Concurrent Face Traversal for Efficient Geometric Routing[☆]

Thomas Clouser, Mark Miyashita and Mikhail Nesterenko[1]

Dept. of Computer Science, Kent State University, Kent, OH, 44242, USA

Abstract

We present a concurrent face routing *CFR* algorithm. We formally prove that the worst case latency of our algorithm is asymptotically optimal. Our simulation results demonstrate that, on average, the path stretch, i.e. the speed of message delivery, achieved by *CFR* is significantly better than by other known geometric routing algorithms In fact, it approaches the shortest possible path. *CFR* maintains its advantage over the other algorithms in pure form as well as in combination with greedy routing. *CFR* displays this performance superiority both on planar and non-planar graphs.

Keywords: Geometric routing; ad hoc wireless routing.

1. Introduction

Geometric routing is an elegant approach to data dissemination in resource-constrained and large-scale ad hoc networks. Geometric routing is attractive because it does not require nodes to maintain, or messages to carry, extensive state or routing information. Geometric routing algorithm rely instead on known global node coordinates. This lack of routing infrastructure makes such algorithms a popular initialization or fallback option for other routing schemes. Therefore, geometric routing optimization is of interest to the broad community of ad hoc wireless network designers.

In geometric routing, each node knows its own and its neighbors' coordinates. Using low-cost GPS receivers or location estimation algorithms [6, 13], wireless nodes can learn their relative location with respect to the other nodes and then use this information to make routing decisions. The message source node knows the coordinates of the destination node. These coordinates may be obtained from a location service [1, 26]. The information that the message can carry does not depend on the network size. Each forwarding node does not maintain any extensive routing data or keep any information about forwarded messages between message transmissions.

[☆]The preliminary version of this article was presented at OPODIS 2008.
[1]This author was supported in part by NSF Award 0347485.

Greedy routing [9] is an elementary approach to geometric routing where the node selects the neighbor that is the closest to the destination and forwards the message there. The process repeats until the destination is reached. Greedy routing fails if the node is a *local minimum*: it does not have neighbors that are closer to the destination than itself. Alternatively, in *compass routing* [20], a node selects the neighbor whose direction has the smallest angle to the direction of the destination. However, this kind of compass routing is prone to livelocks.

One way to circumvent these delivery problems in geometric routing is to flood a region of the network with messages [9, 17, 19, 28, 30]. This is useful in *geocasting* [18] where each node in a certain region of the network needs to receive a message. However, for point-to-point communication flooding may not be efficient. Another way is to pre-compute some graph information in advance: such as local minima [3] or local Delaunay triangulation [12]. This, however, may require extensive computation and communication before the routing information is available for message transmission. Moreover, the nodes have to store the pre-computed information locally which may diminish the saving from the statelessness of geometric routing.

The *face routing* variants of geometric routing are designed to guarantee message delivery without incurring the message overhead associated with flooding. A source-destination line intersects a finite number of faces of a planar graph. Hence, if a message carries the coordinates of the source and destination nodes, the routing nodes may determine this intersection and switch the message from one face traversal to the next. This way the message reaches the destination by sequentially traversing these faces. In the algorithms published thus far, the faces are traversed sequentially. *GFG*/*GPSR* [5, 14] combines greedy and face routing. Greedy routing is used for speed, and face routing helps to recover from local minima. Datta et al [7] propose a number of optimizations to face traversal. Kuhn et al [21, 22, 24] propose a worst case asymptotically optimal geometric routing algorithm *GOAFR+*. They compare the performance of multiple geometric routing algorithms and demonstrate that in the average case *GOAFR+* also performs the best. Kim et al [16] discuss challenges of geometric routing. Frey and Stojmenovic [10] address some of these challenges and discuss different approaches to geometric routing. Stojmenovic [11] provides a comprehensive taxonomy of geometric routing algorithms.

One of the shortcomings of traditional geometric routing is the need to planarize the graph. This can be done effectively only for unit-disk graphs. However, a unit-disk graph is a poor approximation for most radio networks where radio propagation patterns are not as regular as assumed in unit-disk graphs. Some researchers [4, 23] explore a more realistic model of *quasi unit disk graphs*. Nesterenko and Vora [29] propose a technique of traversing voids in non-planar graphs similar to face traversal. This traversal may be combined with greedy routing similar to *GFG*. Barrière et al [4], Kim et al [15], Leong et al [25], and Kuhn et al [23] propose alternative ways of performing geometric routing over non-planar graphs.

Kuhn et al [21, 24] conduct extensive evaluation of geometric routing algo-

rithms' performance. They compare the ratio of the path selected by a routing algorithm to the optimal path depending on the graph density. Their findings indicate that at low and high density the performance of most algorithms, especially if combined with greedy routing, approaches optimal. In sparse graphs, due to the limited number of available routes, a geometric routing algorithm is bound to select a route that is close to optimal. In dense graphs, an algorithm nearly always runs in greedy mode which tends to select a nearly optimal route as well. Kuhn et al identified a *critical density* range between 3 and 7 nodes per unit-disk where the paths selected by geometric routing algorithms may substantially differ from the shortest paths and where performance optimization has the greatest impact.

Somewhat related studies [2, 8] demonstrate how randomization or limited directional flooding can be used to effectively route in dense networks.

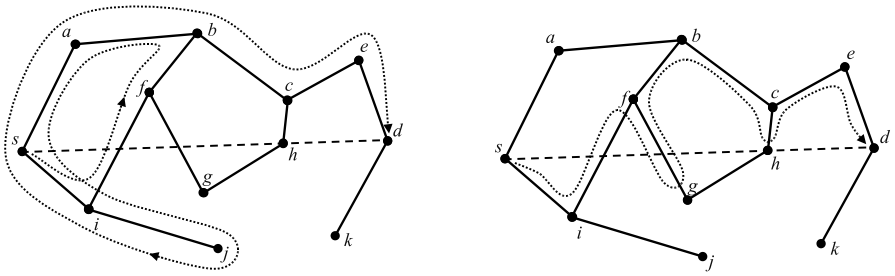
Despite their individual differences, the foundation of most geometric routing algorithms is face traversal. In such traversal, a message is routed around a face. However, the resultant route may vary greatly depending on the choice of traversal direction and the point at which the message switches between adjacent faces. The imbalance is usually exacerbated if the message has to traverse the external face of the graph. However, if the message traverses the faces sequentially, exploring the faces to find a shorter route may result in lengthening the route itself. Hence, traditional geometric routing algorithms are inherently limited in the amount of route optimization they can achieve.

