Efficient Geometric Routing in Three Dimensional Ad Hoc Networks

Cong Liu and Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431

***Abstract*—Efficient geometric routing algorithms have been studied extensively in two-dimensional ad hoc networks, or simply 2D networks. These algorithms are efficient and they have been proven to be the worst-case optimal, localized routing algorithms. However, few prior works have focused on efficient geometric routing in 3D networks due to the lack of an efficient method to limit the search once the greedy routing algorithm encounters a local-minimum, like face routing in 2D networks. In this paper, we tackle the problem of efficient geometric routing in 3D networks. We propose routing on hulls, a 3D analogue to face routing, and present the first 3D partial unit Delaunay triangulation (PUDT) algorithm to divide the entire network space into a number of closed subspaces. The proposed *greedy-hull-greedy* (GHG) routing is efficient because it bounds the local-minimum recovery process from the whole network to the surface structure (hull) of only one of the subspaces.**

***Index Terms*—Delaunay triangulation, geometric routing, ad hoc networks, three-dimensional (3D) networks.**

I. Introduction

In this paper, we focus on efficient geometric routing algorithms for three-dimensional ad hoc networks, or simply 3D networks. Exploiting the geometry of the network to perform routing is a commonly-used approach for overcoming the challenges posed by resource-limited ad hoc networks. An important property of geometric routing algorithms is that they are based on local information, which can easily be updated to reflect the unavoidable topology changes in mobile networks.

Most efficient geometric routing protocols start with greedy forwarding, which is simple and close to optimal. However, greedy forwarding is not always successful as it fails when a message reaches a *local-minimum* node whose neighbors are all further away from the destination than the node itself. For 2D networks, face routing is the most prominent solution to recover from local-minimum [1], [2], [3], [4], and [5].

Because face routing relies on planar graphs, face-based geometric routing algorithms are only applicable in networks on strictly plane surfaces. Therefore, there is a need for 3D geometric routing algorithms that can be deployed in more adversary situations, such as on a hilly territory, in the sky, underground, and underwater. We found that few previous research works [6], [7], [8] have attempted to find a counterpart of face routing in 3D networks.

In this paper, we propose an efficient, delivery guaranteed, 3D geometric routing protocol, called *greedy-hull-greedy* (GHG) geometric routing. This algorithm contains a greedy forwarding algorithm and a hull routing algorithm. It is a 3D analogy of the greedy-face-greedy (GFG) routing protocol in 2D networks. Logically, the 3D network is divided into many subspaces. Once a message travels to a local-minimum in greedy forwarding, one of the adjacent subspaces of the local-minimum is identified, such that a local-minimum recovery search can succeed by only searching the nodes that are on the surface structure (hull) of this subspace. Our algorithm is efficient because it efficiently bounds the searched nodes to a portion of nodes in the network.

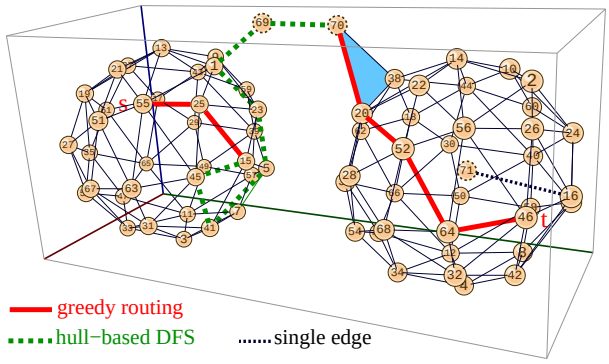


Fig. 1. An example of greedy-hull-greedy (GHG) routing.

