# GPS-Free Greedy Routing With Delivery Guarantee and Low Stretch Factor on 2-D and 3-D Surfaces

Su Xia, Hongyi Wu, and Miao Jin

***Abstract*—This paper focuses on greedy routing in wireless networks deployed on 2-D and 3-D surfaces. It introduces a distributed embedding scheme based on the conformal map theory. The proposed scheme identifies the convex hull of each boundary and employs Yamabe flow to compute flat metric under convex hull boundary condition to establish virtual coordinates. Such virtual coordinates are then used for greedy routing. Since the proposed embedding algorithm maps the outer boundary to a convex shape and an interior concave void to a circle-like convex polygon, it effectively eliminates local minimum and attains guaranteed delivery. At the same time, it introduces a small distortion only and consequently achieves a low stretch factor. Our simulations show that its stretch factor is lower than any existing greedy embedding algorithms. Moreover, the proposed scheme is merely based on local connectivity and consumes a small constant storage, thus scaling to arbitrarily large networks.**

***Index Terms*—Greedy routing, wireless sensor networks.**

## I. Introduction

With both storage space and computation complexity bounded by a small constant, greedy routing is deemed an appealing approach for resource-constrained networks [1]. In most cases, a node under greedy routing only needs to store its coordinates and perform a standard distance calculation for routing, rendering it particularly suitable for networks where the nodes have limited memory space or computing capacity.

One of the earliest greedy routing schemes is introduced in [2] and [3]. It assumes that nodes know their geographic locations. To route a data packet, the algorithm greedily advances it to the next-hop node that is the closest to the destination. If a dead-end is reached, it employs face routing to move the packet around the perimeter of the void. Followup studies have extended investigations into variances of face routing [4]–[9], location errors [10], and 3-D space [11].

The geographic location information is not always available or precise enough to support efficient greedy routing. This constraint has naturally stimulated the development of virtual coordinates-based schemes. To this end, a collection of interesting approaches have been explored for establishing virtual coordinates in the network without physical location information. In [12], the rubber band algorithm is adopted to create the virtual coordinates based on a convex embedding of the network graph. In [13]–[18], the virtual coordinates are defined as the hop distances to an array of selected anchor nodes. A tree structure is proposed in [19] to name the nodes sequentially, creating an ordered 1-D coordinate for every node in the network. In addition, [20] and [21] propose to partition the network into convex regions such that greedy routing can be applied in each region.

The majority of the aforementioned virtual coordinate-based greedy routing schemes [12]–[17], [20], [21] do not guarantee delivery. In other words, dead-end still exists and special routing schemes (such as face routing or flooding) must be employed to handle it. References [18] and [19] do ensure successful delivery between any pair of nodes in the network. However, [18] potentially requires large storage space per node, which grows with the network size (in contrast to other greedy routing schemes that need a small constant storage only); on the other hand, the 1-D coordinates adopted in [19] may lead to long routing path between adjacent nodes.

While experimental results have shown the efficiency of virtual coordinates-based greedy routing [12]–[21], the quest for theoretical understanding of its delivery guarantee and routing efficiency leads to a thrust of exploratory research that has revealed several interesting findings recently.

1) There is a great interest to delve into the conditions that ensure *greedy embedding*. A greedy embedding is an embedding of a graph such that, given any two distinct nodes $s$ and $t$, there is a neighbor of $s$ that is closer to $t$ than $s$ is [22]. In other words, greedy embedding ensures the success of greedy routing. While it is yet an open problem, studies have shown that greedy embedding does not exist for all graphs. But a three-connected graph always admits a greedy embedding in the plane [22]–[24].
2) Subsequently, it is discovered in [25] that any connected graph has a greedy embedding in the hyperbolic plane. To enable greedy routing, a spanning tree is constructed and the nodes on the tree are mapped to a hyperbolic plane. Then, the hyperbolic distance is used for greedy routing. This approach is extended for dynamic graphs in [26].
3) Reference [27] introduces a mapping approach that employs Ricci flow to map holes (that can be local minimum and lead to dead-end) to perfect circles, and thus ensuring successful greedy routing.
4) Greedy routing usually leads to a longer path compared with the shortest path in conventional routing. The efficiency of greedy routing is gauged by its *stretch factor*,

Manuscript received February 10, 2014; revised March 29, 2014; accepted April 15, 2014. Date of publication April 25, 2014; date of current version May 30, 2014. The work of H. Wu was supported in part under Grant NSF CNS-1018306 and Grant CNS-1320931. The work of M. Jin was supported in part under Grant NSF CCF-1054996, Grant CNS-1018306, and Grant CNS-1320931.

S. Xia is with Cisco Systems Inc., Milpitas, CA 95035 USA (e-mail: suxia.ull@gmail.com).

H. Wu and M. Jin are with the Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA 70503 USA (e-mail: wu@cacs.louisiana.edu; mjin@cacs.louisiana.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JIOT.2014.2320260

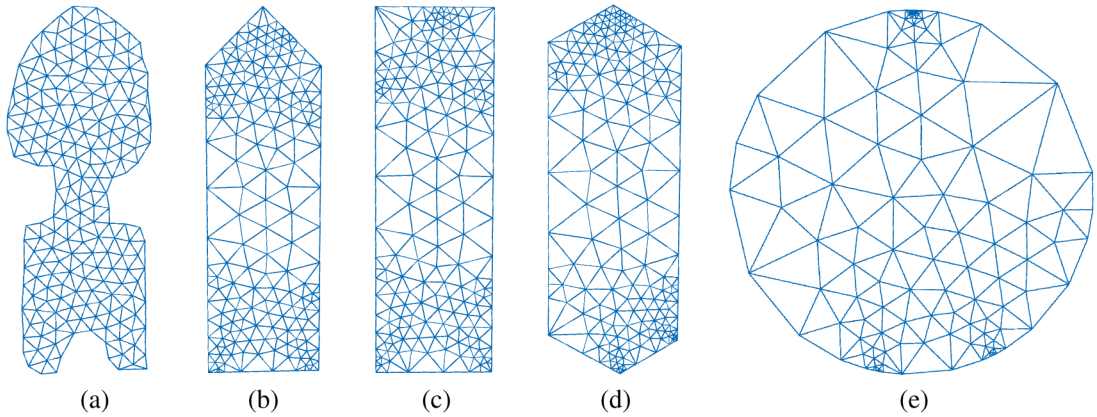



Fig. 1. Distortion under different boundary conditions. (a) Original mesh. (b) Mesh conformally embedded into a 2-D plane under a convex boundary condition. (c) Mesh conformally embedded into a 2-D plane under a rectangle boundary condition. (d) Mesh conformally embedded into a 2-D plane under a hexagon boundary condition. (e) Mesh conformally embedded into a 2-D plane under a circular boundary condition.

which is the ratio between the average path length in greedy routing and the average shortest path length. [28] is the first paper that reports a bounded stretch factor in greedy routing. It introduces a centralized embedding algorithm that yields virtual coordinates with $O(log^3 n)$ bits. It achieves a constant stretch factor under combinatorial unit disk graphs and $O(log(n))$ stretch for general graphs.

*Our contribution*: This work continues the research thrust for greedy embedding. We aim to develop a practical distributed embedding scheme that guarantees the success of greedy routing between any pair of nodes in the network and, at the same time, achieves a low stretch factor.

