Convex Partition of Sensor Networks and Its Use in Virtual Coordinate Geographic Routing

Guang Tan, Marin Bertier, Anne-Marie Kermarrec

Convex Partition of Sensor Networks and Its Use in Virtual Coordinate Geographic Routing

Guang Tan Marin Bertier Anne-Marie Kermarrec
INRIA/IRISA, Rennes, France. Email: {guang.tan, marin.bertier, anne-marie.kermarrec}@irisa.fr

***Abstract*— Virtual coordinate geographic routing is an appealing geographic routing approach for its ability to work without physical location information. We examine two representative such routing protocols, namely NoGeo and BVR, and show through experiments and theoretical analysis their limitation in adapting to complex field topologies, in particular fields with concave holes. Based on the new insights, we propose a distributed *convex partition* protocol that divides the field to subareas with convex shapes, using only connectivity information. A new geographic routing protocol, called *CONVEX*, that builds upon the partitioning protocol is then described. Simulations demonstrate significant performance improvement of the new routing protocol over NoGeo and BVR, in terms of transmission stretch and maintenance overheads.**

I. Introduction

Geographic routing [10] has recently attracted a great deal of attention from the sensor/ad-hoc network community. 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 [10].

There are, however, several problems that stand in the way to the real deployment of geographic routing. First, such a routing strategy requires that each node knows 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 [16]. Second, network planarization algorithms that are needed for correct face routing either rely on an inaccurate unit-disk graph model [10], [13], or are complex and costly [11]. Third, even if the above issues could be ignored, the performance of geographic routing may be far off from optimal in general sensor fields [14].

To address these issues, many protocols have been proposed that use *virtual coordinates* for geographic routing [4], [5], [8], [20], [21], [23]. 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 [6], [8], [19]; 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.

In this paper we investigate this problem. Through experiments and theoretical analysis of two prominent protocols, NoGeo [20] and BVR [8], which represent the iterative approach and landmark approach for generating virtual coordinates, respectively, we show serious limitations of existing protocols in topologically nontrivial environments. Based on the new insights, we propose a distributed *convex partition* protocol that divides the field to subareas with convex shapes. Using only connectivity information, it generates a number of nice-shaped pieces of the network. We then design a new routing protocol, referred to as *CONVEX*, by incorporating existing routing techniques that have proved to be successful within regular fields. Simulations demonstrate dramatically improved performance of the new protocol over NoGeo and BVR.

The remainder of this paper is structured as follows. Section II introduces related work; Section III investigates the behavior of NoGeo and BVR; Section IV describe the convex partition protocol; Section V presents a routing protocol that combines our partition protocol and NoGeo; Section VI presents simulation results, and Section VII concludes the paper.

II. Related Work

Rao et al. [20] first propose the NoGeo family of virtual coordinate assignment algorithms. In NoGeo, perimeter (boundary) nodes use a triangulation algorithm to computer their coordinates. These coordinates are projected onto a

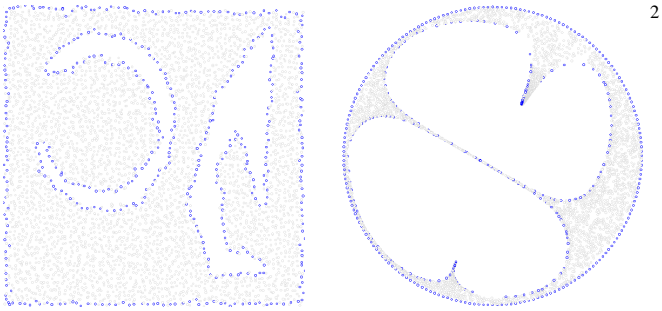
virtual circle, and other nodes then determine their virtual coordinates through an iterative relaxation procedure. NoGeo shows a very high GFSR in a regular field, and is also robust to field irregularity to some degree. NoGeo represents an iterative approach for generating the virtual coordinates, 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 [15] is another iterative virtual coordinate assignment scheme that uses a modified spring relaxation algorithm to incrementally increase the convexity of voids in the network. In [2], Arad and Shavitt 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. [8], represents another approach of virtual coordinate generation. In BVR, nodes'coordinates are assigned in reference to a set of pre-selected landmarks 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 [4], [5], [23], with slight differences in the definition of distance function and dead-end overcoming strategies. The GLIDER protocol [6] 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 [19].

