Spherical Representation and Polyhedron Routing for Load Balancing in Wireless Sensor Networks

Xiaokang Yu* Xiaomeng Ban† Wei Zeng‡ Rik Sarkar† Xianfeng Gu† Jie Gao†

* Department of Computer Science, Shandong University, P.R. China. xiaokang22@gmail.com
† Department of Computer Science, Stony Brook University, USA. {xban, rik, gu, jgao}@cs.sunysb.edu
‡ Department of Computer Science, Wayne State University, USA. zeng@wayne.edu

***Abstract*—In this paper we address the problem of scalable and load balanced routing for wireless sensor networks. Motivated by the analog of the continuous setting that geodesic routing on a sphere gives perfect load balancing, we embed sensor nodes on a convex polyhedron in 3D and use greedy routing to deliver messages between any pair of nodes with guaranteed success. This embedding is known to *exist* by the Koebe-Andreev-Thurston Theorem for any 3-connected planar graphs. In our paper we use discrete Ricci flow to develop a distributed algorithm to *compute* this embedding. Further, such an embedding is not unique and differs from one another by a Möbius transformation. We employ an optimization routine to look for the Möbius transformation such that the nodes are spread on the polyhedron as uniformly as possible. We evaluated the load balancing property of this greedy routing scheme and showed favorable comparison with previous schemes.**

I. Introduction

In large scale sensor networks it is critical to balance out work load on different sensors, to prevent some nodes running out of battery immaturely. In terms of routing, we hope that the selected routes are 'load balanced' such that the node residual battery levels are as uniform as possible. There has been a lot of work in the literature about load balanced routing. One formulation is to select route such that the maximum load is minimized. Unfortunately finding the optimal solution for this problem is NP-hard (modeled as unsplittable flow problem) even in very simple networks (such as grid). There have been approximate algorithms [16], [17] developed for the problem. But they all require centralized knowledge, and thus are inappropriate for a distributed setting.

In this paper we focus on the load balancing issue of *greedy routing* solutions, where nodes deliver messages by forwarding to the neighbor closer to the destination under some distance metric [10], [4]. Greedy routing may not always be successful if there is a node whose neighbors are all further away from the destination. In the past few years various schemes have been proposed to find an embedding and a related distance function such that greedy routing guarantees delivery [14], [6], [12], [1], [18], [7], [11], [20], [2]. But these methods have nearly no consideration of load balancing.

In this paper we design load balanced, greedy routing solutions by first examining how load balancing relates to the network 'geometry'. Take an example with sensors uniformly spread in a disk or a square and consider shortest path routing (or geographical greedy routing [10], [4]) for all pairs, the nodes near the center carry more traffic than the nodes near the boundary. In general, for a surface in 3D, the points with negative curvatures (the saddle points) attract geodesic paths, while points with positive curvatures repel geodesic paths. Therefore surface curvature and geometry are intrinsically related to load distribution in routing.

Existing load balanced, greedy routing methods. The idea of examining network geometry and changing network shape for better load balancing has only been done on networks of special topology. In [8], a greedy routing scheme that achieves both constant routing stretch factor (compared with shortest path routing) and constant load balancing ratio (compared with the optimal load balanced routing) is proposed, but only for wireless nodes distributed in a narrow strip. Mei and Stefa [13] studied load balancing for a regular square shape sensor network and proposed to use the 'outer space' by essentially wrapping up the network into a torus. Routing is done in a greedy manner by using the coordinates on the torus, possibly bounding off from the boundary. The method depends on geographical coordinates and thus does not guarantee delivery. Popa *et al.* [15] examined load balanced routing for a disk shape network and proposed to use stereographic projection to map the network on a hemisphere. Routing is guided by the spherical distance in a greedy manner. Improved load balancing is shown as the routes are made to 'curve' around the network center. Again, delivery is not guaranteed. These algorithms are for special cases. It is unclear how they apply to irregular networks possibly with holes.