Failures in greedy routing are due to local minimum. For example, when a packet is greedily forwarded to an intermediate node along the outer boundary of a network or the boundary of an inner hole with concave shapes [see the concave hole in Fig. 5(a)], the node finds itself to be the local minimum (i.e., the one with the shortest distance to the destination in its neighborhood), and thus fails to further advance the packet toward the destination. To avoid such failures, our proposed embedding algorithm maps the outer boundary to a convex shape and an inner concave hole to a circle-like convex polygon [see Fig. 5(e)]. Therefore, it effectively eliminates local minimum and attains guaranteed delivery in greedy routing.

A side-effect of such mapping is distortion. For the sake of a visualization and conceptual understanding of the mapping process (that will be elaborated in Section III), let us analogize edges as rubber bands. When the boundary of a void is forced to deform to a convex shape, a network-wide distortion will be observed, i.e., the edges will deform and the nodes will move accordingly. For example, the nodes in the original network shown in Fig. 5(a) are uniformly distributed. After mapping [see Fig. 5(e)], the nodal density at the left bottom corner becomes obviously lower. The amount of distortion can be quantified by the method discussed in [29]. To minimize distortions, the original boundary condition should be preserved [29], [30]. However, an embedding based on original boundaries is not useful in greedy routing, because a void may be concave and thus results in failures. When the boundaries must be deformed, the closer to the original boundary condition, the less distortion is introduced. This is verified by our simulations. For example, Fig. 1 illustrates a network and its embedding under different convex boundary conditions. As can be seen, since a convex hull is by definition the minimal convex set of the original boundary, mapping the boundary to its convex hull leads to the smallest distortion. The quantified distortions are given in Fig. 2 (see the values at the top of the histogram bars).

Meanwhile we note that, when greedy routing succeeds in the original network, its path length is generally close to the shortest path length, although greedy routing does not always follow the shortest path. Due to distortions, however, the greedy routing path in the embedded space can noticeably deviate from its counterpart in the original space and consequently the true shortest path. Any such deviations lead to stretch, i.e., a longer route than the shortest path. As a result, a greedy routing path in the embedded space is generally stretched, and its stretch factor is proportional to the amount of distortions. This is illustrated in Fig. 2, which shows the distortion and the stretch factor of the network embedded under different convex boundary conditions.

Therefore, to achieve greedy embedding with low stretch factor, we must keep distortions as small as possible in mapping. To this end, we propose an embedding scheme, where we identify the convex hull of each boundary and employ Yamabe flow to compute flat metric under convex hull boundary condition to establish virtual coordinates that are used for greedy routing. It maps the outer boundary to its convex hull and an interior void to a circle-like convex polygon, yielding several appealing features summarized as follows.

1) *GPS-Free*: The proposed scheme does not require location information. It is based on local connectivity only.
2) *Constant storage*: Its storage is a small constant, unlike [18], [28] whose storage grows with network size.
3) *Guaranteed greedy forwarding*: The proposed scheme achieves 100% delivery rate under greedy routing.
4) *Low stretch*: Our simulations show that its stretch factor is lower than any existing greedy embedding algorithms.
5) *Distributed implementation*: Distinct from [28], our scheme is distributed, based on its local information only.
6) *Scalability*: Our scheme scales to large networks. This is in sharp contrast to [25] and [26] whose required computation precision grows dramatically with network size.

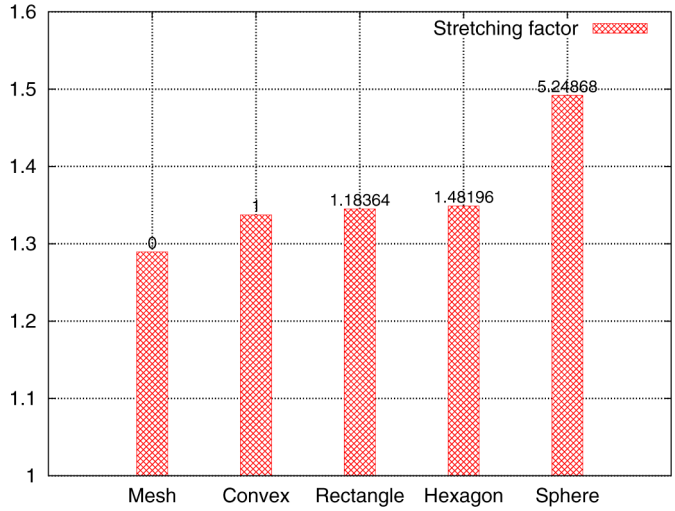



Fig. 2. Higher distortion results in higher stretch factor. The value at the top of each histogram bar indicates the amount of distortions (calculated according to [29]) of a embedding scheme illustrated in Fig. 1.

7) *3-D compatibility*: While earlier studies mainly concentrate on 2-D space, our proposed scheme can establish virtual coordinates in both 2-D and 3-D surfaces.

This paper is organized as follows. Sections II–IV introduce our proposed algorithm in three steps, including: 1) preprocessing, 2) virtual coordinates calculation, and 3) routing based on virtual coordinates. Section V presents simulation results. Finally, Section VI concludes this paper.

## II. Preprocessing

Our proposed routing scheme consists of three steps: 1) preprocessing, 2) virtual coordinates calculation, and 3) routing based on virtual coordinates. Preprocessing is a distributed process that prepares the necessary network information for the Yamabe flow-based conformal mapping. The outcome of the mapping algorithm is the virtual coordinates for every node in the network, which are used for greedy routing. The properties of Yamabe flow-based conformal mapping ensure the success of such greedy routing between any pair of nodes in the network and achieve low stretch factor at the same time.

The preprocessing process aims to identify a subset of the nodes (dubbed landmarks) and their connections in the network to construct a triangle mesh structure. Such triangle mesh and the corresponding boundary information are required as inputs for running the Yamabe flow-based conformal mapping algorithm. Furthermore, the triangulation forms a backbone representation of the network, which can effectively reduce the complexity in conformal mapping and routing.

With no location information available, we adopt the method proposed in [31] for triangulation. It first identifies a set of landmarks such that any two neighboring landmarks are $k$-hops apart and a nonlandmark node is associated with the nearest landmark within $k$-hops ($k > 5$). This distributed method creates a set of approximated Voronoi cells. Then, it connects the landmarks to yield a *combinatorial delaunay map* (*CDM*). The algorithm guarantees that landmarks are chosen uniformly, but not densely from a network and the constructed CDM is a planar graph [31]. An example of a constructed CDM is shown in Fig. 3.

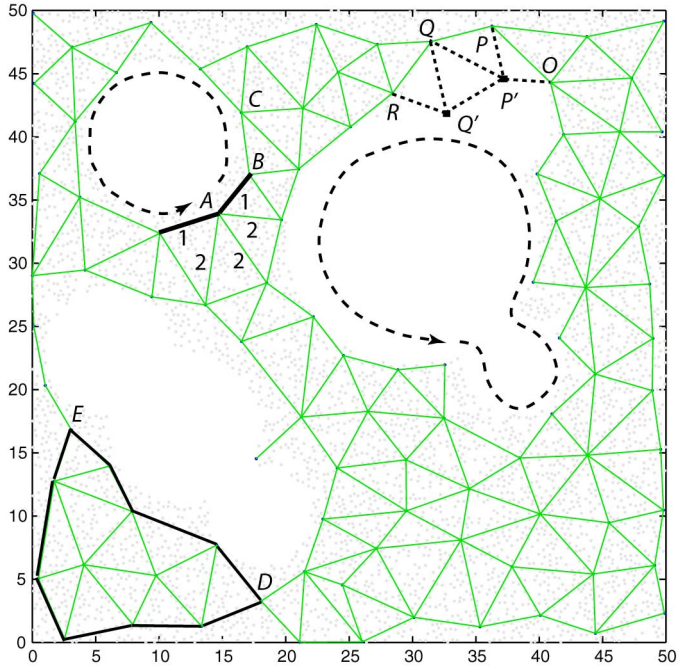



Fig. 3. Example of triangulation.