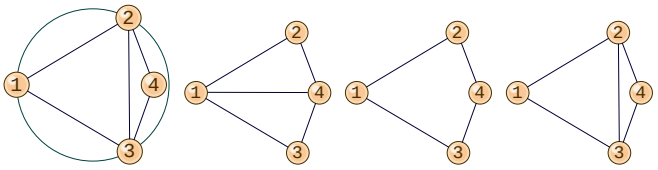
An example routing process of GHG is shown in Figure 1, where greedy routing sends the message from node 55 (source) to node 15. On local-minimum 15, hull routing sends the message from 15 to 70, where greedy forwarding can be continued (since 70 is closer to 46 than 15). Finally, greedy routing sends the message from 70 to 46 (destination).

Our contributions in this paper include: (1) We first propose to use *the partial unit Delaunay triangulation* (PUDT) to define network hulls (structures corresponding to subspaces) in 3D networks. (2) We present a localized PUDT algorithm, which reduces the PUDT construction cost to a little more than 1-hop worth. (3) We devise a hull routing algorithm, which efficiently recovers from local-minima in 3D networks. (4) We perform a simulation evaluation. Interested readers can refer to the long version of this paper available at [9].

II. Preliminaries

A. Geometric routing

This paper considers a geometric routing algorithm in ad hoc networks with all nodes distributed in 3D. Following



(a) UDG (b) DT (c) UDT (d) PUDT

Fig. 2. A comparison of UDG, DT, UDT, and PUDT.

the convention in geometric routing research, all wireless nodes have distinctive identities, each wireless node knows its location information (i.e. through a GPS receiver), and all wireless nodes have the same transmission range, which is normalized to one unit. Consequently, all wireless nodes (V) together define a *unit-disk graph* $UDG(V)$ in a 2D network, or a *unit-ball graph* $UBG(V)$ in a 3D network.

In [10], Liu and Wu proposed several force-based geometric routing protocols, which do not rely on planar graph and can be applicable to 3D networks. These algorithms outperform several state-of-the-art geometric routing protocols in 2D random networks; however, they do not have a small theoretical worst-case bound on route length.

B. Localized planar graphs in 2D networks

A planar graph construction algorithm logically removes some of the links (edges) from a graph such that there is no crossing edges in the graph. A planar graph construction is localized if every node u can remove the edges incident on u using only the information of the nodes within a constant number of hops (maximally 2-hops in our case), and each pair of adjacent nodes remove their common edge consistently. The *relative neighborhood graph* (RNG) consists of all edges uv such that $\|uv\| < 1$ and that there is no point w such that $\|uw\| < \|uv\|$ and $\|wv\| < \|uv\|$. Let $disk(u, v)$ be the closed disk with diameter uv. The *Gabriel graph* (GG) consists of all edges uv such that $\|uv\| < 1$ and the interior of $disk(u, v)$ does not contain any other node. There is no crossing edges in RNG and GG because at least one edge in any pair of crossing edges must be removed according to their definitions.

In a d-dimensional Euclidean space, a Delaunay triangulation is a triangulation $DT(V)$ such that no point in V is inside the circum-hypersphere of any d-simplex in $DT(V)$. Here, a d-simplex is the d-dimensional analogue of a triangle. We are interested in 3-dimensional spaces where the 3-simplex is a tetrahedron. In Euclidean space, the Delaunay triangulation of V corresponds to the dual graph of the Voronoi tessellation for V.

A *unit Delaunay triangulation* (UDT) [11] is a subgraph of DT in terms of edges and UDT = DT $\cap$ UDG. That is, UDT differs from DT in that UDT only contains the edges which are shorter than one. *Partial unit Delaunay triangulation* (PUDT) differs from UDT in that PUDT might contain extra edges and PUDT always results in a connected graph. In a PUDT graph in a 2D network, an edge is removed when it crosses another edge and it does not belong to any DT. That is, PUDT removes edges more reluctantly than RNG and GG. Details of 3D PUDT construction will be presented in Section IV. A comparison example of DT, UDT and PUDT in a 2D network is shown in Figure 2. In UDT (Figure 2(c)), $edge(p_1, p_4)$ is removed because it is longer than 1, and $edge(p_2, p_3)$ is removed for not belonging to any DT. In PUDT (Figure 2(d)), $edge(p_2, p_3)$ is retained because it does not intersect any other edge in UDG. To summarize, UDT $\subseteq$ DT, but PUDT $\subseteq$ DT is not true.