Our approach: spherical embedding. For a surface with positive constant curvature everywhere, i.e., a sphere, the shortest paths have uniform distribution and greedy routing (in terms of spherical distance) on the sphere has guaranteed delivery and perfect load balancing. Motivated by this, we investigate the mapping of a sensor network to a spherical metric. The well-known Koebe-Andreev-Thurston Theorem describes the *spherical embedding* of a 3-connected planar graph, which can be seen as a convex polyhedron with all edges tangent to a unit sphere. Each vertex is associated with a circle on the sphere. The circles of adjacent vertices are tangent to each other. See Figure 3. With the spherical embedding, the distance function $d(u, v) = -c(u) \cdot c(v)$, where $c(u)$ is the 3D coordinate of u, has guaranteed delivery for all pairs of vertices [14]. This is referred to as *polyhedron routing*.

Our contributions in this paper are two folds. First we proposed a distributed algorithm to compute the spherical embedding for a general 3-connected planar graph using discrete Ricci flow. Previous algorithms either require a triangulation as input [19], [5] or a centralized computation [3]. Second, the spherical embedding is not unique and differs from one another by a Möbius transformation. Thus we exploit this degree of freedom and find the spherical embedding that favors load balancing. By simulations we show that this idea works well in practice and compares favorably with previous load balanced routing schemes that also use greedy routing ideas [13], [15].

Jonckheere *et al.* [9] examined the negative curvature and load congestion in both artificial data and real networks. They proposed to use Yamabe flow with free boundary condition to alleviate network congestion. We remark that the relationship of the network 'curvature' and routing congestion remains as an interesting open problem.

II. Spherical Representation

A. Circle Packing and Spherical Representation

A *circle packing* is a connected collection of circles on some (Riemann) surface, whose interiors are disjoint. The intersection graph (the tangency graph or contact graph) of a circle packing is the graph having a vertex for each circle, and an edge for every pair of circles that are tangent.

Suppose G is a 3-connected planar graph, and $\tilde{G}$ is the dual graph[1] of G, the following theorem shows the existence of a pair of circle packings for G and $\tilde{G}$ respectively.

Theorem 2.1 (The Koebe-Andreev-Thurston Theorem): There is a pair of circle packing $P, \tilde{P}$, where the intersection graph of P and $\tilde{P}$ are isomorphic to G and $\tilde{G}$ respectively. Furthermore, for any vertex $v \in G$ and an adjacent face f, the vertex circle in P is orthogonal to the face circle in $\tilde{P}$.

Corollary 2.2 (Spherical Representation): Each 3-connected planar graph can be realized by a 3-polytope which has all edges tangent to the unit sphere. The realization is unique up to Möbius transformations.

A Möbius transformation is a map that maps a complex plane to itself, $f(z) = \frac{az+b}{cz+d}$, where a, b, c, d are four complex numbers satisfying $ad - bc = 1$. A Möbius transformation is a conformal map and maps circles to circles.

B. Algorithm for Computing Spherical Representation

Step 0: Extract a Planar Subgraph. Algorithms for extracting a planar graph from the communication graph have been developed in the past literature, see the references in [18]. In our implementation, we have used the restricted Delaunay graph approach [18].

Step 1: Compute the Dual & Overlap Graph. Given a planar 3-connected graph G, denote the k-th vertex as k, the j-th face as f_j. Note that, one face is the infinite face. The overlap graph

[1]Each face of G is a vertex of $\tilde{G}$. Each vertex of G is a face of $\tilde{G}$. An edge connecting two vertices of $\tilde{G}$ if and only if there is an edge shared by the corresponding faces in G.