Instead of assigning virtual coordinates based only on hop distances, GEM [18] 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 [3], 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.

We are not the first to consider the partitioning of sensor fields. In [6], the authors propose to partition the field 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. In [24], Zhu et al. propose to segment an irregular sensor fields into nice-shaped pieces. This scheme does not have a notion of convexity, thus may generate concave pieces that contain local minima areas. In the context of computational geometry, convex decomposition of a polygon has been well studied. It is shown [17] that computing a minimum number of a 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 paper is a simple and practical protocol that can be implemented in a distributed way.



(a) Real network topology (b) Virtual coordinate topology

Fig. 1. 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.

III. Understanding Existing Protocols

In this section we analyze two representative algorithms: NoGeo and BVR. We examine their behavior in networks with relatively complex, yet realistic, 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* [8]. 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.

A. 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, for example, 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 [20] through several simple obstacle settings.

What is not shown in [20] 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 dead-ends. The expanding ring is essentially a scoped flooding technique which is commonly used for location-free dead-end recovery [6], [8], [19], [23], and is communication costly.

Figure 1 shows a field with two back-to-back C-shaped holes. Execution of the algorithm for 20,000 randomly chosen

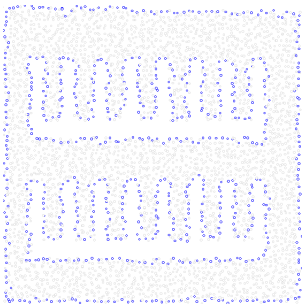


Fig. 2. A network with 3996 nodes, avg degree 9.7.

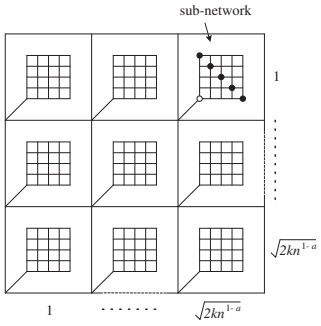


Fig. 3. Lower bound graph for BVR's transmission stretch.

source-destination pairs in Figure 1(b) shows a surprising result: the average GFPR 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 GFPR, the average/maximum transmission stretch is as high as 11.50/75.97. In comparison, a location-dependent GPSR [10] algorithm achieves a much better average/maximum transmission ratio of 2.32/21.8, despite its even lower GFPR 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.

B. 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 [8] 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 2, 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 [8].

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 [19]. As a result, many nodes, especially those near the field/hole boundaries, may end up being far away from their nearest 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 he 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 1: In an n-node network with $O(n^{1-a}), 0 < a \leq 1$ landmarks, BVR has $\Omega(n^{a/2})$ worst-case 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 3. 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 a random node on the diagonal as the destination. The source node can by no means tell in which direction the destination lies, thus 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 the shortest path length between the source and destination is $O(n^{a/2})$, the transmission stretch is therefore $\Omega(n^{a/2})$. This proves the theorem. ■

Remark 1: The same lower bound example also applies to HopID [23], LCR [4] and [5] for showing their limitations. It holds even when every node knows its c-hop neighborhood, where c is a constant.

Theorem 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 2, 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 [18]) 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.

IV. The Convex Partition Protocol

In a discrete network, we are most interested in the high-level topological features such as the number and shape of field holes, often caused by large obstacles. Those holes should be orders of magnitude larger than average inter-node distance; such sizes make their impact on routing performance so significant that they deserve special treatment. Considering the uneven node distribution and limited node densities, our aim is to partition the field into approximately convex polygonal sub-areas, and to let every node know which partition(s) it belongs to.

We assume that each node p has a unique ID, denoted by $ID(p)$. The *boundary nodes* are nodes on the field/hole boundaries, and are determined by some boundary detection process. We do not assume a regular radio model (e.g., unit disk model).The field can be heterogenous in sensor densities, but across a small region the density does not change significantly.

Definition 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 2: r**-hop interior node.** An *r-hop interior node* is a node whose nearest boundary node is at least r-hops away.