In this paper, we present an algorithm that accelerates the message propagation by sending messages to concurrently traverse faces adjacent to the source-destination line. We call this algorithm concurrent face routing (*CFR*). *CFR* preserves all properties of classic geometric routing algorithms except for one: when one of the messages encounters a face that is closer to the destination, the message spawns two messages to traverse the new face and continues traversing the old face. *CFR* ensures that all faces are explored and none of the adjacent edges is traversed more than once. The node memory and message-size requirements for *CFR* are the same as for the other geometric routing algorithms. We show that the latency of *CFR* is asymptotically optimal in the worst case. That is, there is no geometric routing algorithm that can deliver a message faster than *CFR*. Moreover, our simulation demonstrates that, on average, *CFR* significantly outperforms other geometric routing algorithms in the critical density region. This average case advantage is preserved if *CFR* is combined with greedy routing or if it runs on non-planar graphs.

Paper organization. The rest of the paper is organized as follows. We introduce our notation in Section 2. We then describe *CFR*, formally prove its correctness and determine its worst case message complexity in Section 3. In Section 4, we discuss how the algorithm can be adapted for greedy routing and for use in non-planar graphs. We evaluate the performance of our algorithm and its modifications in Section 5 and conclude the paper in Section 6.

2. Preliminaries

Graphs. We model the network as a connected geometric graph $G = (V, E)$. The set of *nodes* (*vertices*) V are embedded in a Euclidean plane and are connected by *edges* E. The graph is *planar* if its edges intersect only at vertices. A *void* is a region on the plane such that any two points in this region can be connected by a curve that does not intersect any of the edges in the graph. Every finite graph has one infinite *external* void. The other voids are internal. A void of a planar graph is a *face*.



(a) COMPASS (b) FACE

Figure 1: Example operation of existing planar face traversal algorithms.

Face traversal. Each message is a *token*, as its payload is irrelevant to its routing. *Right-hand-rule* face traversal proceeds as follows. If a token arrives to node a from its neighbor b, a examines its neighborhood to find the node c whose edge (a, c) is the next edge after (a, b) in a clockwise manner. Node a forwards the token to c. This mechanism results in the token traversing an internal face in the counter-clockwise direction, or traversing the external face in the clockwise direction. *Left-hand-rule* traversal is similar, except the next-hop neighbor is searched in the opposite direction.

A source node s has a message to transmit to a destination node d. Node s is aware of the Euclidean coordinates of d. Node s attaches its own coordinates as well as those of d to the messages. Thus, every node receiving the message learns about the sd-line that connects the source and the destination. Depending on whether the token is routed using right- or left-hand-rule, it is denoted as R or L. Each node n knows the coordinates of its *neighbors*: the nodes adjacent to n in G. A *juncture* is a node whose adjacent edge intersects the sd-line. A node itself lying on the sd-line is also a juncture. Thus, the source and destination nodes are junctures themselves. Two faces are *adjacent* if their borders share a juncture. Note that a single node may be a juncture to multiple faces if more than one of its adjacent edges intersect the sd-line.

To simplify the algorithm presentation, we use anthropomorphic terms when referring to the nodes of the network such as "know", "learn" or "forget".

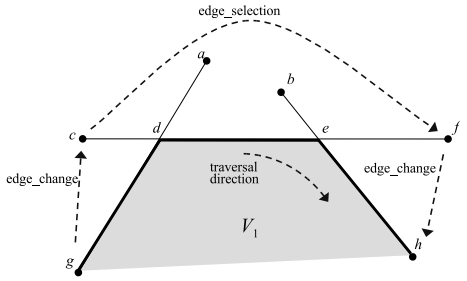


Figure 2: Traversing non-planar voids.

Performance metrics. The *message cost* of an algorithm is the largest number of messages sent in a single computation. This number is calculated in terms of the network graph parameters. A *latency* is the shortest path the message in the algorithm takes to reach the destination. Essentially, message cost captures the expense of communication while the latency captures its speed. Note that for sequential traversal algorithms, such as traditional geometric routing algorithms, where there is always a single message in transit, the two metrics are the same. Note also that the latency of a certain algorithm, i.e. the shortest path that the message takes, is not necessarily the *optimum* or the shortest possible path between the source and the destination. A *path stretch* is the ratio between the latency of the algorithm and the shortest path in the graph.

Existing face traversal algorithms. One of the first known face routing algorithms that guarantees delivery is Compass Routing II [20]. In this paper we refer to it as *COMPASS*. In *COMPASS*, the token finds the juncture that is closest to the destination. For this, the token traverses the entire face and returns to the initial point of entry. The token then goes to the discovered closest juncture. There, the token changes faces and the process repeats. Refer to Figure 1(a) for an example route selected by *COMPASS*. The message complexity of *COMPASS* is $3|E|$ which is in $O(|E|)$.

In *FACE* [5, 7], the token changes faces as soon as it finds the first juncture (refer to Figure 1(b)). In degenerate cases, *FACE* allows the token to traverse the same edge multiple times. Hence, its worst case message complexity is in $O(|V|^2)$. It is worse than that of *COMPASS*. However, *FACE* tends to perform better in practice. Both algorithms may select a route that is far from optimum. The selected route may be particularly long if the token has to traverse the external phase as in the above examples. *OAFR* [24] mitigates long route selection by defining an ellipse around the source-destination pair that the message should not cross. If the message traverses a face and reaches the boundary of the ellipse, the message changes the traversal direction. *OAFR* has the best worst case efficiency for a sequential face traversal algorithm to date. Its path stretch is in $O(\rho^2)$, where ρ is the length of the optimum path.

Algorithms *COMPASS*, *FACE* and *OAFR* operate only on planar graphs.

Obtaining such a graph from a graph induced by a general radio network may be problematic. There are several attempts to allow geometric routing on non-planar graphs [7, 15, 25, 29]. In particular, Nesterenko and Vora [29] propose to traverse non-planar voids similar to faces. Note that in general the edges that are adjacent to voids do not intersect at the incident vertices. However, the idea is to have the message follow the segments of the edges that are adjacent to the void. Refer to Figure 2 for illustration.