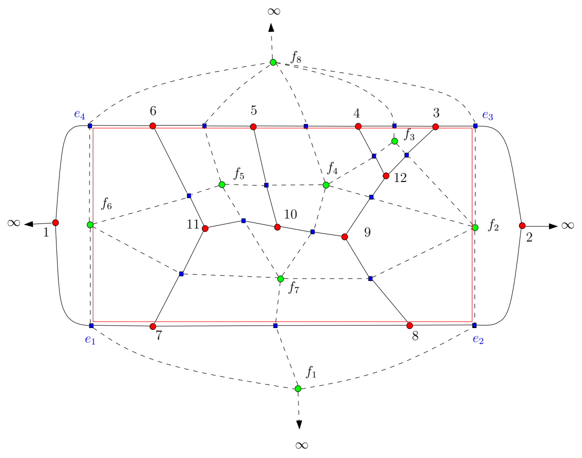


Fig. 1. An example of the reduced graph.

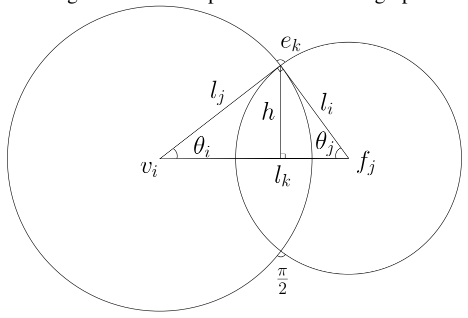


Fig. 2. Step 4. Ricci flow

D is defined as $G \cup \tilde{G}$, where $\tilde{G}$ is the dual graph of G. On the overlap graph D, there are three types of nodes:

1) Vertex node v_i: corresponding to a vertex in G, is represented as a red round dot.
2) Face node f_j: corresponding to a face in G, is represented as a green round dot.
3) Edge node e_{ij}^{kl}: corresponding to an intersection of an edge $[v_i, v_j]$ in G, and an edge $[f_k, f_l]$ in the dual graph $\tilde{G}$, where $[v_i, v_j]$ is the common edge of the faces f_k and f_l, represented as a blue square.

Each facet on the overlapped graph D is a topological quadrilateral, we denote the quadrilateral as $\square(v_i, f_j)$.

Step 2: Select an Infinity Edge Node. Select one edge node to be mapped to the infinity point. We call it the *infinity edge node* and denote it as e_∞.

Step 3: Compute the Reduced Graph $\bar{G}$. Suppose the infinity edge node is given by $e_\infty = [v_i, v_j] \cap [f_k, f_l]$, then we remove all the quadrilateral facets adjacent to v_i, v_j or f_k, f_l in the overlap graph D, to get the reduced graph $\bar{G}$. Figure 1 shows the reduced graph in which the edge node $e_\infty = [v_1, v_2] \cap [f_1, f_8]$ is selected as the infinity edge node. The boundary of the reduced graphs are labeled as the red edges

Step 4: Run Discrete Ricci Flow. On the reduced graph $\bar{G}$, each quadrilateral facet $\square(v_i, f_j)$ is triangulated by adding one virtual edge $[v_i, f_j]$ connecting the vertex node v_i and the face node f_j. Then we run discrete Ricci flow on the triangulated reduced graph in the following way. Each triangle on the triangulated reduced graph has one vertex node v_i, one face node f_j and one edge node e_k, $[v_i, f_j, e_k]$, as shown in

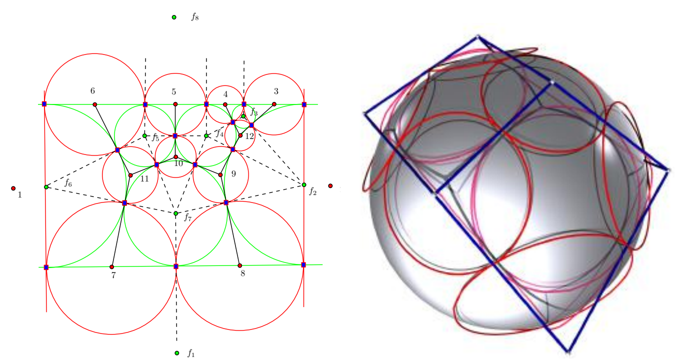


Fig. 3. The circle packing embedded in the plane and the spherical presentation of the graph and convex polytope realization in 3D.