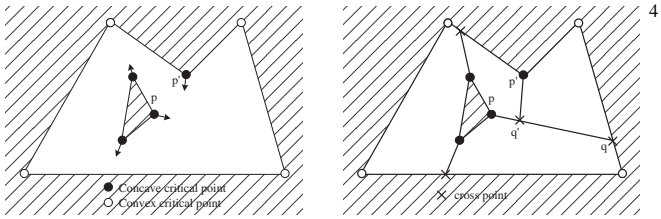
Given r, let D_r be the average r-hop neighborhood size of all r-hop interior nodes in a certain region. D_r measures a "full" r-hop neighborhood size, which is analogous to the area of a full disk of radius r in the continuous domain. As such, we also call D_r the *r-hop node density* in that region.

Definition 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 4: **Concave/Convex Critical point.** Let $\delta_1 > 0$ and $\delta_2 < 1$ be two pre-defined 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 4 for an illustration.



(a) Drawing bisectors (b) The final partitions

Fig. 4. Convex partition of a polygonal environment. The sensor field is shown as non-shaded area. Both critical points and cross points are vertices of the final partitions.

The convex partition protocol requires the boundary nodes to be identified first. One of existing techniques (e.g., [7], [9], [12], [22]) can be adopted to do this. These algorithms only need network connectivity information and typically require $O(n)$ message transmission. After running such an algorithm, each hole is assigned a unique ID, and each boundary node belongs to a certain hole and is tagged with the hole ID. The rest of 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.

A. 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-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 wether 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 non-boundary 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 the above 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 the above process is to noises. In our experiments values between $[0.1, 0.2]$ prove to be a good choice for practical purposes. Figure 5(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.

B. 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 4(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 travelling, 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 4(b)).

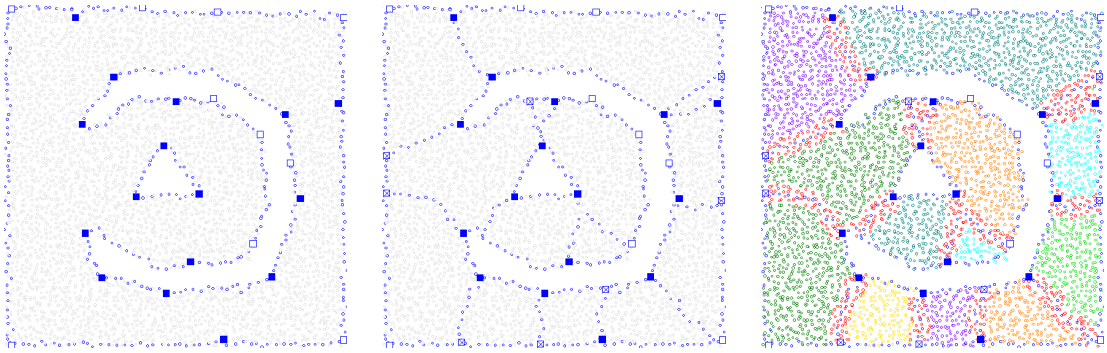
Theorem 2: All the partitions generated by the above protocol are convex polygons.

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. ■

C. 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 5(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 clockwise and counterclockwise 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 above. 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 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



(a) Concave/convex critical points. (b) Bisectors. (c) 14 partitions generated.

Fig. 5. 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. $r_0 = 5, \delta_1 = 0.2, \delta_2 = 0.1$.

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, 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 4(c) shows an example of the partition result on a network.

D. 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 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 beyond the belt areas, one in clockwise direction and the other in counterclockwise direction, on its field/hole boundary. If such a node exists, then p commands it to issue a flood.

Suppose a node p 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 p via the reverse routes. p then collects this information into an edge set

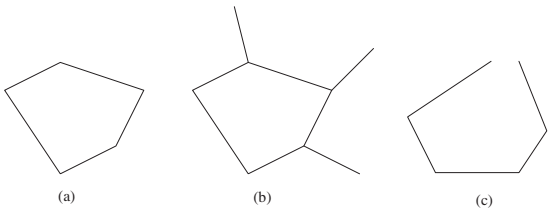


Fig. 6. The three cases in partition construction.