Each pair of nodes u and v adjacent to an edge (u, v) keeps the information about which edges intersect (u, v) and how to route to the nodes adjacent to these edges. Suppose node g receives the token that traverses void V_1. Node g forwards the token to c in an *edge_change* message. Recall that in a non-planar graph, edges do not have to intersect at nodes. Edges (g, a) and (c, f) intersect at point d. The objective of nodes c and f is to select an edge that intersects (c, f) as close to d as possible. At first c selects an edge and forwards the token with its selection to f in an *edge_change* message. Node f consults its own data, selects edge (b, h) and forwards the token to one of the nodes adjacent to this edge. Thus, the message can completely traverse the void.

Note that once the void traversal is designed, the various techniques of void-change and exploration can generate the non-planar equivalents of *COMPASS*, *FACE* and *OAFR*.

Combining greedy routing and face traversal. *GFG/GPSR* [5, 14] improves the quality of route selection by combining face routing with greedy routing. *GOAFR+* [21] does the same for *OAFR*. *GOAFR+* achieves remarkable characteristics. It retains the asymptotic worst case optimality of *OAFR* and achieves the best average case path stretch known to date.

In the combination of greedy routing and face traversal the token has two traversal modes: greedy and face. The token starts in the greedy mode but switches to face mode if it encounters a local minimum (a node with no neighbors closer to the destination). The token continues in the face mode until it finds a node that is closer to the destination than this local minimum. Then the token switches to greedy mode again until another local minimum is discovered.

Execution model. To present our algorithm, we place several assumptions on the execution model. We assume that each node can send only one message at a time. The node does not have control as to when the sent message is actually transmitted. After the node appends the message to the send queue SQ, the message may be sent at arbitrary time. Each channel has zero capacity; that is, the sent message leaves SQ of the sender and instantaneously appears at the receiver. Message transmission is reliable (i.e. there is no message loss). The node may examine and modify SQ. We assume that SQ manipulation, including its modification and message transmission, is done *atomically*. We assume that the execution of the algorithm is a sequence of atomic actions. The system is *asynchronous* in the sense that the difference between algorithm execution speed at each node is arbitrary.

```
node s
/* let F be a face bordering s
     and intersecting the sd-line */
add L(s, d, F) to SQ
add R(s, d, F) to SQ

node n
if receive L(s, d, F) then
     if R(s, d, F) ∈ SQ then
          /* found matching token */
          delete R(s, d, F) from SQ
     else
          if n = d then
               deliver L(s, d, F)
          if n is a juncture and
            F locally intersects the sd-line then
               foreach F' ≠ F that locally
                         intersects the sd-line do
                    add L(s, d, F') to SQ
                    add R(s, d, F') to SQ
          add L(s, d, F) to SQ
if receive R(s, d, F) then
     /* handle similar to L(s, d, F) */
```

Figure 3: Pseudocode of *CFR* at each node.

3. *CFR* Description, Correctness Proof Perfomance and Bound Computation

Description. The pseudocode of *CFR* is shown in Figure 3. The algorithm operates as follows. Initially, the source node s sends left and right tokens into the face that intersects the *sd*-line by adding them into its send queue SQ. Then the operation of the algorithm is driven by token receipts. If a node receives a left message traversing some face F, the node checks if there is a matching token in SQ. If the matching token is found, the node removes it from SQ effectively destroying both tokens.

Otherwise, the node checks whether it is the destination node. If it is, the node delivers the message and continues to process it further. If the node determines that it is a juncture, i.e, there is at least another face F' that intersects the *sd*-line, the node injects the left and right tokens into each such face. The node also forwards the message further along the old face F. The right token receipt is processed similarly.

Refer to the pictures in Figure 4 for the illustration of the algorithm's operation. In the figure, we show three snapshots of a single computation. Thin solid lines denote particular tokens. The tokens are numbered. To reduce clutter in the pictures, we only reproduce token numbers. Thus, token t_5 is shown as 5. Some tokens are destroyed before they leave their originating node. See for

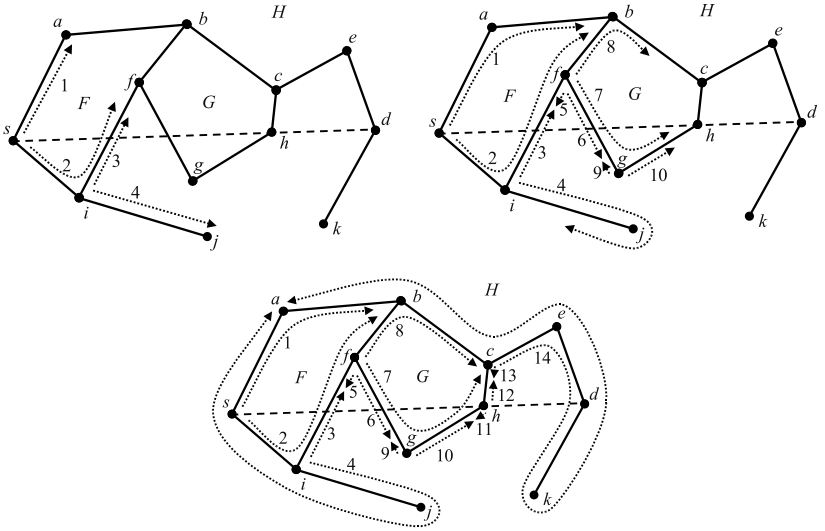


Figure 4: Example of *CFR* operation on a planar graph.

example t_5 or t_9. We denote such tokens by short arrows. In the picture, the face names are for illustration only, the global face names are not known to the incident nodes. The token carries its traversal direction: L or R. Note that when a node receives a token, it can locally determine which adjacent face the token traverses on the basis of its sender and its traversal direction. For example, when node a receives L token t_1 from node s, a knows that t_1 traverses the adjacent face F. Two tokens at a node *match* if they traverse the same face in the opposite directions and at least one of them did not originate in this node. For example, t_6 and t_9 at g as well as t_3 and t_5 at f match. However, t_{11} and t_{12} at h do not match because h originated both of these tokens.

