, the base station incorporates the remaining partitions into the global framework one by one in a certain traversal order (e.g., Breadth First Search). Again, the fact of nonoverlapping partitions determines a unique way of putting adjacent partitions together. This way, we are able to fix a global framework of the network. We finally use an iterative relaxing algorithm to smooth the partition edges and reduce accumulated errors.

In a last step, the algorithm uses a mass-spring algorithm [Dabek et al. 2004] to smooth the framework of the network. This usually produces a better looking layout.

5.2.3. Localizing Individual Nodes. Since the locations of vertices nodes are known, the other nodes of each partition can be localized by running trilateration algorithm by utilizing hop distance measurement to the vertices of the same partition.

5.3. Performance Evaluation

In this section, we present some simulation results of our algorithm and compare it with a state-of-the-art solution in Lederer et al. [2008], referred to as LWG08, which is shown to perform the best among several recent proposals. We use the network topologies in Lederer et al. [2008] as a benchmark for the evaluation. Figure 19 shows several networks reproduced from the specification of network parameters in Lederer et al. [2008] as well as the visual results of the compared algorithms. It appears that the two algorithms perform comparably in terms of network shape recovery quality. Note that, for ease of visual comparison, the localization results have been scaled and oriented properly.

To quantify localization quality, we feed the two algorithms the true locations of three predetermined nodes (which allow trilateration) and physical average hop length. These allow us to translate the relative coordinates to absolute ones. Based on this, *average location error* is defined by the average distance between the true location and the generated location. Table I gives the error results of the two algorithms and we can see that our algorithm performs better than LWG08.

6. CONCLUSION

Sensor network algorithms that assume a simple geometric environment often do not work well in topologically complex environments. We consider a convex partition approach to tackling this problem. In a convex network piece, algorithms face relatively regular connectivity patterns, thus can often work efficiently. We show how convex partition benefits two applications: virtual coordinate based geographic routing and connectivity-based localization. Experimental results show that the partitioning approach can significantly improve the performance of both applications in comparison with traditional solutions.

REFERENCES

N. Arad and Y. Shavitt. 2006. Minimizing recovery state in geographic ad-hoc routing. In *Proceedings of ACM MobiHoc*.

A. Arora, R. Ramnath, and P. Sinha. 2005. Project exscal. In *Proceedings of the 1st International IEEE Conference on Distributed Computing in Sensor Systems (DCOSS)*.

J. Bruck, J. Gao, and A. Jiang. 2005. MAP: Medial axis based geometric routing in sensor networks. In *Proceedings of ACM MOBICOM*.

Q. Cao and T. Abdelzaher. 2004. A scalable logical coordinates framework for routing in wireless sensor networks. In *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*.

A. Caruso, A. Urpi, S. Chessa, and S. De. 2005. GPS free coordinate assignment and routing in wireless sensor networks. In *Proceedings of IEEE INFOCOM*.

F. Dabek, R. Cox, F. Kaashoek, and R. Morris. 2004. Vivaldi: A decentralized network coordinate system. In *Proceedings of ACM SIGCOMM*.

Q. Fang, J. Gao, L. Guibas, V. de Silva, and L. Zhang. 2005. GLIDER: Gradient landmark-based distributed routing for sensor networks. In *Proceedings of IEEE INFOCOM*.

S. P. Fekete, A. Kroeller, D. Pfisterer, S. Fischer, and C. Buschmann. 2004. Neighborhood-based topology recognition in sensor networks. In *Proceedings of Algorithmic Aspects of Wireless Sensor Networks: First International Workshop (ALGOSENSOR)*.

R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. Stoica. 2005. Beacon vector routing: Scalable point-to-point routing in wireless sensornets. In *Proceedings of USENIX NSDI*.

S. Funke. 2005. Topological hole detection in wireless sensor networks and its applications. In *Proceedings of the Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*.

H. Jiang, T. Yu, C. Tian, G. Tan, and C. Wang. 2012. CONSEL: Connectivity-based segmentation in large-scale 2D/3D sensor networks. In *Proceedings of IEEE INFOCOM*.