III. The Proposed Approach

The proposed geometric routing protocol starts with greedy forwarding. Whenever the message is forwarded to a local-minimum, a recovery process starts, searching in one of the subspaces adjacent to the local-minimum that contains the line segment between the local-minimum and destination. Once the recovery process sends the message to a node that is closer to the destination than the local-minimum, it switches back to greedy forwarding.

This paper tackles several challenges in 3D geometric routing. The first challenge is dividing the network space into subspaces in order to limit the local-minimum recovery search in one of these subspaces, which improves recovery efficiency. In 2D networks, faces are areas enclosed by edges. Local-minimum recovery is performed on a particular face by searching along consecutive edges bordering the face. In 3D networks, we first propose to use triangles, each of which is defined by three connected nodes, to divide the 3D network space into subspaces. The entire network space is divided by triangles into one outer subspace and zero to any number of inner subspaces. Any inner subspace is enclosed by triangles. Traveling from a point inside one subspace to a point inside another subspace must go through at least one triangle. In Figure 1, the network consists of two spherical inner subspaces and one outer subspace which fills the rest of the entire space.

The second challenge is removing intersecting triangles. In 2D planarization, crossing edges need to be removed such that the whole network area can be divided into faces. Similarly, non-overlapping subspaces cannot be divided when intersecting triangles exist. We propose a low-cost, localized PUDT algorithm in Section IV to remove intersecting triangles which use just over 1-hop worth of information.

The third challenge is identifying nodes in a particular subspace. This is required by our local-minimum recovery search, which is limited to the nodes belonging to a particular subspace. To do so, we define a *hull* of a subspace as a structure containing triangles (that enclose the subspace) and single edges (that do not belong to any triangle and are inside the subspace). Nodes on the vertices of the triangles or single edges of a hull belong to the subspace of the hull. Defining hulls essentially groups the triangles and single edges (and consequently nodes) into different subspaces. We propose a local hull construction in Section V, in which each node locally groups its triangles and single edges into different local hulls. Specifically, we distinguish each triangle by its

two sides; then each sided triangle or single edge belongs to exactly one local hull.

In Figure 1, the hull of the right-side inner subspace contains all the triangles on the right-side ball and the single $edge(p_{16}, p_{71})$ inside the ball. Some triangles have two of their sides belonging to different hulls, and others have both sides belonging to the same hull. For example, both sides of the blue triangle, $\Delta(p_{20}, p_{38}, p_{70})$, belong to the outer hull (outer subspace). Each single edge belongs to only one hull which contains it.

The last, but not least, challenge deals with searching (Section VI) on the target hull (hull routing). For a particular destination, a target hull is a hull that is adjacent to the local-minimum and contains the line segment between the local-minimum and destination. Recovery from the local-minimum is guaranteed when searching on the target hull. For enhanced performance, we propose a hull-based connected dominating set (CDS) in Section VI, which further limits the node in the local-minimum recovery search.

IV. Partial Unit Delaunay Triangulation (PUDT)

A. The basic approach and its correctness

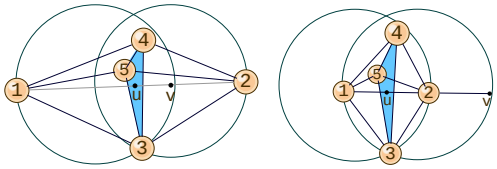
We denote nodes as $p_1, p_2, \ldots$, an edge between nodes p_1 and p_2 as $edge(p_1, p_2)$, a triangle determined by three points not in a line as $\Delta(p_1, p_2, p_3)$, a tetrahedron as $T(p_1, p_2, p_3, p_4)$, and a ball determined by four points not on the same plane as $ball(p_1, p_2, p_3, p_4)$. Note that $ball(p_1, p_2, p_3, p_4)$ is the circumsphere of $T(p_1, p_2, p_3, p_4)$.