Vertices of a CDM are landmark nodes. An edge in CDM connecting two neighboring vertices is a shortest path between the corresponding two landmarks. To run conformal map algorithms, however, it must be further processed. More specifically, we must discover the boundaries of the CDM and repair its degeneracy edges.

### A. Boundary Detection

To detect boundaries in CDM, we first identify the edges that define the boundaries. As can be seen in Fig. 3, a boundary edge belongs to no more than one triangle. Based on this observation, we devise a simple approach for discovering boundary edges, which does not require topology information like [27]. Each landmark broadcasts a probe packet with its own identification (ID) number and time-to-live (TTL) equal to three. Its neighboring nodes (in CDM) rebroadcast the packet with the TTL decreased by one. The process repeats as long as the TTL of a packet is greater than zero, whenever a copy of the packet returns to the landmark that initiates the probe, it must have gone through a triangle. As a result, a landmark can discover how many triangles a neighboring edge is attached to by observing the number of its own probe packets received through that edge. If less than two probe packets are received from that edge, it must be a boundary edge. By the end of this process, each landmark can identify all boundary edges it connects to. For example, Landmark $A$ in Fig. 3 connects to five edges. Two of them are identified as boundary edges (see the thick lines), because only one probe packet is received from each of them, while two packets are received from every other edge.

After all the boundary edges are identified, a landmark that connects to a boundary edge (e.g., $A$) sends a boundary discovery packet through that edge to its neighboring landmark. If the neighboring landmark (e.g., $B$ in Fig. 3) connects to more than

one boundary edge (other than the one connected with $A$), it adds its ID into the boundary discovery packet and forwards it through a different boundary edge; otherwise it simply drops the packet. If the boundary discovery packet loops back to the landmark that initiates the process (i.e., $A$), a boundary is discovered. This method works efficiently except a special case illustrated in the lower left corner of Fig. 3. If landmark node $D$ initiates boundary discovery, it may end up with finding a loop highlighted by the thick lines and assume it is a boundary. This problem is due to degenerated nodes. A node connecting to more than two boundary edges is called a degenerated node (see Nodes $D$ and $E$ for example). With the presence of degenerated nodes, the subgraph can flip and lead to misleading results in boundary discovery. To solve this problem, we add the following rule in forwarding the boundary discovery packet: if a boundary discovery packet goes through two adjacent boundary edges that intersect at a degenerated node and form one or multiple adjacent triangles, the packet is dropped. This rule avoids the problem discussed above and ensures correct boundary discovery.

In addition, a random chosen landmark node can initiate a process to assign a consistent orientation to the whole triangulation. We let the one with the smallest ID compared with other landmarks to start the process. There are many different distributed methods to find the starting node. For example, each landmark broadcasts a packet with its ID to its neighboring landmarks. A landmark receives a packet and compares with its own ID. If the received ID is larger than its own, the landmark will drop this packet, otherwise, just forward it. Eventually, all landmarks will know the smallest landmark node ID. The landmark node with the smallest ID then assigns an orientation (either clockwise or counterclockwise) to one of its connecting triangles. The triangle will then propagate its orientation to its neighboring triangles such that any neighboring triangles share the same orientation—all clockwise or all counterclockwise. Eventually the entire triangles of the network share the same orientation—all clockwise or all counterclockwise.

### B. Degenerated Nodes and Edges

A degenerated node is a node connecting to more than two boundary edges (see Nodes $D$ and $E$ for example). A degenerated edge is an edge connecting two degenerated nodes (see Edge $PQ$ in Fig. 3 for example). The degenerated nodes and edges must be repaired before computing the virtual coordinates in the second step of our algorithm.

It is obvious that the degenerated nodes must locate on boundaries. Any landmark node on a boundary can initiate the process of detecting and repairing degenerated nodes and edges. To avoid multiple nodes along the same boundary loop start the process simultaneously, we can choose the node with the smallest ID along a boundary loop to start the process. The initiator node sends a probe packet that travels along the boundary according to its orientation. Whenever the probe packet reaches a degenerated node (e.g., Node $P$ in Fig. 3), it triggers the repairing process. More specifically, a virtual node is added along the boundary (see $P'$) and connected to the degenerated node and its predecessor and successor (i.e., Nodes $O$ and $Q$, respectively). The boundary edges are updated accordingly. For example, Edges $P'O$ and $P'Q$ are now the boundary edges. The probe packet is then forwarded to the next node $Q$, which is again a degenerated node, with predecessor and successor of $P'$ and $R$, respectively. Similarly, a virtual node $Q'$ and the corresponding edges are added. The process repeats, until the probe packet returns to its initiator.

In summary, preprocessing establishes a triangular mesh with all boundaries detected and degenerated edges repaired.

## III. Virtual Coordinates Calculation

Our algorithm in this step aims to find the flat metric of the triangular mesh, which can be embedded on a 2-D plane with minimum distortion and ensure successful greedy forwarding. The flat metric is used to determine virtual coordinates that are to be employed for greedy routing. It is a distributed procedure. In the rest of this section, we first introduce the theory of discrete conformal mapping and Yamabe flow in Section III-A, and then present the algorithm itself in Section III-B. The readers may choose to skip Section III-A if s/he is not interested in the theoretic background.

### A. Theory of Discrete Yamabe Flow

In this section, we present the discrete theory that serves as the basis of our proposed algorithm. We also give a brief introduction of Yamabe flow, a tool to be used to compute flat metric.

In discrete setting, we let $M = (V, E, F)$ to represent an abstract surface triangulation mesh (or mesh in short), consisting of vertices ($V$), edges ($E$), and faces ($F$). How to create such a mesh has been discussed in Section II. We do not restrict its topology or the number of boundaries.

First of all, we introduce several relevant definitions as follows.

*Definition 1:* A discrete metric on $M$ is a function $l$ on the set of edges, assigning to each edge $e_{ij} \in E$ a positive number $l_{ij}$ so that the triangle inequalities are satisfied for all triangles $t_{ijk} \in F$: $l_{ij} + l_{jk} > l_{ki}$.

The edge lengths of $M$ induced from Euclidean space are sufficient to define a discrete metric on $M$

$$l : E \to \mathbb{R}^+. \tag{1}$$

*Definition 2:* Two discrete metrics $l$ and $\overline{l}$ on $M$ are conformally equivalent if, for some real numbers $\{u_i | v_i \in V\}$ assigned to the vertices $V$, the two metrics satisfy

$$\overline{l_{ij}} = e^{u_i + u_j} l_{ij}. \tag{2}$$

We call $u_i$ the discrete conformal factor of vertex $v_i$.

*Definition 3:* Let the vertex set be $V = (v_1, v_2, \ldots, v_n)$. For each vertex $v_i$, its conformal factor is $u_i$. A discrete metric of a mesh can be represented by a vector $\boldsymbol{u} = (u_1, u_2, \ldots, u_n)^T$.

*Definition 4:* The discrete Gaussian curvature $K_i$ on a vertex $v_i \in V$ is defined as the excess angle sum

$$K_i = \begin{cases} 2\pi - \sum_{f_{ijk} \in F} \angle_{kij}, & v_i \notin \partial M \\ \pi - \sum_{f_{ijk} \in F} \angle_{kij}, & v_i \in \partial M \end{cases} \tag{3}$$

where $\angle_{kij}$ represents the corner angle attached to vertex $v_i$ in face $f_{ijk}$ and $\partial M$ is the boundary of the mesh. The discrete Gaussian curvatures are determined by the discrete metrics.