B. Karp and H. T. Kung. 2000. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of ACM MOBICOM*.

A. Kroller, S. Fekete, D. Pfisterer, and S. Fischer. 2006. Deterministic boundary recognition and topology extraction for large sensor networks. In *Proceedings of ACM-SIAM SODA*.

F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. 2003. Geometric ad-hoc routing: Of theory and practice. In *Proceedings of ACM PODC*.

F. Kuhn, R. Wattenhofer, and A. Zollinger. 2002. A asymptotically optimal geometric mobile ad hoc routing. In *Proceedings of ACM DIALM-POMC Joint Workshop on Foundations of Mobile Computing*.

S. Lederer, Y. Wang, and J. Gao. 2008. Connectivity-based localization of large scale sensor networks with complex shape. In *Proceedings of IEEE INFOCOM*.

B. Leong, B. Liskov, and R. Morris. 2007. Greedy virtual coordinates for geographic routing. In *Proceedings of IEEE ICNP*.

Mo Li and Yunhao Liu. 2007. Rendered Path: Range-free localization in anisotropic sensor networks with holes. In *Proceedings of ACM MOBICOM*.

J-M Lien and N. M. Amato. 2004. Approximate convex decomposition of polygons. In *Proceedings of ACM Symposium on Computational Geometry (SoCG)*.

A. Lingas. 1982. The power of non-rectilinear holes. In *Proceedings of the 9th International Colloqnium Automata Languages Program*.

W. Liu, D. Wang, H. Jiang, W. Liu, and C. Wang. 2012. Approximate convex decomposition based localization in wireless sensor networks. In *Proceedings of IEEE INFOCOM*.

Y. Lu, J.-M. Lien, M. Ghosh, and N. Amato. 2012. Alpha-decomposition of polygons. In *Proceedings of Shape Modeling International*.

J. Newsome and D. Song. 2003. GEM: Graph embedding for routing and data-centric storage in sensor networks without geographic information. In *Proceedings of ACM SENSYS*.

A. Nguyeny, N. Milosavljevic, Q Fang, J. Gao, and L. J. Guibas. 2008. Landmark selection and greedy landmark-descent routing for sensor networks. In *Proceedings of IEEE INFOCOM*.

A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica. 2003. Geographic routing without location information. In *Proceedings of ACM MOBICOM*.

Z. Ren, J. Yuan, C. Li, and W. Liu. 2011. Minimum near-convex decomposition for robust shape representation. In *Proceedings of ICCV*.

J.-R. Sack and J. Urrutia. 2000. *Handbook of Computational Geometry*. Elsevier Science Publishers, B. V. North-Holland, Amsterdam.

Y. Shang and W. Ruml. 2004. Improved MDS-based localization. In *Proceedings of IEEE INFOCOM*.

Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz. 2003. Localization from mere connectivity. In *Proceedings of ACM MOBIHOC*.

M. Tsai, H. Yang, and W. Huang. 2008. Axis-based virtual coordinate assignment protocol and delivery-guaranteed routing protocol in wireless sensor networks. In *Proceedings of IEEE INFOCOM*.

Y. Wang, J. Gao, and J. S. B. Mitchell. 2006. Boundary recognition in sensor networks by topological methods. In *Proceedings of ACM MOBICOM*.

Y. Wang, S. Lederer, and J. Gao. 2009. Connectivity-based sensor network localization with incremental delaunay refinement method. In *Proceedings of IEEE INFOCOM*.

Y. Zhao, B. Li, Q. Zhang, Y. Chen, , and W. Zhu. 2005. Hop ID based routing in mobile ad hoc networks. In *Proceedings of IEEE ICNP*.

X. Zhu, R. Sarkar, and J. Gao. 2008. Shape segmentation and applications in sensor networks. In *Proceedings of IEEE INFOCOM*.

Received June 2012; revised April 2013; accepted June 2013