$\mathcal{E}$, 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 6).

1) The edges in $\mathcal{E}$ form exactly a cycle (Figure 6(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 6(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 6(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 p 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 p_{max} with the highest ID prevails, and the nodes in P only take p_{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.

E. 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.

a) Boundary nodes identification phase: This phase incurs $O(n)$ message overhead.

b) 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.

c) 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 message is $O(n \times n_{cri})$. n_{cri} 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.

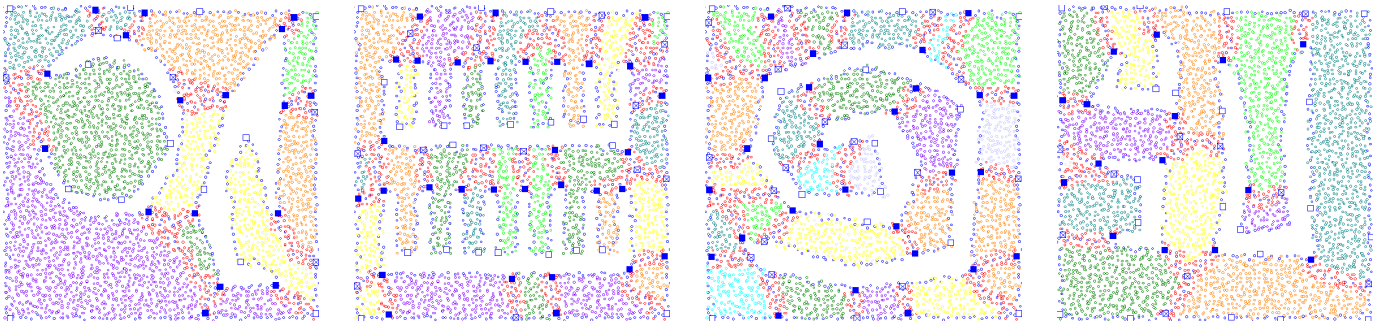
d) Partition recognition phase: Let n_{par} be the number of partitions generated. It is easy to see that $n_{par} \in O(n_{cri})$, therefore this phase incurs $O(n \times n_{cri})$ messages as well.

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.

V. Convex Partition based 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.

We employ the NoGeo method [20] 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 [20] 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 [20]. 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.



(a) 2-C, 11 partitions (b) Combs, 26 partitions (c) Spiral, 25 partitions (d) 3-U, 11 partitions

Fig. 7. Four $1000m \times 1000m$ sensor fields after partitioning.

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 intra-partition routing. Upon reaching a dead-end, a node uses expanding ring search to find a node closer to the destination than its current position.

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 inter-partition 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 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.

VI. 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 7 shows four example field layouts. The number of sensors are around 4000 with average node degree around 10 [1]. Sensor nodes are placed in a perturbed grid distribution [3], [16]. 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 [10]. 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 [8]; 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.

A. Greedy Forwarding Success Rate

Figure 8 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.

Using two-hop neighborhood brings considerable improvement to all schemes (detailed results not shown due to space limitation). 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.

B. Transmission Stretch

Figure 9 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 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

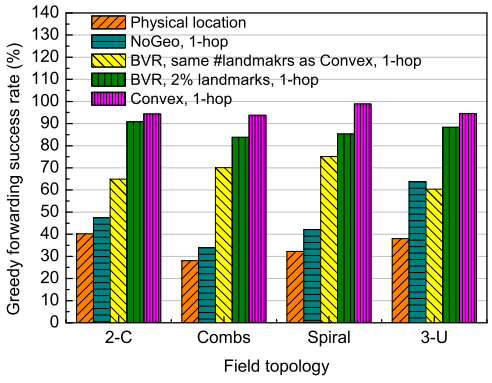


Fig. 8. Greedy routing success rate with one-hop neighborhood.

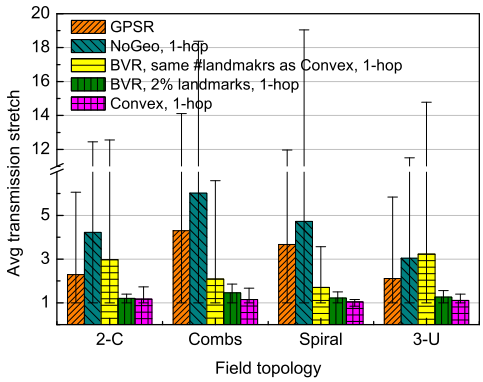


Fig. 9. Average transmission stretch (with 5th and 95th percentiles).