Figure 2. In particular, For each vertex node v_i, we associate it with a circle $C(v_i, \gamma_i)$, centered at v_i with radius γ_i; For each face node f_j, we associate it with a circle $C(f_j, \gamma_j)$; For each edge node e_k, we associate it with a circle with zero radius all the time, $C(e_k, 0)$; The vertex node circle and the face node circle intersect at a right angle; The intersection angle between the edge node circle with other circles are zeros.

Therefore, from Figure 2, $l_i = \gamma_j, l_j = \gamma_i, l_k = \sqrt{\gamma_i^2 + \gamma_j^2}, \theta_k = \frac{\pi}{2}$. For both vertex nodes and face nodes with circle radius γ, let $u = \log \gamma$, then the discrete Ricci flow is given by the following differential equation: $\frac{du_i}{dt} = -K_i$, where K_i is the *discrete Gaussian curvature* at a vertex v_i, defined as the angle deficit at v_i. Specifically, the Ricci flow algorithm is carried out in a distributed manner. Each vertex v_i modifies its radius γ_i with respect to the current curvature. In an iterative way, the radii are adjusted until the curvature becomes sufficiently close to 0. The proof that this version of the Ricci flow indeed produces a pair of circle packings is omitted from this abstract due to space limitation.

We then flatten the reduced graph triangle by triangle and compute the circle packing embedded in the plane, as shown in Figure 3. Suppose the infinity edge node is $e_\infty = [v_i, v_j] \cap [f_k, f_l]$, then the vertex node circles of v_i and v_j are mapped to vertical lines, the face node circles of f_k and f_l are mapped to horizontal lines.

Step 5. Stereo-graphic Projection. Finally, we use stereograph projection $\phi : (u, v) \to (x, y, z)$ to map the circle packing on the plane to the sphere, $\phi(u, v) = \left(\frac{2u}{1+u^2+v^2}, \frac{2v}{1+u^2+v^2}, \frac{-1+u^2+v^2}{1+u^2+v^2}\right)$. For each face node circle $C(f_i, \gamma_i)$ on the plane, we compute its image of the stereographic projection $\phi(C(f_i, \gamma_i))$, which is a circle in $\mathbb{R}^3$. We can compute the planes through all the spherical face node circles. The convex hull bounded by these planes is the convex polytope P. Figure 3 shows the spherical representation of the graph and the induced convex polytope. The convex polytope will be used for our routing purpose.

Communication Costs. The majority of computation and communication costs is for the discrete Ricci flow algorithm, as all other steps can be handled with only constant cost per node. The curvature error decreases exponentially fast. Therefore, the number of steps to reach the desired curvature error bound ε is given by $O(\frac{\log(1/\varepsilon)}{\delta})$, where δ is the step size in the Ricci flow algorithm. The total communication cost is thus $O(\frac{n \log(1/\varepsilon)}{\delta})$.

C. Polyhedron Routing

Given source v_i and destination v_j, polyhedron routing is a greedy routing method with distance function $d(v_i, v_j) = -c(v_i) \cdot c(v_j)$, where $c(v_i)$ is the 3D coordinate of v_i in the spherical representation. v_i delivers the message to the neighbor closer to v_j. To see that this polyhedron routing guarantees delivery, we note that the distance function $d(v_i, v_j)$ is essentially the projection of the vector $\overrightarrow{v}_i$ on the vector $-\overrightarrow{v}_j$. $d(v_i, v_j)$ is clearly a linear function of v_j and achieves the global minimum when $v_i = v_j$. The function can not have a local minimum as for a linear function any local minimum is also the global minimum.