Given a mesh $M = (V, E, F)$, the total Gaussian curvature is a topological invariant. It holds on the mesh as follows:

$$\sum_{v_i \in V} K_i = 2\pi\chi(M) \tag{4}$$

where $\chi(M) = |V| + |F| - |E|$ is the Euler characteristic of $M$. For a mesh with boundaries embedded in either 2-D plane or 3-D Euclidean space, we can compute its flat metric by assigning the total Gaussian curvatures to boundary vertices and keeping the angle sums of all inner vertices as $2\pi$. Therefore, the mesh can be conformally embedded onto the plane with these flat metrics that satisfy the prescribed Gaussian curvatures on boundaries (see Fig. 1 for examples).

The Yamabe problem aims at finding a conformal metric with constant scalar curvature for compact Riemannian manifolds [32], [33]. The Yamabe flow on discrete surfaces has been studied in [30] and [34].

Let $e_{ij}$ be an edge with end vertices $v_i$ and $v_j$, and $d_{ij}$ be the edge length of $e_{ij}$ induced by the Euclidean metric of $\mathbb{R}^3$. The conformally deformed edge length $l_{ij}$ is defined as $l_{ij} = e^{u_i+u_j}d_{ij}$. Let $K_i$ and $\overline{K_i}$ denote the current and the target Gaussian curvatures on vertex $v_i$, respectively. The discrete Yamabe flow is defined as

$$\frac{du_i(t)}{dt} = \overline{K_i} - K_i \tag{5}$$

with initial condition $u_i(0) = 0$. The convergence of Yamabe flow is proven in [34].

The Yamabe flow is the gradient flow of the following Yamabe energy, given as an integration of a differential one-form and proved to be locally convex

$$f(\boldsymbol{u}) = \int_{\mathbf{u}_0}^{\mathbf{u}} \sum_{i}^{n} (\overline{K_i} - K_i) du_i \tag{6}$$

where $\boldsymbol{u}$ and $\boldsymbol{u}_0$ stand for the current and the initial metric vector, respectively. The convex property of the Yamabe energy ensures that the discrete Yamabe flow can quickly and stably converge to the final $\boldsymbol{u}$, which induces the desired metric that satisfies the prescribed target Gaussian curvatures.

Given the high cost to perform centralized algorithms in wireless networks, we employ a distributed implementation of the discrete Yamabe flow to compute desired conformal metrics. Each node only exchanges information with its one-hop neighbors. The number of steps required for the convergence of discrete Yamabe flow depends on both the curvature error threshold $\varepsilon$ and the step length $\delta$. For given $\varepsilon$ and $\delta$, the algorithm convergence time is bounded by $O(-\frac{log\varepsilon}{\delta})$. It is worth mentioning that the discrete Yamabe flow needs no special initialization, unlike other methods such as discrete Ricci flow [35], [36] that must construct a good initial circle packing metric with all acute edge intersection angles.

### B. Algorithm Description

As we have discussed in Section I, mapping a boundary to its convex hull leads to the smallest distortion, and consequently the lowest path stretch, among the mappings under different boundary conditions. To this end, our algorithm first identifies convex hulls of all boundaries of the network and then determines the flat metric under convex hull boundary condition. The Yamabe flow is involved in both of these two steps.

**Algorithm 1:** Distributed Yamabe Flow (Free Boundary)

```
1  u_i = 0
2  if v_i is an interior node then
3      K̄_i = 0;
4      while |K̄_i − K_i| > ε do
5          Exchange u_i with its neighbors;
6          for each neighboring triangle f_ijk do
7              ∠_kij = cos^{-1} (e^{2(u_k+u_i)} + e^{2(u_i+u_j)} − e^{2(u_j+u_k)}) / (2e^{2(u_k+u_i)} e^{2(u_i+u_j)});
8          end
9          K_i = 2π − Σ_{f_ijk} ∠_kij;
10         u_i = u_i + δ(K̄_i − K_i);
11     end
12 end
```

*Convex hull identification*: A node has its local connectivity information only. To determine convex hulls, boundary geometry information must be recovered. This is done by using Yamabe flow, with edge length initialized to be 1. The algorithm is summarized in Algorithm 1 and elaborated below.

The discrete conformal factor $u_i$ associated with node $v_i$ is assigned its initial value of zero. The target Gaussian curvatures of interior nodes are set to zero too. In each step of Yamabe flow, all interior nodes (but not boundary nodes) are involved. Specifically, an interior node $v_i$ collects the discrete conformal factor values from its adjacent neighbors. For each triangle $f_{ijk}$ adjacent with node $v_i$, the corner angle $\angle_{kij}$ can be easily computed based on inverse cos law on $v_i$

$$\angle_{kij} = \cos^{-1} \frac{e^{2(u_k+u_i)} + e^{2(u_i+u_j)} - e^{2(u_j+u_k)}}{2e^{2(u_k+u_i)}e^{2(u_i+u_j)}}.$$

Then, the current discrete Gaussian curvature can be computed as the excess of the total angle sum at $v_i$: $K_i = 2\pi - \sum_{f_{ijk}} \angle_{kij}$. The Yamabe flow stops when for every interior node, the difference between the target Gaussian curvature $\overline{K_i}$ (that is set to zero) and current Gaussian curvature $K_i$ is less than a threshold $\varepsilon$, which can be set relatively high to speed up its convergence since we only need the approximated boundary geometry information. Otherwise, the discrete conformal factor associated with each node needs to be updated as: $u_i = u_i + \delta(\overline{K_i} - K_i)$, where $\delta$ is a constant step size. Yamabe flow is proven to converge [34], with a bounded convergence delay of $O(-\frac{log\varepsilon}{\delta})$. When it stops, a nonboundary edge (e.g., an edge between nodes $v_i$ and $v_j$) has its

conformally deformed edge length of $l_{ij} = e^{u_i+u_j}$; for a boundary edge, its edge length remains 1.

Next, the new metric (i.e., the calculated edge length) is embedded in a 2-D plane via discrete breadth first search. Note that, to determine convex hulls, we do not need a global embedding; instead, local embedding at boundaries (that have been identified in preprocessing) is sufficient. Similar to boundary discovery, a node on a boundary may initiate a probe message that travels around the boundary. It informs the boundary nodes that are $k$ hops apart to start embedding. For each of such selected nodes, e.g., node $v_i$, its coordinates are set to (0,0). Then, it arbitrarily selects one of its adjacent neighbors, e.g., $v_j$, and sets its coordinates to $(0, l_{ij})$. To determine the coordinates of $v_k$, which is adjacent to both $v_i$ and $v_j$, it calculates the intersection points of the two circles with centers at $v_i$ and $v_j$, and radii of $l_{ik}$ and $l_{jk}$, respectively. Then, one of the intersection points that conforms the orientation of triangle $f_{ijk}$ (that has been determined in preprocessing) is chosen as the coordinates of $v_k$. The procedure continues until all nodes within $k$ hops have computed their coordinates.

Note that the above embedding is not greedy yet. Those virtual coordinates are not usable for greedy routing. They are the initial coordinates that reflect the boundary geometric information. Based on them, the convex hull for each boundary can be locally determined by well-known algorithms with local messaging [37].