When the edges can be arbitrarily long, the result of applying Delaunay triangulation to a set of vertices is a space uniquely divided into a number of non-overlapping tetrahedra (the 3D simplexes) and a single outer subspace. Each tetrahedron is a subspace enclosed by four triangles. The rule to determine the Delaunay tetrahedra is that only the tetrahedra without a fifth vertex inside its circumsphere is valid. Two tetrahedra are overlapping if there is a common point inside both tetrahedra.

Our PUDT algorithm in 3D networks is analogous to planar graph construction in 2D networks: planar graph construction removes intersecting edges, while our PUDT construction removes intersecting triangles and edges. It can be proven that if there is no intersecting edge and triangle, then there is no overlapping tetrahedra. This is because when two tetrahedra overlap, one of the four triangles on the first tetrahedron must intersect a triangle on the second tetrahedron; moreover, if two triangles intersect, an edge of one of the triangles must intersect the other triangle.

When the network is very dense, the retained triangles partition the network space into a number of tetrahedra. Otherwise, there are some irregular polyhedra, each of which consists of a number of tetrahedra combined because the triangle between them does not exist for owning edges that are longer than 1.

Our basic PUDT algorithm logically removes the edges and triangles. The triangles and edges to be removed (invalid triangles and edges) are defined in Definition 1. Definition 1 guarantees that an edge and a triangle cannot exist at the same time if they intersect. Note that we define edges and triangles as two kinds of objects. Removing any of the three edges of a triangle results in removing the triangle. However, when a triangle is removed, all of its three edges can be retained in the graph.



(a) Case 1 (b) Case 2

Fig. 3. Illustrations for Definition 1.

Definition 1 (Invalid edge & triangle): If $edge(p_1, p_2)$ intersects $\Delta(p_3, p_4, p_5)$ and p_2 is outside $ball(p_1, p_3, p_4, p_5)$, then $edge(p_1, p_2)$ is an invalid edge (invalidated by $\Delta(p_3, p_4, p_5)$). Otherwise, if p_2 is inside $ball(p_1, p_3, p_4, p_5)$, then $\Delta(p_3, p_4, p_5)$ is an invalid triangle (invalidated by $edge(p_1, p_2)$). $\Delta(p_3, p_4, p_5)$ is also invalid if any of its three edges are invalid or if there exists a p_6 such that the radius of $ball(p_1, p_3, p_4, p_5)$ is greater than 1.

The last constraint on the validity of triangles in Definition 1 is a patch for Delaunay triangulation regarding not guaranteeing delivery [3]. It can be proven that invalid edges and triangles determined distributively are consistent among the nodes, that the position information of a 2-hop neighbor is sufficient for the correctness of Definition 1 in removing intersecting triangles and edges, and that network connectivity is conserved when invalid edges are removed.

B. A low-cost PUDT algorithm

We have showed a basic PUDT algorithm in which the nodes propagate 2-hops of position information and then remove all invalid edges and triangles. In this subsection, we will calculate the PUDT with just over 1-hop worth of position information. In this low-cost PUDT algorithm, each node sends its own position and might also send a small amount of *advertised information* to its neighbors. Advertised information is the minimal information of 1-hop neighbors' positions that need to be sent in order to achieve consistency. Please refer to [9] for the details of this algorithm.

V. Localized Hulls

After the PUDT construction, each node knows all of its adjacent valid edges and triangles that do not intersect each other. Note that greedy forwarding uses all nodes in the network, whereas hull routing only uses nodes on the valid edges and triangles. Therefore, starting from this section, only valid edges and triangles are considered. For simplicity, when we talk about triangles, we only refer to valid triangles, and when we talk about edges, we only refer to valid and single edges, and we refer to both valid triangle and valid single edge as object.