III. Load Balancing

The spherical representation of the 3-connected graph is not unique. They differ by a Möbius transformation. All these spherical representations guarantee successful delivery with polyhedron routing. But they give different load balancing properties. In the following we look for the one that spreads the vertices such that the vertex circles are proportional to their residual battery levels.

For a discrete finite network, the network outer boundary is a large face. We map it to the equator of a unit sphere, such that all the vertices are then mapped to a hemisphere. For that, we apply a *circular reflection*, i.e., a special case of Möbius transformation that maps the points inside a circle C with center $\mathbf{c}$ and radius r to the points outside, and vice versa: $\tau(z) = \mathbf{c} + \frac{\gamma^2}{\bar{z} - \bar{\mathbf{c}}}$.

We first translate and scale the exterior face node circle $C(f_\infty, \gamma_\infty)$ to the unit circle. Then all the other face circles are mapped to the exterior of the unit disk. Then we use circular reflection to map the exterior of the unit circle to the interior. Then we use the Möbius transformation to map the interior of the unit disk to itself. All such kind of maps are parameterized by an interior point of the unit disk z_0, $\eta_{\theta, z_0}(z) = e^{i\theta} \frac{z - z_0}{1 - \bar{z}_0 z}$. The choice of η is to make the spherical vertex node circle area proportional to the residual battery level.

Because the graph is 3-connected, each vertex circle C_v goes through at least 3 edge nodes, denoted as $\{e_1, e_2, e_3\}$. The stereo-graphic projection $\phi : \mathbb{C} \to \mathbb{S}^2$ maps them to points on the unit sphere $\{\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3\}$, $\mathbf{s}_k = \phi(e_k)$. These three points determine a plane in $\mathbb{R}^3$, the normal is given by $\mathbf{n} = \frac{(\mathbf{s}_1 - \mathbf{s}_3) \times (\mathbf{s}_1 - \mathbf{s}_2)}{|(\mathbf{s}_1 - \mathbf{s}_3) \times (\mathbf{s}_1 - \mathbf{s}_2)|}$, the plane is given by $\pi_v : \langle \mathbf{p} - \mathbf{s}_1, \mathbf{n} \rangle = 0$. The distance from the origin to the plane π_v is given by $d_v = \langle \mathbf{s}_1, \mathbf{n} \rangle$.

Now we want to find the Möbius transformation such that all the vertex radii are as uniform as possible, by minimizing the energy $E(\theta, z_0) = \sum_{v \in G} (d_v - \bar{d})^2$, where $\bar{d} = \sum_v d_v / n$, n is the total number of nodes in G. We can also use the following energy to consider the vertex residual battery $E(\theta, z_0) = \sum_{v \in G} (d_v - w\bar{d})^2$, where $w = B_v / \bar{B}$, B_v is the

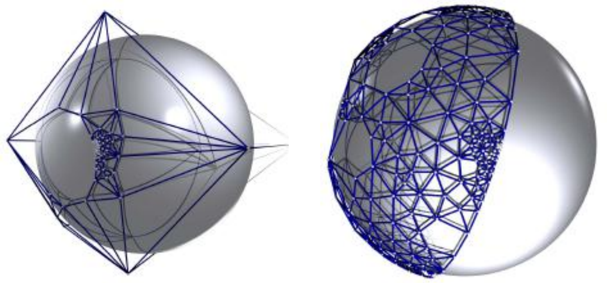


Fig. 4. The spherical embedding before and after the Möbius optimization. Nodes residual battery levels are assumed to be uniform in this case.

residual battery power of node v, and $\bar{B}$ is the average battery level of all nodes.

The energy $E(\theta, z_0)$ is highly non-linear and has multiple local optima. Direct gradient descend method does not guarantee to reach the global optimum. We randomly choose initial seeds and use gradient descend method to reach the optimum in the neighborhood of each seed. Figure 4 shows the spherical representation before and after the Möbius optimization.