*Flat metric under convex boundary condition*: With the recovered boundary geometry information, the desired flat metric can be computed with Yamabe flow under convex boundary condition. The procedure is similar to the discussion above for computing the Yamabe flow under free boundary condition. There are two minor differences only. 1) We set the target Gaussian curvatures of all nodes to zero except those on the corners of the convex boundaries. They are set to be approximated convex corner angles. 2) Boundary nodes will now be involved in computation. In each step of the Yamabe flow, the equation to compute current Gaussian curvature at a boundary node is $K_i = \pi - \sum_{f_{ijk}} \angle_{kij}$.

The same method is employed to determine the flat metric, and embed it to a 2-D plane. But note that the embedding here is global. It starts from a node and propagates to the entire network, yielding virtual coordinates for every node. Since this embedding is under convex hull outer boundary and circle-like convex polygon inner boundary conditions, the established virtual coordinates ensure greedy routing in general. There is only one exception that rarely exists in practice, but it is discussed below for completeness of our algorithm. The exception occurs when a node on an inner convex boundary is local minimum for a given destination. Our solution is as follows. A message goes around the inner boundary to collect the virtual coordinates of all nodes on the boundary and compute the average, which serves as the estimated convex center. Then, the message goes around the boundary the second time to disseminate this information. Each node on the boundary computes its distance to the center. Using the minimal distance as the radius of a circle, each boundary node calculates its alternative coordinates by projecting itself onto the circle. The alternative coordinates are used under such rare scenarios only.

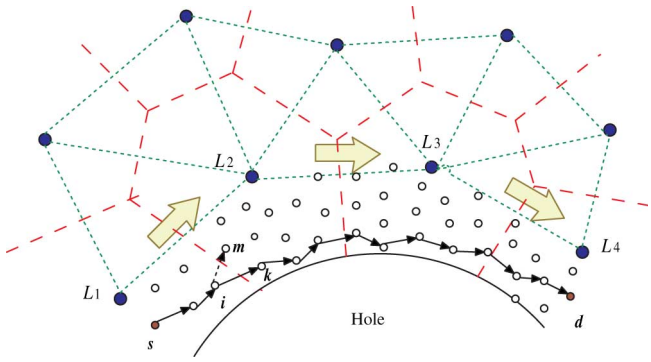



Fig. 4. Greedy routing based on virtual coordinates, where the routing path does not go through the landmarks.

## IV. Routing Based on Virtual Coordinates

Till now each node on the triangular mesh (i.e., a landmark) has obtained its virtual coordinates. Routing between any two such nodes is straightforward by using greedy forwarding. But note that the triangular mesh is only a sampled backbone representation of the network. Other nodes that are not on the mesh must be considered too. To this end, each of them calculates its virtual coordinates in barycentric form, based on three nearby landmarks. More specifically, consider a triangle $T$ defined by three landmarks $v_i$, $v_j$, and $v_k$. Any node $x$ located in this triangle may then be represented as a weighted sum of the three landmarks, i.e., $x = \alpha_1 v_i + \alpha_2 v_j + \alpha_3 v_k$, where $\alpha_1 + \alpha_2 + \alpha_3 = 1$. $\alpha_i$ is also called area coordinates. It is determined as follows. Let us connect $x$ and the three landmarks. The triangle is divided into three sub-triangles. $\alpha_i$ is the ratio between each sub-triangle's area and the whole triangle's area. This method is simple and introduces little distortion as demonstrated in our simulations. In addition, each node keeps the virtual coordinates of its neighboring landmarks and the hop distances to them.

To deliver a packet, the source node first checks the destination's coordinates via a coordinates lookup service which is required in all virtual or physical coordinates-based greedy routing schemes. According to the coordinates of the destination, it chooses a neighboring landmark that is the closest to the destination, and sends its packet toward it. The packet will be forwarded hop by hop, and in each hop, the above procedure repeats. Note that although the packet is forwarded toward a neighboring landmark, the actual routing path does not have to go through the landmark. In general, when a packet becomes close to and before it actually reaches a landmark, an intermediate node may discover a new neighboring landmark that is now the closest to the destination and thus adjust the routing path to bypass the current landmark. This leads to efficient end-to-end routing, as illustrated in Fig. 4.

## V. Simulations and Comparison

We implement our proposed Yamabe flow-based scheme and compare it with the harmonic approach [12], hyperbolic approach [25], Ricci flow-based approach [27], and the shortest path routing. We carried out simulations in wireless sensor networks with quasi-UDG topologies. Among various networks with different topological shapes being investigated, we note that

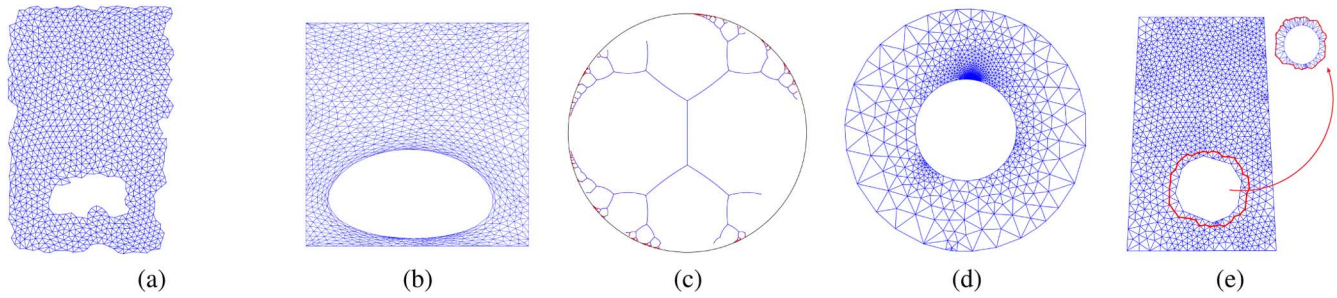



Fig. 5. (a) Original topology with one concave hole. (b) Harmonic embedding with angle and area distortion along the inner boundary. (c) Hyperbolic embedding. (d) Ricci embedding with severe area distortion above the inner circle. (e) Yamabe embedding with well preserved geometry and least area distortion. The part on the right side is a circle embedding for the inner convex hole, which generates alternative coordinates in case two sides of the inner convex boundary are perfectly parallel as discussed in Section III.

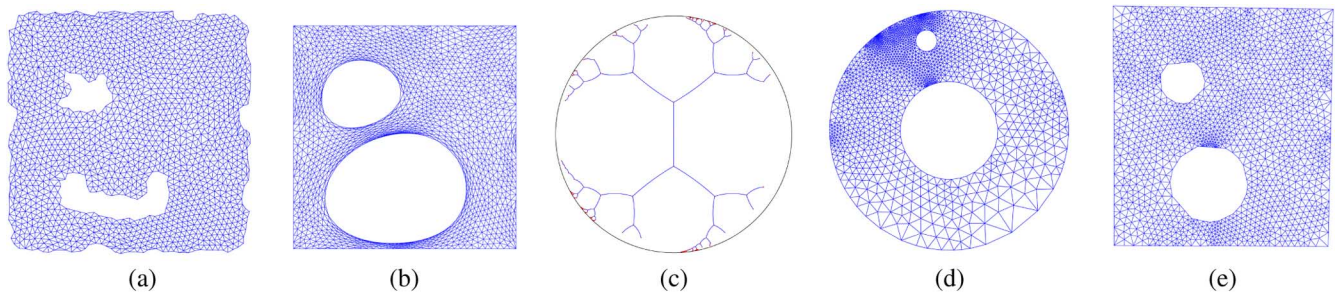



Fig. 6. (a) Original topology with two concave holes. (b) Harmonic embedding. (c) Hyperbolic embedding. (d) Ricci embedding. (e) Yamabe embedding.