Note that a juncture node can locally determine if an adjacent face locally intersects the sd-line. For example, s knows that F intersects the sd-line while H does not. If a token arrives at a juncture and the token traverses a face that locally intersects the sd-line the juncture node injects a pair of tokens into each other neighboring face that intersects the sd-line. For example, when f receives t_2 traversing F that locally intersects the sd-line, f sends t_5 and t_6 to traverse H, and t_7 and t_8 to traverse G. Similarly when h receives t_7, it sends t_{11} and t_{12} to traverse H. Observe that a juncture node injects the new tokens only if the token it receives is traversing the face that locally intersects the sd-line. For example, when juncture node c receives t_{14} from e, it just forwards the token to b without injecting tokens into G.

If the destination node receives the token, even though the node delivers it, it processes the token further as an ordinary node. That is, node d forwards the token and injects tokens in adjacent faces if necessary.

Example operation. Let us now consider example operation of CFR in the computation in Figure 4 in detail. Node s initiates the transmission by sending tokens t_1 and t_2 to traverse face F. When t_2 reaches juncture node i, i injects t_3 and t_4 into H and forwards t_2 to f. Node f is also a juncture. Thus, besides forwarding t_2 to b, it injects t_5 and t_6 into H as well as t_7 and t_8 into G. Token t_2 meets a matching token t_1 at b and both tokens are destroyed. This completes the traversal of F. Tokens t_7 and t_8 traverse G and meet in c, where they destroy each other. In the process t_7 reaches all the remaining juncture nodes: g, h and c where the tokens are injected in the adjacent faces. Specifically, t_7 causes the injection of t_9 and t_{10} at g, t_{11} and t_{12} at h and t_{13} and t_{14} at c. All tokens are injected into the external face H. The tokens traversing H find matching tokens and are quickly eliminated at f, g, h and c. Tokens t_4 and t_{14} complete the traversal of H. They arrive at a which destroys them. On its way t_{14} visits d, which delivers it.

Correctness proof.

Lemma 1. *For each node n bordering a face F that intersects the sd-line one of the following happens exactly once: either (1) n receives token $T(s, d, F)$ where T is either R or L and forwards it or (2) n has a token, receives a matching token and destroys them both.*

Proof: According to the algorithm, a token visits a node and proceeds to the next node along the face, or two matching tokens meet at a node and disappear. Thus, to prove the lemma, we have to show that each node bordering face F is reached and that it is visited only once. A sequence of adjacent nodes of the face is a *visited segment* if each node has been visited at least once. A *border* of a visited segment is a visited node whose neighbor is not visited. By the design of the algorithm, a border node always has a token to send to its neighbor that is not visited. As we assume reliable message transmission, eventually the non-visited neighbor joins the visited segment. Thus, every node in a face with a visited segment is eventually visited.

Observe that the face bordering s has at least one visited segment: the one that contains s itself. Thus, every node in this face will eventually be visited. As graph G is connected, there is a sequence of adjacent faces intersecting the *sd*-line from the face bordering s to the face bordering d. Adjacent faces share a juncture node. Due to the algorithm design, when a juncture is visited in one face that intersects the *sd*-line, the juncture injects a pair of tokens in every adjacent face. That is, visiting a juncture node creates a visited segment in all adjacent faces. By induction, all nodes in the sequence of adjacent faces are visited, including the destination node.

Let us discuss whether a token may penetrate a visited segment and arrive at an interior (non-border) node. Observe that the computation of *CFR* starts with a single segment consisting of the source node. Thus, initially, there are no tokens inside any of the segments. Assume there are no internal tokens in this computation up to some step x. Let us consider the next step. The

token may penetrate the segment only through a border node or through an interior junction node. A token may arrive at a border node b only from the border node of another segment of the same face. Because b is a border node, it already holds the token of the opposite traversal direction. These two tokens are matching. Thus, b destroys both tokens and the received token does not propagate to the interior nodes. Let us consider a juncture node j. Because j is interior to the segment, it was visited earlier. When a juncture node receives a token, it creates a pair of tokens in all adjacent faces. That is, once a juncture is visited, it becomes visited in all adjacent faces at once. Since we assumed that there are no internal tokens up to step x, j cannot receive a token. By induction, a token may not penetrate a segment. That is, each node bordering a face is visited at most once. This completes the proof of the lemma. □

The below theorem follows from Lemma 1.

Theorem 1. *Algorithm CFR guarantees the delivery of a message from s to d.*

According to Lemma 1, the total number of messages sent in a computation is equal to the sum of the incident edges of the faces intersecting the *sd*-line. Note that an edge can be incident to at most two faces. That is, the total number of messages sent throughout the computation is at most $2|E|$. Hence, the following corollary.

Corollary 1. *The worst case message complexity of* CFR *is* $O(|E|)$.

Theorem 2. *The latency of* CFR *is asymptotically optimal and is within* $O(\rho^2)$ *where ρ is the number of hops in the shortest path in between the source and destination in the planar subgraph of G.*

Proof: The proof of the theorem parallels the optimality proof of GOAFR [21]. Let us consider the upper bound on the latency first. Kuhn et al argue (see [21, Lemma 5.4]) that to derive a bound it is sufficient to consider a bounded degree traversal graph. If the degree of the graph is unbounded, a bounded degree connected dominating set subgraph can always be locally constructed. Since it takes just one hop to reach this subgraph from any point in the graph, the path length over general graph is only 2 hops more than the length of the path over this subgraph. Let k be the maximum node degree in the traversal graph.

Since the graph to be traversed is a unit-disk graph, if ρ is the number of hops in the shortest path between s and d, then the Euclidean distance between the two points is no more than ρ. Let us consider a disk $D(d, \rho)$ with radius ρ centered in d. Since the shortest path between s and d is no longer than ρ, this path lies completely inside the disk . Note that the shortest path intersects *sd*-line at least twice: at the source and destination node. Let us consider two consequent points of intersection. Refer to Figure 5 for illustration. Since the graph is planar, the segment of the path between these points, includes the borders of all faces that intersect the *sd*-line and lie on the same side of the line as the shortest path segment (see figure). Since *CFR* traverses these faces,

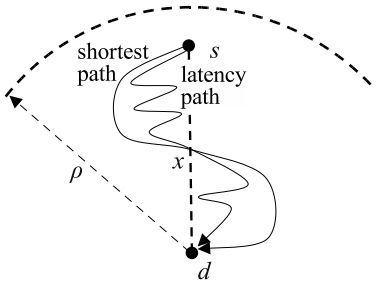


Figure 5: Illustration for the proof of optimality of *CFR*.

there is a path selected by *CFR* whose segment is completely enclosed by sd-line on one side and this shortest path segment on the other. Examining all such segments of the shortest path, we observe that there is a path of *CFR* that is completely enclosed in the disk $D(d, \rho)$.