IV. Simulations

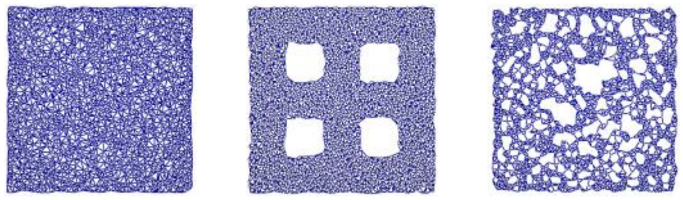
We conducted extensive simulations to evaluate routing properties of our method and compared with previously published methods, namely, curveball routing [15] and outer space routing [13]. We also included simple greedy routing as a baseline comparison scheme.

Curveball and Outer Space Routing. In curveball routing [15], The entire network is mapped to a sphere using stereographic projection. The network is assumed to be of the shape of a disk of some radius R. In outer space routing [13], the network is assumed to be in the shape of a square in the first quadrant. The quadrupled network therefore has the topology of a torus. Neither of these methods guarantee delivery. The presence of holes causes these methods to fail easily.

Summary of Simulation Results. Our main observations are:

- The polyhedron routing is the only one that has 100% delivery guarantee. Other methods in consideration perform progressively worse as the density decreases and the number of holes increases.
- Polyhedron routing is better in distributing loads than other methods, closely followed by curveball.
- Outer space routing balances traffic loads by creating high loads everywhere.
- Polyhedron routing is the only method that can adapt to specific requirements of a network, such as variations in density and variations in available energy (battery levels) across the network.

Delivery Guarantees. Different routing methods have different success rates in delivering messages, depending on the structure of the network. In a simple dense network such as the one in Figure 5(a), all the methods have 100% success rate. In networks with more complex topology such as Fig 5(b) or sparse Fig 5(b), the success rates vary. The results are shown in Table I. This shows that polyhedron routing is the only one



(a) A dense Network. (b) A network with holes. (c) A sparse network.

Fig. 5. Experimental networks. (a) Dense Network. 1850 nodes, avg. degree 14.88 (b) Dense network with large holes. 2100 nodes, avg degree 12.14 (c) Sparse network. 1774 nodes, avg. degree 3.32.

with perfect delivery guarantee. Outer space routing is the worst. The long travel path increases the chances of failure.

Load balancing becomes difficult to compare when messages are not always delivered. In the rest of this section, we will consider only the dense network in Figure 5(a), where messages are delivered by all the methods, to ensure a fair comparison.

	polyhedron	curveball	outer space	simple greedy
Fig 5(b)	100%	95.42%	95.76%	93.94%
Fig 5(c)	100%	61.14%	50.32%	57.16%

TABLE I
Success rates on networks shown in Fig 5(b), (c).

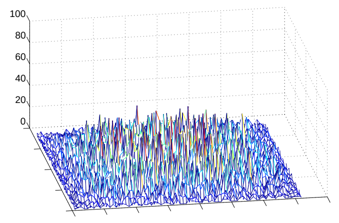
Load balancing. For a set of 5000 random routing attempts, we monitored the traffic load at each node. The polyhedron routing comes out superior to others in the experiment.

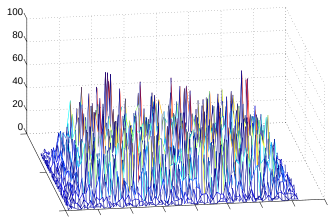
	simple greedy	outer space	curveball	Polyhedron
average load	25.39%	46.65%	25.85%	26.96%

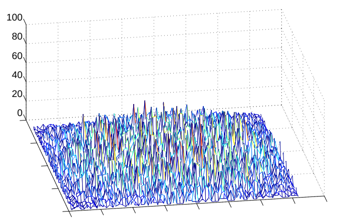
TABLE II
Mean loads on nodes for different routing methods.