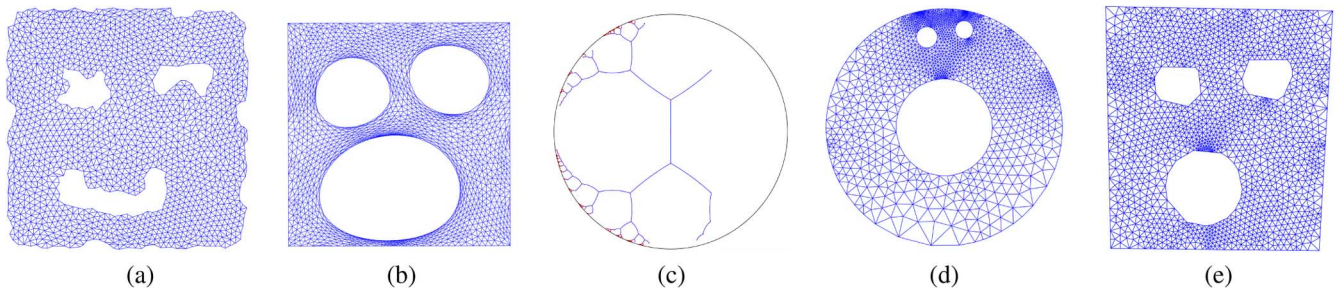



Fig. 7. (a) Original topology with three concave holes. (b) Harmonic embedding. (c) Hyperbolic embedding. (d) Ricci embedding. (e) Yamabe embedding.

the harmonic and hyperbolic approaches may fail in some of them because the algorithms result in edge flips or the required computational precision is beyond the capacity of the computer used for simulation.

### *A. Distortions in Embedding*

First, let us examine several representative networks and the corresponding embedding results under different algorithms.

Fig. 5 is based on a network with one hole at the bottom. There are about 200 000 nodes in the network, among which 1114 nodes are landmarks. For conciseness, only landmarks in the mesh structure (or a tree-like structure for hyperbolic embedding) are shown to demonstrate distortions due to embedding. For the hyperbolic scheme [see Fig. 5(c)], a three-degree tree is employed to realize embedding. We observe that due to the limit in computation precision, we cannot embed all nodes in the network to a Poincaré disk, because with the increase in the depth of the tree, the Euclidean distance between adjacent nodes decreases exponentially. The computation precision soon becomes insufficient to differentiate them (64 bits for double type in C++, i.e., 15 decimals, are used in our simulations). So, we show the nodes that can be embedded only. As can be seen in Fig. 5(d), Ricci embedding yields a great amount of area distortions (see the clustered area at the top of the inner circle), which has severe impact on the routing performance (i.e., stretch factor). The harmonic embedding preserves the relative position of the hole and has less area distortion. But it generates significant angle distortion and some edge distortions along the inner boundary. In addition, it does not avoid local minimums and thus resulting in critical route failures to be discussed later. On the other hand, Yamabe well preserves the geometry of the original network, and has the least area distortion. The separate part on the right side of Fig. 5(e) is a circle embedding for the inner convex hole, which generates alternative coordinates in case a local minimum occurs at a node on the inner convex boundary as discussed in Section III.

Networks with more holes are shown in Figs. 6 and 7. They both have about 290 000 nodes and 1700 landmarks. Harmonic embedding still keeps the relative positions of the holes, but the distortions along the inner boundaries become higher; on the

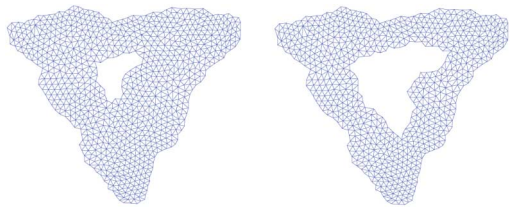


Fig. 8. Two networks with a hole of increasing size.

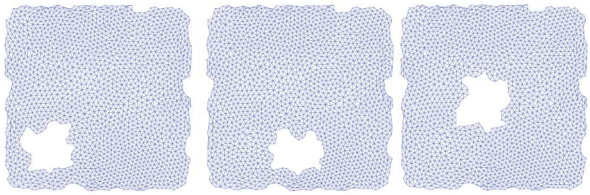


Fig. 9. Three networks with a hole at corner, bottom, and center, respectively.

other hand, no significant changes in distortions are observed in Ricci and Yamabe embedding. In addition, since the hyperbolic approach employs a tree for embedding, the actual geometry has no impact on its results.

### B. Routing Failures

We study the routing success rate by running greedy routing between every pair of nodes in a network. Ricci- and Yamabe-based schemes achieve guaranteed delivery. This has been verified by our simulations that exhibit 100% success rate under these two schemes.

Routing failures are observed under the harmonic scheme. The more the holes, the higher the failure rate. For example, 4.11% routes fail in the network shown in Fig. 6(a), while the failure rate increases to 4.38% in the network given in Fig. 7(a). With more holes introduced into the network, more nodes may become local minimums for any given destination. Since the harmonic scheme does not completely eliminate such local minimums, more greedy routing failures are observed. We have also studied the impact of the size and the position of holes by using the networks shown in Figs. 8 and 9, respectively. The harmonic scheme is very sensitive to the size of the hole. For example, when the hole enlarges as illustrated in Fig. 8, the routing failure rate increases from 5.4% to 15.5%. With the increase in the hole's size, the perimeter of the boundary increases, which results in more local minimums and accordingly more route failures.

For the hyperbolic approach, we only test the routes between those embedded nodes. Theoretically, the hyperbolic embedding is greedy and thus guarantees greedy delivery. However, around 6.13% routing attempts are failed in our simulations. Investigating into such anti-intuitive results, we realize that the failures are again due to the limited computational precision. Adjacent nodes of the constructed spanning tree have equal hyperbolic distances in Poincaré disk, but their Euclidean distances are decreasing almost exponentially. More specifically, the deeper layer a pair of adjacent nodes is from the root of the tree, the closer their Euclidean distance is in Poincaré disk. After embedding around 10 layers of the spanning tree in Poincaré disk, our computer used for simulation fails to distinguish two nodes with different embedding positions. Note that since all hyperbolic models are equivalent, any hyperbolic model equally requires extremely high computational precision for the hyperbolic approach.

TABLE I
AVERAGE ROUTING FAILURES OF DIFFERENT SCHEMES ON VARIOUS NETWORK TOPOLOGIES SHOWN IN FIGS. 5(a), 6(a), 7(a), 8, AND 9

| Schemes | Hyperbolic | Harmonic | Ricci | Yamabe |
|---|---|---|---|---|
| Average routing failure | 6.13% | 5.75% | 0 | 0 |

TABLE II
AVERAGE STRETCH FACTORS OF DIFFERENT SCHEMES

| Scenario | Hyperbolic | Harmonic | Ricci | Yamabe |
|---|---|---|---|---|
| Topology 1 [Fig. 5(a)] | 1.65606 | 1.45715 | 1.83518 | 1.38846 |
| Topology 2 [Fig. 6(a)] | 1.65501 | 1.46334 | 1.54808 | 1.39032 |
| Topology 3 [Fig. 7(a)] | 1.64974 | 1.48278 | 1.55544 | 1.38717 |

Table I summaries the average routing failures of different schemes on various network topologies shown in Figs. 5(a), 6(a), 7(a), 8, and 9.