Let us estimate the length of this path. Kuhn et al argue (see [21, Figure 2]), that the whole plane can be covered by disks of a diameter of one unit by placing them on a square grid with sides $1/\sqrt{2}$. Let us determine how many such squares cover $D(d, \rho)$. Each square that intersects $D(d, \rho)$ lies completely within $D(d, \rho + 1)$. Thus, the number of such squares is no more than

$$\frac{\pi(\rho+1)^2}{(1/\sqrt{2})^2} = 2\pi(\rho+1)^2$$

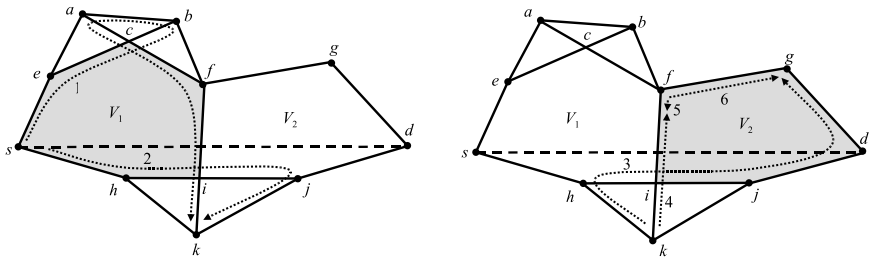
Recall that the graph is unit-disk and all nodes within the unit distance are connected. The graph is of degree k. Thus, the maximum number of nodes in a single disk of diameter one, is k. Therefore, the number of nodes inside $D(d, \rho)$ is no more than $2k\pi(\rho+1)^2$.

Note that there is a path, selected by *CFR* that lies completely inside $D(d, \rho)$. Observe that according to Lemma 1 a message of *CFR* can visit the same node at most k times. Thus, the length of this path of *CFR* is no more than $2k^2\pi(\rho+1)^2$ which is in $O(\rho^2)$.

The asymptotic optimality of *CFR* follows from the lower bound established by Kuhn et al [22, Theorem 5.1]. □

4. *CFR* Application and Extensions

Combining with greedy routing, using various traversal types. For efficiency, a single direction face traversal may be combined with greedy routing as in *GFG* or *GOAFR+*. Algorithm *CFR* can be used in a similar combination. We call the combined algorithm *GCFR*. The message starts in greedy mode and switches to *CFR* once it reaches a local minimum. Because multiple messages traverse the graph simultaneously, unlike *GFG*, once the message switches to face traversal in *GCFR*, it continues in this mode until the destination is reached.



(a) traversing V_1 (b) traversing V_2

Figure 6: Example of *CVR* operation on a non-planar graph.

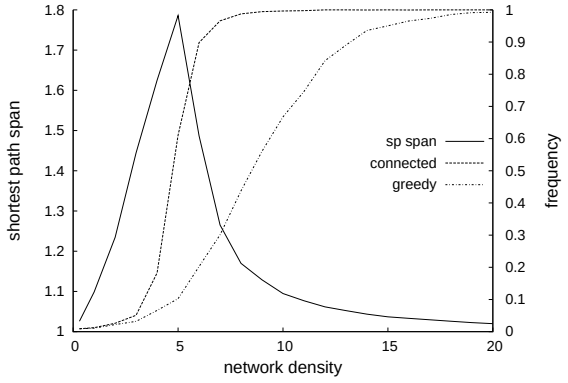
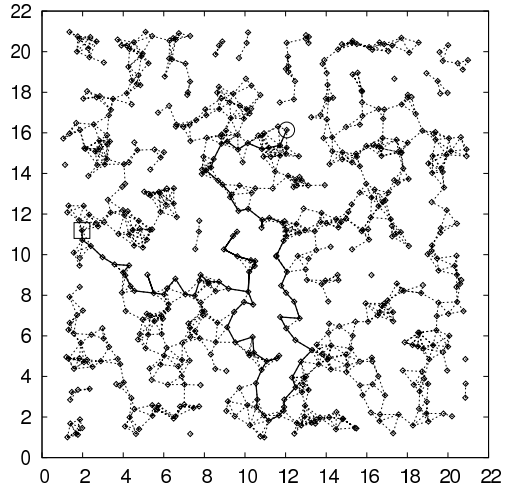


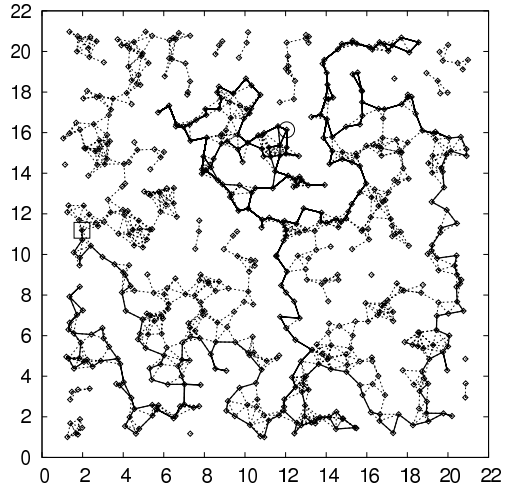
Figure 7: Graph parameters depending on its density. Shortest path span (ratio between Euclidean and path distance), ratio of connected graphs, and rate of success of pure greedy routing. Last two plotted against the right y axis.

Using non-planar graphs. *CFR* can be adapted to concurrent void traversal [29]. The resultant algorithm is *CVR*. *CVR* can also be combined with greedy routing to form *GCVR*. Before we describe the necessary changes let us recall how void traversal operates. Note that void traversal is performed over segments of edges adjacent to the void, rather than over complete edges. After getting the message, two nodes c and f (see Figure 2 again), adjacent to the edge (c, f) that contains the segment (d, e), jointly determine the edge whose intersection point produces the shortest segment in the traversal direction. Then, the token is forwarded to one of the nodes adjacent to the new edge (b, h). In the example node f forwards the token to h. Observe that in a non-planar graph f and h may be more than one hop apart.