The reason to divide a network into different subspaces is to reduce the overhead in the local-minimum recovery process from searching in the whole network to searching only the nodes in a particular subspace. The recovery process guarantees success when searching the target subspace. The notion of hull is used here to identify nodes belonging to each subspace. The nodes belonging to a subspace include the nodes on the vertices of the triangle enclosing the subspace and the nodes on the vertices of the single edges that are inside the subspace. The recovery process, i.e. hull routing, will be presented in Section VI.

A. Identifying localized hulls

Local hull construction is a process in which each node classifies the objects belonging to its local hulls. Each object belongs to only one local hull. Recall that a hull for a particular subspace is a structure which contains the triangles enclosing the subspace and the single edges that are inside the subspace. Due to space limitation, we cannot provide the details of the algorithm to identify local hulls. Again, refer to [9] for the details of this algorithm.

B. Determining the target hull

When a message reaches a local-minimum, one of the adjacent hulls of the local-minimum is selected such that the message can recover from the local-minimum by searching the nodes on this hull, which is called *target hull*. We define target hull as the hull whose subspace contains the imaginary m-t segment connecting the local-minimum and destination.

Since each *object* belongs to only one hull, in order to determine the target hull, we only need to find a representative object (a triangle or a single edge), which belongs to the target hull. Since the m-t segment is contained in the target hull, we only need to find the *closest object* [9] of the m-t segment. This object will be in the same subspace as the m-t segment, and the hull that contains this object is the target hull. The proof that hull routing on the target hull can always make progress is shown in [9].

VI. GHG Routing

Greedy-hull-greedy (GHG) routing is analogous to greedy-face-greedy (GFG) routing. All geometric routing algorithms contain a greedy routing algorithm and a recovery algorithm, since greedy routing (which forwards the message ever closer to its destination) is the most simple and efficient. Note that greedy routing has nothing to do with hull routing and it can use any node in the network. An execution of GHG is a repetitive alternation between greedy forwarding and hull routing. GHG can be easily extended with a bounded circle as in [5] to achieve the worst case bound $O(d^3)$, where d is the distance between the source and destination.

A. Efficient searching on the target hull

From this subsection on, we are only concern with hull routing. We use a depth-first-search to travel the target hull which will eventually send the message to the node where greedy forwarding can be recovered. In 2D, searching the border of a face for a recovery node is a trivial one-dimensional search. Random walk is proposed in [6] to search the hull of a virtual cube structure which is also proposed in [6]. In this paper, we use an efficient hull-based, depth-first search, in which each message is forwarded at most twice the number of nodes on a target hull (leaf nodes forward at most once and non-leaf nodes forward at most D times, where D is the node degree). Therefore, we conserve the worst-case bound.

In [8], a depth-first search (DFS) has been proposed for geometric routing where depth is defined as the reciprocal of the distance between the node and destination. In this algorithm, for each message m, if a node u receives it for the first time, the node creates a record for m. If v is the first node that forwards m to u, then v is the ancestor of u with respect to m's DFS search. When u receives m from some node w and u received m before, if u has not forwarded m to w before, u returns m to w. Otherwise, u sorts its neighbors to which it has not forwarded m, in decreasing order of their depth with respect to the destination of m, and forwards m to the first of them. If such a neighbor does not exist, u returns m to its ancestor v.

In this algorithm, message m does not need to maintain any information about its routing state. All information is stored in the nodes. Such information can be deleted after m has expired. Since both the messages' time-to-live (TTL) and the bandwidth in ad hoc networks are small, the amount of routing state information stored in each node is small. We improve this algorithm in [12], in which we make use of the broadcast nature of wireless communication. Check [9] for its details.