### C. Stretch Factor

Stretch factor is a key parameter to evaluate the performance of greedy routing. It is defined as the ratio between the path length in greedy routing and the shortest path length. We consider all pairs of nodes (that are greedily routable) in the network and calculate the average stretch factors given in Table II.

The hyperbolic scheme relies on the shortest path tree rooted at *one* node in the network. As a result, two adjacent nodes may have a long routing path, leading to poor stretch factor. We observe that Ricci experiences high and unstable stretch factor. As discussed above, in the first topology [Fig. 5(a)], Ricci exhibits severe distortions, resulting in a stretch factor as high as over 1.8. In the second and third topologies, the distortions are reduced, and thus yielding moderate stretch. The stretch factor under harmonic embedding is stable and low. But recall that it pays the price of many routing failures. Our proposed Yamabe flow-based scheme achieves not only the lowest but also the stablest stretch factor, always between 1.3 and 1.4 in all of our experiments.

The size of the hole does not noticeably affect the stretch factor under Ricci, harmonic, or Yamabe. As a matter of fact, the stretch factor actually decreases slightly in both Ricci and Yamabe-based schemes, when the hole enlarges (see Fig. 8).

As another interesting observation, we find Ricci is sensitive to the position of the hole. For example, consider three positions at corner, bottom, and center illustrated in Fig. 9. Ricci performs better when the hole is at the center (achieving a stretch factor of 1.4), while its stretch factor increases to 1.5 when the hole locates at border or corner. This phenomenon is because Ricci always maps the largest hole (in this case only one hole) to the center. When the original position of the hole is far away from the center, it results in large compression and stretching of the edges around the hole. On the other hand, Yamabe and Harmonic are not affected much by the hole's position.

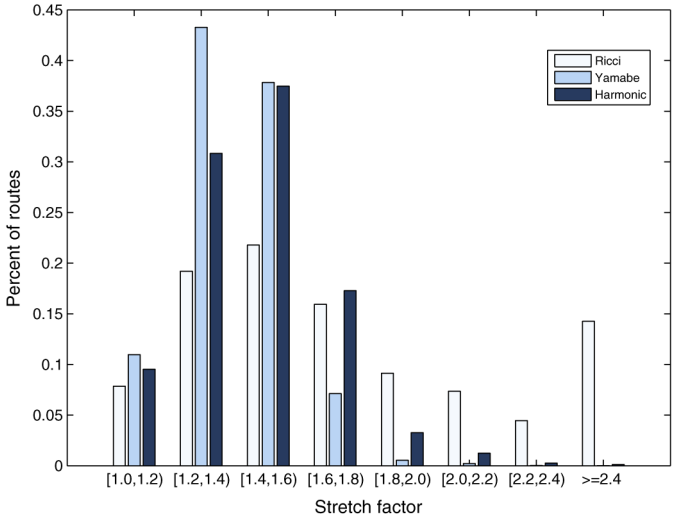



Fig. 10. Stretch factor distribution of Ricci, Yamabe, and harmonic.

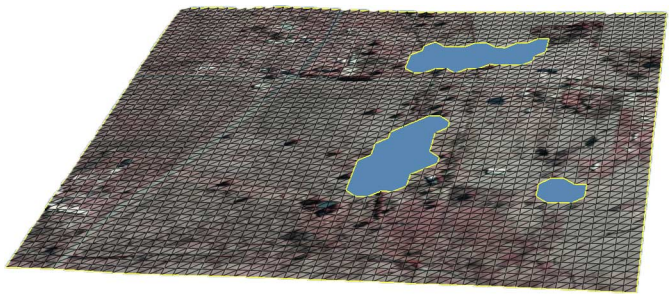


Fig. 11. 3-D surface (UTM zone 15 of North American Datum 1983) in an area of $1\ \mathrm{km} \times 1\ \mathrm{km}$, with a height variation of $40\ \mathrm{m}$ and a resolution of $5\ \mathrm{m}$.

The stretch factor distribution is illustrated in Fig. 10. This is obtained in the network shown in Fig. 5(a). We observe similar statistics in other networks. As can be seen, Ricci results in many routes with their stretch factors greater than 1.8, while the stretch factors of Harmonic and Yamabe are nicely distributed at a lower range.

### *D. 3-D Surface*

To demonstrate our proposed approach on 3-D surfaces, we adopt a real 3-D surface data set (UTM zone 15 of North American Datum 1983). It was distributed by "Atlas: The Louisiana Statewide GIS," LSU CADGIS Research Laboratory, Baton Rouge, LA. We choose an area of $1\ \mathrm{km} \times 1\ \mathrm{km}$, with a height variation of $40\ \mathrm{m}$ and a resolution of $5\ \mathrm{m}$, as shown in Fig. 11. The area includes hills and water ponds. Since the points of the data set are on a grid structure, we assume that the sensors are placed at a grid too, where adjacent nodes are $20\ \mathrm{m}$ apart. It is straightforward to establish a mesh structure and apply the Yamabe flow-based embedding. Our results show a stretch factor of 1.36 and 100% delivery between all pairs of nodes.

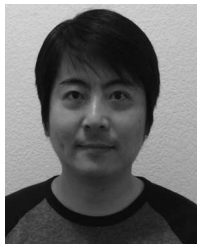
## VI. Conclusion

In this paper, we have proposed a distributed Yamabe flow-based scheme to enable greedy routing in large scale wireless networks deployed on 2-D and 3-D surfaces. It identifies the convex hull of each boundary and employs Yamabe flow to compute flat metric under convex hull boundary condition to establish virtual coordinates. Such virtual coordinates are used for greedy routing. Since the proposed embedding algorithm maps the outer boundary to a convex shape and an interior concave void to a circle-like convex polygon, it effectively eliminates local minimum and attains guaranteed delivery. At the same time, it introduces a small distortion only and consequently achieves a low stretch factor. Our simulations have shown that its stretch factor is lower than any existing greedy embedding algorithms. Moreover, the proposed scheme is merely based on local connectivity and consumes a small constant storage, thus scaling to arbitrarily large networks.