Similar operations happen during the concurrent traversal in *CVR*. However, care must be taken to ensure that mates find each other. In particular a mate traversing the same face might be traveling along the path connecting f and



(a) CFR, 102 hops



(b) OAFR, 1051 hops

Figure 8: Latency paths selected by CFR and OAFR (shown in solid lines). The source and destination nodes are marked by a circle and a square respectively. Graph density is 5 nodes per unit disk.

h. Thus, h and f have to agree on the forwarding path and the tokens have to carry enough information to recognize their mates. Another complication to be resolved is the treatment of junctures. For *CVR*, a juncture is the node incident to the edge whose segment intersects the *sd*-line. Note that, unlike planar graphs, the segments can intersect at points other than nodes. Thus, the segment intersection point itself may potentially lie on the *sd*-line. This case generates multiple junctures. However, the mates generated by these junctures meet and destroy each other.

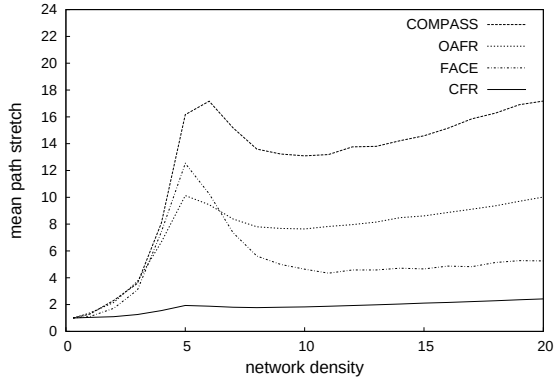
Refer to Figure 6 for an illustration of *CVR* operation. To simplify the presentation we show the traversal of the two adjacent voids V_1 and V_2 separately in Figures 6(a) and 6(b) respectively. As before, to avoid cluttering the picture, we only show the token numbers. We explain the traversal of V_1 in detail. The traversal starts when s sends two tokens t_1 and t_2 in the opposite directions around V_1. When t_1 arrives at e, the nodes incident to edge (e, b) have to determine the edge that intersects (e, b) closest to the beginning of the segment. In this case the beginning of the segment is node e itself. Node e sends t_1 to b and the two nodes determine that the appropriate edge is (a, f). Therefore, b forwards t_1 to a which is one of the nodes incident to (a, f). Node a forwards t_1 to f. Note that f is a juncture. Hence, f injects a pair of tokens: t_5 and t_6 into V_2. After that, f forwards t_2 to k. Note that k is also a juncture. Hence, k injects another pair of tokens: t_3 and t_4 into V_2. Meanwhile, t_1 reaches h. To determine the segment of (h, j) that is adjacent to V_1, h forwards t_1 to j. The intersecting edge correctly determined, j forwards the message to k where it meets its mate — t_2. This concludes the traversal of V_1. The traversal of V_2 is completed similarly.

5. Performance Evaluation

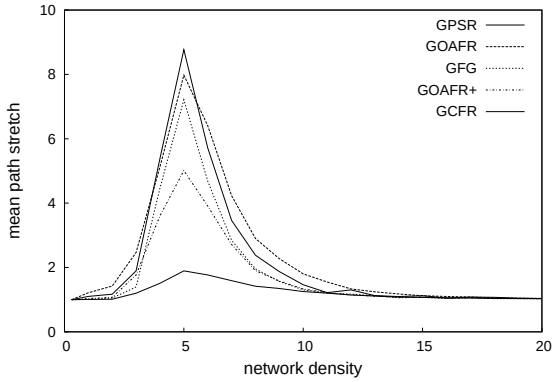
Simulation environment. To evaluate the performance of *CFR* we recreated the simulation environment used by Kuhn et al [21, 24]. This model is the most comprehensive way of evaluating the performance of geometric routing algorithms that we are aware of. The practical behavior of the algorithms primarily depends on the network density. This model examines the operation of the algorithms across all network densities of interest.

For the simulation, we used the graphs formed by uniformly placing the nodes at random on a 20×20 unit square field. The number of nodes depended on the selected density. The edges of the graph were selected according to the unit-disk model: two nodes are connected if and only if they are within the unit-distance of each other. For each graph, a single source and destination pair was randomly selected. We used 21 different density levels. To validate our environment we measured the same preliminary graph parameters as in Kuhn et al [21, Figure 3],[24, Figure 3]. For each density level we carried out $2,000$ measurements. Our results are plotted in Figure 7. They concur with the previous studies.

Evaluation description. We implemented *CFR* and compared its perfor-



(a) Pure geometric routing.

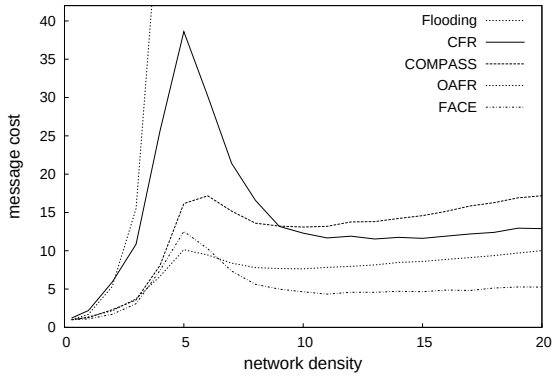


(b) Greedy and geometric combined.

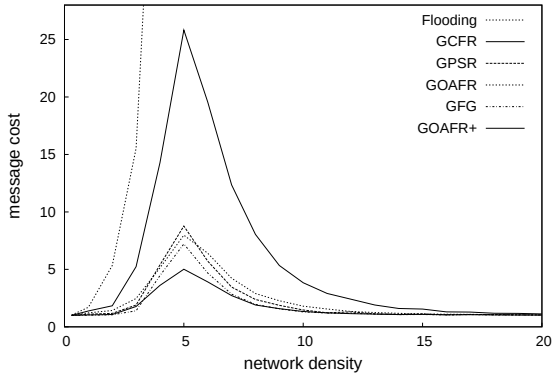
Figure 9: Mean path stretch (ratio of the path selected by the algorithm to the shortest unit-disk graph path) of geometric routing algorithms on planar graphs depending on the density (nodes per unit disk) of the unit disk graph.

mance against the major known geometric routing algorithms. We took $2,000$ measurements at each graph density level. Refer to Figure 8 for an illustration of the resultant graphs and algorithm path selections.