CONVEX, implying a significantly higher overhead. These results demonstrate the great advantage of CONVEX over other schemes.

C. Effect of Partition Parameters

The parameters δ_1 and δ_2 determine how sensitive the convex partition protocol is to boundary noises. Generally, the smaller the two parameters, the more critical points will be created, which result in more partitions being generated. As a result those smaller partitions tend to be more convex, leading to better routing performance within each partition. The downside is that the more landmarks will incur higher maintenance overheads. Taking the spiral topology (Figure 7(c)) as an example, if we increase them from their default values $\delta_1 = 0.2, \delta_2 = 0.1$ to $\delta_1 = 0.3, \delta_2 = 0.2$, the number of partitions generated will drop from 25 to only 15, and the average transmission stretch increases from 1.05 to 1.10.

VII. Conclusion

We have presented a convex partition protocol of sensor networks. The motivation for designing such a protocol is the poor performance of existing virtual coordinate geographic routing protocols in sensor fields with complex (in particular concave) topologies. Partitioning the fields into convex components that suit those protocols thus provides a natural way to overcome the field irregularity problem. We have demonstrated the great benefit of our protocol through a routing protocol that integrates the partition protocol and techniques from NoGeo.

References

[1] A. Arora, R. Ramnath, P. Sinha, and et al. Project exscal. In *Proc. 1st Int. IEEE Conf. on Distributed Computing in Sensor Systems (DCOSS), 2005.*

[2] N. Arad and Y. Shavitt. Minimizing recovery state in geographic ad-hoc routing. *Proc. of ACM MobiHoc 2006*.

[3] J. Bruck, J. Gao, A. Jiang. MAP: Medial Axis Based Geometric Routing in Sensor Networks, *Proc. of ACM MobiCom 2005*.

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

[5] A. Caruso, A. Urpi, S. Chessa, and S. De. GPS free coordinate assignment and routing in wireless sensor networks. *Proc. of IEEE INFOCOM 2005*.

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

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

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

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

[10] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proc. of ACM MobiCom 2000*.

[11] Y-J Kim, R. Govindan, B. Karp, and S. Shenker. Geographic Routing Made Practical. *Proc. of NSDI 2005*.

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

[13] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric ad-hoc routing: Of theory and practice. In *Proc. of 22nd ACM International Symposium on the Principles of Distributed Computing (PODC)*, 2003.

[14] F. Kuhn, R. Wattenhofer, and A. Zollinger. A Asymptotically Optimal Geometric Mobile Ad Hoc Routing. In *Proc. of ACM DIALM-POMC Joint Workshop on Foundations of Mobile Computing*, 2002.

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

[16] M. Li and Y. Liu. Rendered Path: Range-Free Localization in Anisotropic Sensor Networks with Holes. In *Proc. of MobiCom 2007*.

[17] A. Lingas. The power of non-rectilinear holes. In *Proc. 9th Internat. Colloq. Automata Lang. Program*, volume 140, LNCS 369-383. 1982.

[18] J. Newsome and D. Song. Gem: Graph embedding for routing and data-centric storage in sensor networks without geographic information. In *Proc. of SenSys 2003*.

[19] A. Nguyeny, N. Milosavljevic, Q Fang, J. Gao, and L. J. Guibas. Landmark Selection and Greedy Landmark-Descent Routing for Sensor Networks. In *Proc. of IEEE INFOCOM 2008*.

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

[21] M. Tsai, H. Yang, and W. Huang. Axis-Based Virtual Coordinate Assignment Protocol and Delivery-Guaranteed Routing Protocol in Wireless Sensor Networks. *Proc. of IEEE INFOCOM 2008*.

[22] Y. Wang, J. Gao, J. S.B. Mitchell. Boundary Recognition in Sensor Networks by Topological Methods. *Proc. of ACM MobiCom 2006*.

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

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