The situation becomes clearer in Figure 6 and Table II. In a dense network, greedy routing behaves almost like shortest path routing, therefore has the smallest average load, but crowds the center, where nodes have high load. Outer space routing has the most uniform load, but at the cost of increasing the load for everyone. The average load is 80% more than that of greedy routing. This happens because the outer space routing algorithm balances load by making messages travel via very long paths. This drawback is also apparent from the plot 6(b). Curveball routing is better than either of the two. Its average load is incrementally (2%) higher than greedy routing, but still has a crowded center. Finally, the polyhedron method has slightly higher (6% of greedy routing) average load, implying it takes paths that are slightly longer, but makes use of this flexibility to get better load balancing than curveball routing (Figure 6(d)).

Route adjustment to battery and density levels. A network is not always uniform. The density of nodes may vary in different regions. As the network runs, non-uniform energy consumption may result in varying levels of residual battery in different parts of the network. We consider the network in Figure 7(a), where battery levels drop from left to right. Then we compute a polyhedral embedding optimized for residual battery, where nodes with higher residual battery level are


(a) Simple Greedy Routing.


(b) Outer Space Routing.


(c) Curveball Routing

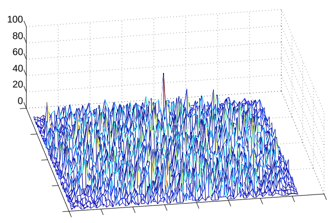

(d) Polyhedron Routing.

Fig. 6. Plots of load distribution.

given larger areas, therefore expected to take up more load. Figure 7(c) shows 3 routes as found by polyhedral routing

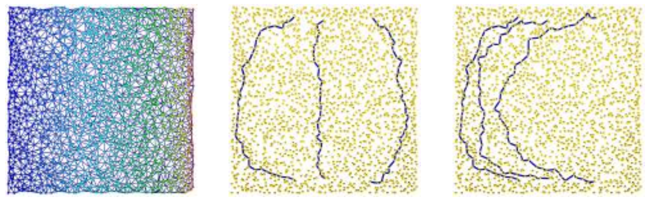
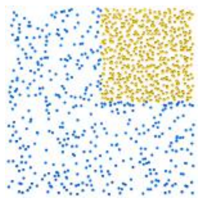
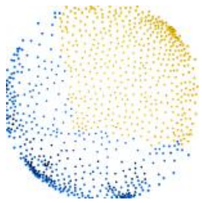


Fig. 7. Battery based optimization. [Left]: the distribution of battery levels (decreasing from left to right). Red indicates lowest battery. After optimizing the routes (shown in [Right]) tend to pass through high battery region, opposed to routes without such optimization (in [Middle]).

when the embedding has been optimized with respect to battery levels. In this case, the routes clearly tend to pass through regions of higher battery levels. Figure 7(b) shows the corresponding routes when the embedding does not consider battery levels. The corresponding routes found by curveball routing are similar to Figure 7(b) and those found by simple greedy routing resemble straight lines. Therefore both these tend to empty the energy of already weak nodes on the right. We omit these figures due to the lack of space.

A similar concern is the variation in density. We would in general like routes to pass through regions of high density where there are more nodes, so that the load per node is lower. Figure 8 shows that a dense region in the network can be embedded such that it is larger, therefore handling more routes.


(a) Network with a dense area (shaded nodes)


(b) Embedding on sphere: dense region expands to take up more space.

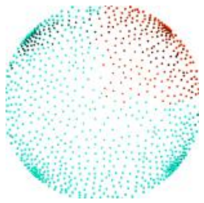

(c) Embedding of a uniformly dense network: a region of same area takes up less space.

Fig. 8. Embedding optimization w.r.t density.

Future Work. How to rigorously characterize the load balancing property induced by the spherical representation and what is the limitation of load balancing by using conformal maps, or by using greedy embeddings, are interesting open problems arising from this paper.