Let us first compare the speed of communication demonstrated by the routing algorithms. In Figure 9 we plot the path stretch achieved by the algorithms in pure form and in combination with greedy routing. Figure 9(a) indicates that pure *CFR* outperforms all the other algorithms. In the critical range, the path stretch that the pure *CFR* provides is up to five times better than the next best algorithm's — *OAFR*. Let us consider the combination of greedy and face routing. Recall that, unlike the other algorithms, after switching from greedy to face traversal mode, *GCFR* does not switch back to greedy again. Thus, *GCFR* may miss on an efficient path selected by greedy routing. However, as the graph density increases, the greedily routed message may not encounter a local min-



(a) Pure geometric routing.

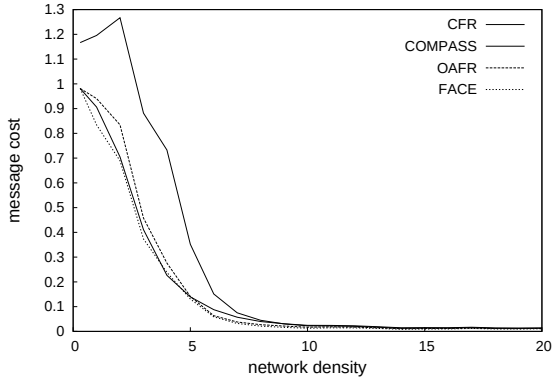


(b) Greedy and geometric combined.

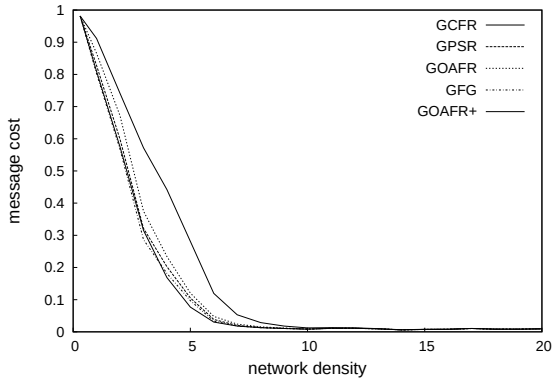
Figure 10: Mean message cost normalized to shortest path of geometric routing algorithms on unit-disk graphs depending on density (nodes per unit disk) of the unit disk graph.

imum altogether. Therefore, the number of such mode switches decreases and this potential disadvantage of *GCFR* is offset. As Figure 9(b) indicates, the path stretch produced by *GCFR* in the critical region is still over 2.5 times better than the next best algorithm.

Let us now consider the message cost of communication of the algorithms. In Figure 10 we show the message cost normalized to the shortest path while in Figure 11 the cost is normalized to flooding (i.e. every node sends exactly one message). The first presentation indicates the cost compared to the distance from source to destination, the second — compared to the whole system participation in the route discovery. The latter metric gives the perspective of cost of geometric routing compared to flooding-based routing algorithms [9, 17, 19, 28, 30]. Figure 10 shows that *CFR* and *GCFR* use more messages than other geometric routing algorithms. However, Figure 11 shows that message cost of *CFR* and



(a) Pure geometric routing.

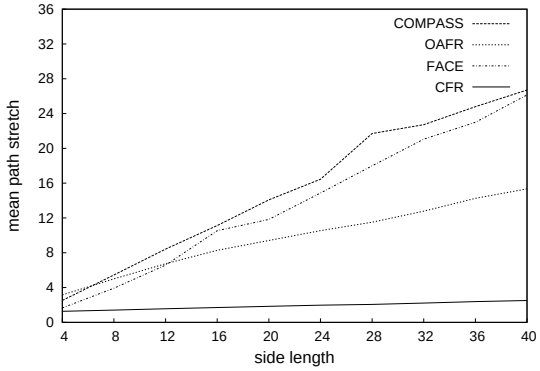


(b) Greedy and geometric combined.

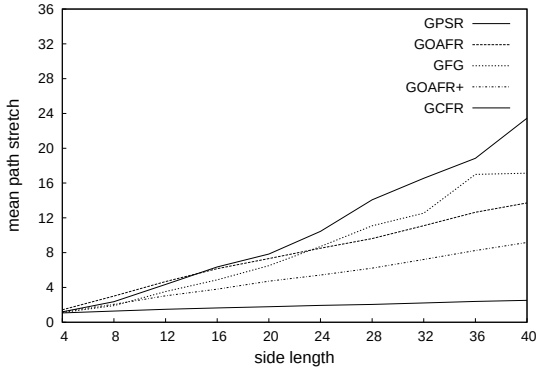
Figure 11: Mean message cost normalized to flooding of geometric routing algorithms on planar graphs depending on density (nodes per unit disk) of the unit disk graph.

GCFR are comparable to the other algorithms.

To study the effect of graph scale on the performance of geometric algorithms, we constructed the simulation scenario similar to that of Kuhn et al [24, Figure 10]. We fixed the density of the graph near the critical value — at 4.5; and varied the field size. Specifically, we selected 10 different lengths of the side of the square field from 4 to 40 units. The number of nodes in the field was selected to match the required density of 4.5. We took $3,000$ measurements for each side length. The results of the simulation are shown in Figure 12. Our simulation indicates that the path stretch achieved by *CFR* and *GCFR* is lower than that of the other routing algorithms at any scale. This is true for pure geometric routing and its combination with greedy routing. Moreover, as graph scale increases, compared to the other routing algorithms, *CFR* and *GCFR* exhibit significantly slower rate of path stretch increase.



(a) Pure geometric.



(b) Greedy and geometric combined.

Figure 12: Mean path stretch of routing algorithms on planar subgraphs of unit-disk graphs depending on graph scale with average graph density of 4.5. The graphs are constructed on square fields with side lengths from 4 to 40 units.

To demonstrate the viability of *CFR* on non-planar graphs, we implemented *CVR* and *GCVR* and compared their performance against conventional *VOID* and *GVG*. For these experiments we also used a 20×20 units square field randomly filled by the nodes with randomly selected source and destination pairs. However, the network was modeled as a quasi unit-disk graph [4, 23]. Specifically, two vertices of u and v: i) were definitely adjacent if $|u, v| \leq d = 0.75$; ii) were adjacent with probability $p = 0.5$ if $d < |u, v| \leq 1$; iii) definitely not adjacent if $|u, v| > 1$. We selected 21 density levels and carried out $2,000$ trials for each density level. Due to the limitations of double precision floating point calculations, some of the trials did not succeed: due to computation errors, the adjacent nodes may not agree on the edge intersection location. To ensure successful runs, for each graph we globally pre-computed all intersection points. The results are shown in Figure 13. Our results indicate that *CFR* retains its

latency advantages over the other algorithms in non-planar graphs.