B. Extension: CDS on a hull

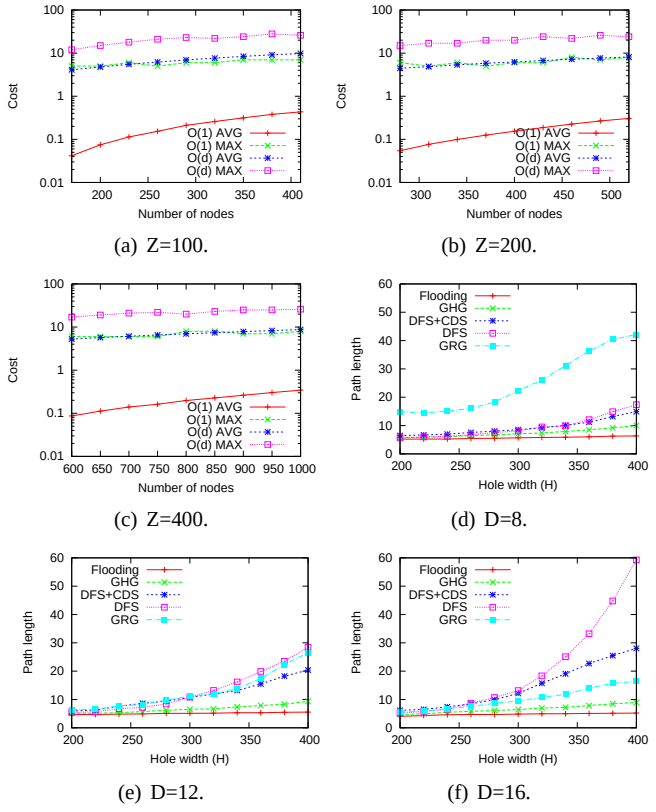
In [13], the Gabriel graph is constructed on the connected dominating set (CDS) of the network to reduce the number of nodes on each face. Similarly, we use CDS to reduce the number of nodes on each hull to make searching more efficient.

VII. Simulation

A. Evaluation of the PUDT algorithm

The simulation was run in our EASIM simulator [14]. PUDT is performed in random 3D networks of size $1,000 \times 1,000 \times Z$, where Z varies among 100, 200, and 400. For each Z, networks containing a different amount of nodes are generated. For each Z and each network density, 100 networks are generated to repeat the simulation by randomly selecting an (x, y, z) coordinate for each node within the specific space.

We compare the cost of the basic PUDT (denoted by $O(d)$ in the simulation results) and the low-cost PUDT (denoted by $O(1)$) in terms of the size of position information exchanged among the nodes. The size is measured by the volume of position information, each of which contains three integers describing the x, y, and z coordinates of a node. The simulation results are plotted in log scale. The measurement does not include information of the node's own position. Simulation results in Figures 4(a)-4(c) show that the low-cost PUDT is



(a) Z=100. (b) Z=200.

(c) Z=400. (d) D=8.

(e) D=12. (f) D=16.

Fig. 4. Comparison of PUDT costs (a-c) and routing perforamnce (d-f).

much cheaper than the simple approach: the average amount of position information ranges from below 0.1 to 0.4 and the maximum number is below 10. That is, the total amortized cost of our low-cost PUDT is just over 1-hop of information. Also, the cost of the low-cost PUDT is only around 3% of the 2-hop, $O(d)$, information in the basic PUDT.

B. Routing performance

We compare the routing performance of Flooding (which finds the optimal paths), DFS [8] and DFS+CDS (DFS runs on the connected dominating set of the network), greedy-random-greedy (GRG) [6], which performs its random walk local-minimum recovery search on the hull we constructed, and GHG. Our simulation metric is in terms of hop-count.

We use networks with randomly placed nodes and artificial holes to emulate obstacles in practical situations. We generate random networks of different node degree D. The average node degree in different networks ranges from 8, 10, or 12 neighbors per node. Small holes, other than the artificial hole, might exist in regions where node density is low. Disconnected networks are discarded. The size of all networks is $500 \times 500 \times 500$ and the transmission range of the nodes is 100. In different networks, a rectangular hole whose size is $H \times H \times 150$ is created at the center of the network, where H is a variable between 200 and 400 in different networks.