## References

[1] I. Stojmenovic, "Machine-to-machine communications with in-network data aggregation, processing and actuation for large scale cyber-physical systems," *IEEE Internet Things J.*, vol. 1, no. 2, pp. 122–128, Apr. 2014.
[2] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," in *Proc. 3rd Int. Workshop Discr. Algorithms Methods Mobile Comput. Commun.*, 1999, pp. 48–55.
[3] B. Karp and H. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proc. ACM/IEEE Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2001, pp. 1–12.
[4] E. Kranakis, H. Singh, and J. Urrutia, "Compass routing on geometric networks," in *Proc. 11th Can. Conf. Comput. Geom. (CCCG'99)*, 1999, pp. 51–54.
[5] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, "Geometric ad-hoc routing: Theory and practice," in *Proc. 22nd ACM Int. Symp. Principles Distrib. Comput. (PODC'03)*, 2003, pp. 63–72.
[6] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Worst-case optimal and average-case efficient geometric ad-hoc routing," in *Proc. ACM Int. Symp. Mobile Ad hoc Netw. Comput. (MobiHOC)*, 2003, pp. 267–278.
[7] B. L. S. Mitra and B. Liskov, "Path vector face routing: Geographic routing with local face information," in *Proc. IEEE Int. Conf. Netw. Protocols (ICNP'05)*, 2005, pp. 147–158.
[8] H. Frey and I. Stojmenovic, "On delivery guarantees of face and combined greedy-face routing in ad hoc and sensor networks," in *Proc. ACM/IEEE Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2006, pp. 390–401.
[9] G. Tan, M. Bertier, and A.-M. Kermarrec, "Visibility-graph-based shortest-path geographic routing in sensor networks," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2009, pp. 1719–1727.
[10] S. Funke and N. Milosavljevic, "Guaranteed-delivery geographic routing under uncertain node locations," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2007, pp. 1244–1252.
[11] C. Liu and J. Wu, "Efficient geometric routing in three dimensional ad hoc networks," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, pp. 2751–2755, 2009.
[12] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, "Geographic routing without location information," in *Proc. ACM/IEEE Int. Conf. Mobile Comput. Netw. (MobiCom)*, 2003, pp. 96–108.
[13] 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," in *Proc. 2nd USENIX/ACM Symp. Netw. Syst. Des. Implement. (NSDI'05)*, 2005, pp. 329–342.
[14] Y. Zhao, B. Li, Q. Zhang, Y. Chen, and W. Zhu, "Efficient hop ID based routing for sparse ad hoc networks," in *Proc. IEEE Int. Conf. Netw. Protocols (ICNP'05)*, 2005, pp. 179–190.
[15] Q. Cao and T. Abdelzaher, "Scalable logical coordinates framework for routing in wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 2, no. 4, pp. 557–593, 2006.
[16] A. Caruso, S. Chessa, S. De, and A. Urpi, "GPS free coordinate assignment and routing in wireless sensor networks," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2005, pp. 150–160.
[17] S. Tao, A. Ananda, and M. C. Chan, "Greedy hop distance routing using tree recovery on wireless ad hoc and sensor networks," in *Proc. IEEE Int. Conf. Commun. (ICC'08)*, 2008, pp. 2712–2716.
[18] M. J. Tsai, H. Y. Yang, and W. Q. Huang, "Axis based virtual coordinate assignment protocol and delivery guaranteed routing protocol in wireless sensor networks," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2007, pp. 2234–2242.

[19] K. Liu and N. Abu-Ghazaleh, "Stateless and guaranteed geometric routing on virtual coordinate systems," in *Proc. 5th IEEE Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS'08)*, 2008, pp. 340–346.
[20] Q. Fang, J. Gao, L. J. Guibas, V. Silva, and L. Zhang, "GLIDER: Gradient landmark-based distributed routing for sensor networks," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2005, pp. 339–350.
[21] G. Tan, M. Bertier, and A.-M. Kermarrec, "Convex partition of sensor networks and its use in virtual coordinate geographic routing," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2009, pp. 1746–1754.
[22] C. Papadimitriou and D. Ratajczak, "On a conjecture related to geometric routing," *Theor. Comput. Sci.*, vol. 344, no. 1, pp. 3–14, 2005.
[23] P. Angelini, F. Frati, and L. Grilli, "An algorithm to construct greedy drawings of triangulations," in *Proc. 16th Int. Symp. Graph Drawing*, 2008, pp. 26–37.
[24] T. Leighton and A. Moitra, "Some results on greedy embeddings in metric spaces," in *Proc. 49th IEEE Annu. Symp. Found. Comput. Sci.*, 2008, pp. 337–346.
[25] R. Kleinberg, "Geographic routing using hyperbolic space," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2007, pp. 1902–1909.
[26] A. Cvetkovski and M. Crovella, "Hyperbolic embedding and routing for dynamic graphs," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2009, pp. 1647–1655.
[27] R. Sarkar, X. Yin, J. Gao, F. Luo, and X. D. Gu, "Greedy routing with guaranteed delivery using Ricci flows," in *Proc. 8th Int. Symp. Inf. Process. Sensor Netw. (IPSN'09)*, Apr. 2009, pp. 121–132.
[28] R. Flury, S. Pemmaraju, and R. Wattenhofer, "Greedy routing with bounded stretch," in *Proc. IEEE Int. Conf. Comput. Commun, (INFOCOM)*, 2009, pp. 1737–1745.
[29] Y.-L. Yang, J. Kim, F. Luo, S.-M. Hu, and X. Gu, "Optimal surface parameterization using inverse curvature map," *IEEE Trans. Vis. Comput. Graph.*, vol. 14, no. 5, pp. 1054–1066, Sep./Oct. 2008.
[30] B. Springborn, P. Schröder, and U. Pinkall, "Conformal equivalence of triangle meshes," in *ACM SIGGRAPH*, 2008, pp. 1–11.
[31] S. Funke and N. Milosavljevi, "How much geometry hides in connectivity? —Part II," in *Proc. 18th Annu. ACM-SIAM Symp. Discr. Algorithms (SODA'07)*, 2007, pp. 958–967.
[32] H. Yamabe, "The Yamabe problem," *Osaka Math. J.*, vol. 12, no. 1, pp. 21–37, 1960.
[33] J. M. Lee and T. H. Parker, "The Yamabe problem," *Bull. Amer. Math. Soc.*, vol. 17, no. 1, pp. 37–91, 1987.
[34] F. Luo, "Combinatorial Yamabe flow on surfaces," *Commun. Contemp. Math.*, vol. 6, no. 5, pp. 765–780, 2004.
[35] B. Chow and F. Luo, "Combinatorial Ricci flows on surfaces," *J. Differ. Geom.*, vol. 63, no. 1, pp. 97–129, 2003.
[36] M. Jin, J. Kim, F. Luo, and X. Gu, "Discrete surface Ricci flow," *IEEE Trans. Vis. Comput. Graph.*, vol. 14, no. 5, pp. 1030–1043, Sep./Oct. 2008.
[37] M. Berg, O. Cheong, M. Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd ed. New York, NY, USA: Springer-Verlag, 2008.



**Su Xia** received the B.S. and M.S. degrees in radio engineering and computer science from Southeast University, Nanjing, China, in 1998 and 2001, respectively, and the Ph.D. degree in computer science from the Center for Advanced Computer Studies (CACS), University of Louisiana at Lafayette, LA, USA, in 2012.

Currently, he is working with the Internet of Things (IoT) Group, Cisco System Inc., Milpitas, CA, USA. His research interest focused on geometric routing in wireless sensor networks, especially in 3-D sensor networks.

**Hongyi Wu** received the B.S. degree in scientific instruments from Zhejiang University, Hangzhou, China, in 1996, and the M.S. degree in electrical engineering and Ph.D. degree in computer science from the State University of New York (SUNY) at Buffalo, New York, NY, USA, in 2000 and 2002, respectively.

Since 2002, he has been with the Center for Advanced Computer Studies (CACS), University of Louisiana at Lafayette (UL Lafayette), Lafayette, LA, USA, where he is now a Professor and holds the Alfred and Helen Lamson Endowed Professorship in Computer Science. His research includes delay-tolerant networks, radio frequency identification (RFID) systems, wireless sensor networks, and integrated heterogeneous wireless systems.

Dr. Wu was the recipient of the NSF CAREER Award in 2004 and the UL Lafayette Distinguished Professor Award in 2011.

**Miao Jin** received the B.S. degree from the Beijing University of Posts and Telecommunication, Beijing, China, in 2000, and the M.S. and Ph.D. degrees in computer science from the State University of New York at Stony Brook, New York, NY, USA, in 2008 and 2006, respectively.

She has been an Assistant Professor with the Center for Advanced Computer Studies, University of Louisiana, Lafayette, LA, USA, since Fall 2008. Her research interests include computational geometric and topological algorithms with applications in wireless sensor networks, computer graphics, computer vision, geometric modeling, and medical imaging. Her research results have been used as cover images of mathematics books and licensed by Siemens Healthcare Sector of Germany for virtual colonoscopy.

Dr. Jin was the recipient of the NSF CAREER Award in 2011.