References

[1] P. Angelini, F. Frati, and L. Grilli, "An algorithm to construct greedy drawings of triangulations," in *Proc. of the 16th International Symposium on Graph Drawing*, 2008, pp. 26–37.

[2] M. Ben-Chen, C. Gotsman, and C. Wormser, "Distributed computation of virtual coordinates," in *Symposium on Computational Geometry*, 2007, pp. 210–219.

[3] A. I. Bobenko and B. A. Springborn, "Variational principles for circle patterns and koebe's theorem," *Trans. Amer. Math. Soc*, vol. 356, pp. 659–689, 2004.

[4] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," *Wireless Networks*, vol. 7, no. 6, pp. 609–616, 2001. [Online]. Available: citeseer.ist.psu.edu/bose01routing.html

[5] C. R. Collins and K. Stephenson, "A circle packing algorithm," *Comput. Geom. Theory Appl.*, vol. 25, no. 3, pp. 233–256, 2003.

[6] R. Dhandapani, "Greedy drawings of triangulations," in *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, 2008.

[7] R. Flury, S. Pemmaraju, and R. Wattenhofer, "Greedy routing with bounded stretch," in *Proc. of the 28th Annual IEEE Conference on Computer Communications (INFOCOM)*, April 2009.

[8] J. Gao and L. Zhang, "Load balanced short path routing in wireless networks," *IEEE Transactions on Parallel and Distributed Systems, Special Issue on Localized Communications*, vol. 17, no. 4, pp. 377–388, April 2006.

[9] E. A. Jonckheere, M. Lou, F. Bonahon, and Y. Baryshnikov, "Euclidean versus hyperbolic congestion in idealized versus experimental networks," *CoRR*, vol. abs/0911.2538, 2009.

[10] B. Karp and H. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 2000, pp. 243–254.

[11] R. Kleinberg, "Geographic routing using hyperbolic space." in *Proceedings of the 26th Conference of the IEEE Communications Society (INFOCOM'07)*, 2007, pp. 1902–1909.

[12] T. Leighton and A. Moitra, "Some results on greedy embeddings in metric spaces," in *Proc. of the 49th IEEE Annual Symposium on Foundations of Computer Science*, October 2008, pp. 337–346.

[13] A. Mei and J. Stefa, "Routing in outer space: fair traffic load in multi-hop wireless networks," in *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2008, pp. 23–32.

[14] C. H. Papadimitriou and D. Ratajczak, "On a conjecture related to geometric routing," *Theor. Comput. Sci.*, vol. 344, no. 1, pp. 3–14, 2005.

[15] L. Popa, A. Rostamizadeh, R. Karp, C. Papadimitriou, and I. Stoica, "Balancing traffic load in wireless networks with curveball routing," in *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2007, pp. 170–179.

[16] P. Raghavan, "Probabilistic construction of deterministic algorithms: approximating packing integer programs," *J. Comp. and System Sciences*, pp. 130–143, 1988.

[17] P. Raghavan and C. D. Thompson, "Provably good routing in graphs: regular arrays," in *Proceedings of the 17th annual ACM Symposium on Theory of Computing*, 1985, pp. 79–87.

[18] R. Sarkar, X. Yin, J. Gao, F. Luo, and X. D. Gu, "Greedy routing with guaranteed delivery using ricci flows," in *Proc. of the 8th International Symposium on Information Processing in Sensor Networks (IPSN'09)*, April 2009, pp. 97–108.

[19] W. Thurston, "The finite riemann mapping theorem," in *Invited talk, An International Symposium at Purdue University on the occasion of the proof of the Bieberbach conjecture*, March 1985.

[20] W. Zeng, R. Sarkar, F. Luo, X. D. Gu, and J. Gao, "Resilient routing for sensor networks using hyperbolic embedding of universal covering space," in *Proc. of the 29th Annual IEEE Conference on Computer Communications (INFOCOM'10)*, March 2010.