First, we compare GHG with Flooding, DFS, and DFS+CDS. Figures 4(d)-4(f) show that, as expected, the routing performance of all routing algorithms degrades as the size of the hole increases. The performance of GHG is, on average, only 20% longer than the optimal path length of Flooding and is at most 50% longer in the worst-case. Second, we compare GHG with GRG. From Figures 4(d)-4(f), the performance of all of Flooding, GRG, and GHG increases as network density increases.

VIII. Conclusion

In this paper, we have proposed some solutions for efficient geometric routing in 3D networks. We present the first 3D localized partial unit Delaunay triangulation (PUDT) algorithm, hull recognition algorithm, and *greedy-hull-greedy* (GHG) algorithm, the first 3D analogue to face routing. Simulation results show that PUDT is low in cost, and GHG is more efficient than depth-first-search (DFS) and greedy-random-greedy (GRG). We believe many future works can be developed on our model: problems in geometric routing in 2D networks can be redefined or extended in 3D networks, which include multicast, geocast, virtual coordinates, handling uncertain position information, and energy efficient routing.

IX. Acknowledgement

This work was supported in part by NSF grants CNS 0422762, CNS 0434533, CNS 0531410, CCF 0545488, and CNS 0626240.

References

[1] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with Guaranteed Delivery in Ad Hoc Wireless Networks. In *Proc. of ACM DIAL-M*, 1999.

[2] B. Karp and H.T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proc. of ACM MobiCom*, 2000.

[3] H. Frey and I. Stojmenovic. On Delivery Guarantees of Face and Combined Greedy-Face Routing in Ad Hoc and Sensor Networks. In *Proc. of ACM MobiCom*, 2006.

[4] F. Kuhn, R. Wattenhofer, and A. Zollinger. Worst-Case Optimal and Average-Case Efficient Geometric Ad-Hoc Routing. In *Proc. of ACM MobiHoc*, 2003.

[5] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric Ad-Hoc Routing: Of Theory and Practice. In *Proc. of ACM PODC*, 2003.

[6] R. Flury and R. Wattenhofer. Randomized 3D Geographic Routing. In *Proc. of IEEE INFOCOM*, 2008.

[7] S. Durocher, D. Kirkpatrick, and L. Narayanan. On Routing with Guaranteed Delivery in Three-Dimensional Ad Hoc Wireless Networks. In *Proc. of ICDCN*, 2008.

[8] I. Stojmenovic, M. Russell, and B. Vukojevic. Depth First Search and Location Based Localized Routing and QoS Routing in Wireless Networks. *Computers and Informatics*, 21(2):149–165, 2002.

[9] C. Liu and J. Wu. Efficient Geometric Routing in Three Dimensional Ad Hoc Networks (Long Version). http://www.geocities.com/gzcliu_pub/infocom09mini.pdf.

[10] C. Liu and J. Wu. Virtual-Force-Based Geometric Routing Protocol in MANETs. In *IEEE TPDS*, 2008.

[11] X.-Y. Li, G. Calinescu, P.-J. Wan, and Y. Wang. Localized Delaunay Triangulation With Application in Wireless Ad Hoc Networks. In *Proc. of IEEE INFOCOM*, 2003.

[12] C. Liu and J. Wu. Poster Abstract: Force-based Geometric Routing. In *ACM MobiHoc Poster Session*, 2008.

[13] S. Datta, I. Stojmenovic, and J. Wu. Internal Node and Shortcut Based Routing with Guaranteed Delivery in Wireless Networks. *Cluster Computing, Special Issue on Mobile Ad Hoc Networks*, April 2002.

[14] C. Liu. EASIM 3D ad hoc network simulator. http://www.geocities.com/gzcliu_proj/.