6. Conclusion

The CFR algorithm presented in this paper improves both the bounds and the practical performance of geometric routing algorithms. Moreover, CFR addresses one of the major drawbacks of geometric routing: its inconsistency due to selection of disadvantageous routes. The proposed technique is simple to implement. The authors are hopeful that it will quickly find its way into practical implementations of geometric routing algorithms.

Note that classic geometric routing algorithms only consider two-dimensional communication graph embeddings. Recently, there appeared studies extending these techniques to three dimensions (see for example Liu and Wu [27]). It would be interesting to investigate how our algorithm can be extended to accommodate three dimensional routing.

Another important research direction is message optimization. Our algorithm accelerates message delivery at the expense of greater number transmitted messages. The next challenge is to preserve the fast message delivery while trying to improve the message overhead.

[1] I. Abraham, D. Dolev, and D. Malkhi. LLS: a locality aware location service for mobile ad hoc networks. In *Proceedings of the Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 75–84, Philadelphia, USA, October 2004.

[2] U.G. Acer, S. Kalyanaraman, and A.A. Abouzeid. Weak state routing for large scale dynamic networks. In *Proceedings of the 13th annual ACM international conference on Mobile Computing and Networking (MobiCom)*, pages 290–301, September 2007.

[3] N. Arad and Y. Shavitt. Minimizing recovery state in geographic ad hoc routing. *IEEE Transactions on Mobile Computing*, 8(2), 2009.

[4] L. Barrière, P. Fraigniaud, L. Narayanan, and J. Opatrny. Robust position-based routing in wireless ad hoc networks with irregular transmission ranges. *Wireless Communications and Mobile Computing*, 3(2):141–153, 2003.

[5] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *The Journal of Mobile Communication, Computation and Information*, 7(6):48–55, 2001.

[6] N. Bulusu, J. Heidemann, D. Estrin, and T. Tran. Self-configuring localization systems: Design and experimental evaluation. *ACM Transactions on Embedded Computing Systems*, 3(1):24–60, February 2004.

[7] S. Datta, I. Stojmenovic, and J. Wu. Internal node and shortcut based routing with guaranteed delivery in wireless networks. *Cluster Computing*, 5(2):169–178, April 2002.

[8] H. Dubois-Ferriere, M. Grossglauser, and M. Vetterli. Age matters: Efficient route discovery in mobile ad hoc networks using encounter ages. In

Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing MobiHoc, pages 257–266, June 2003.

[9] G.G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISI/RR-87-180, March 1987.

[10] H. Frey and I. Stojmenovic. On delivery guarantees of face and combined greedy-face routing in ad hoc and sensor networks. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking, MOBICOM 2006, Los Angeles, CA, USA, September 23-29, 2006*, pages 390–401. ACM, 2006.

[11] S. Giordano, I. Stojmenovic, and L. Blazevic. Position based routing algorithms for ad hoc networks - a taxonomy. *Ad Hoc Wireless NetWorking*, pages 103–136, January 2004.

[12] Peng He, Jiandong Li, and Lei Zhou. A novel geographic routing algorithm for ad hoc networks based on localized delaunay triangulation. In *AINA*, pages 49–54. IEEE Computer Society, 2006.

[13] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, 2001.

[14] B. Karp and H.T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobille Computing and Networking (MobiCom)*, pages 243–254. ACM Press, August 2000.

[15] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker. Geographic routing made practical. In *2nd Symposium on Networked Systems Design and Implementation (NSDI)*, Boston, MA, USA, May 2005.

[16] Y-J. Kim, R. Govindan, B. Karp, and S. Shenker. On the pitfalls of geographic face routing. In *3d ACM/SIGMOBILE International Workshop on Foundations of Mobile Computing (DIAL-M-POMC)*, pages 34–43, 2005.

[17] V.B. Ko and N.H. Vaidya. Location-aided routing LAR in mobile ad-hoc networks. *In Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 66–75, October 1998.

[18] Y.-B. Ko and N.H. Vaidya. Geocasting in mobile ad hoc networks: Location-based multicast algorithms. In *2nd Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 101–110, New Orleans, LA, February 1999. IEEE Computer Society.

[19] Y.-B. Ko and N.H. Vaidya. GeoTORA: A protocol for geocasting in mobile ad hoc networks. In *8th International Conference on Network Protocols (ICNP)*, pages 240–251, Osaka, Japan, may 2000.

[20] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proc. 11 th Canadian Conference on Computational Geometry*, pages 51–54, Vancouver, August 1999.

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

[22] F. Kuhn, R. Wattenhofer, and A. Zollinger. Asymptotically optimal geometric mobile ad-hoc routing. In *6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM)*, Atlanta, Georgia, USA, September 2002.

[23] F. Kuhn, R. Wattenhofer, and A. Zollinger. Ad-hoc networks beyond unit disk graphs. In *Joint Workshop on Foundations of Mobile Computing (DialM-POMC)*, pages 69–78, San Diego, CA, USA, September 2003.

[24] F. Kuhn, R. Wattenhofer, and A. Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *4th International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 267–278, Annapolis, Maryland, USA, June 2003.

[25] B. Leong, B. Liskov, and R. Morris. Geographic routing without planarization. In *3rd Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, USA, May 2006.

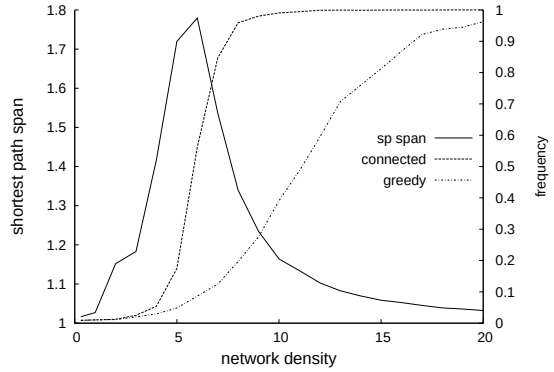
[26] J. Li, J. Jannotti, D.S.J. De Couto, D.K. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 120–130, August 2000.

[27] C. Liu and J. Wu. Efficient geometric routing in three dimensional ad hoc networks. In *INFOCOM*, pages 2751–2755. IEEE, 2009.

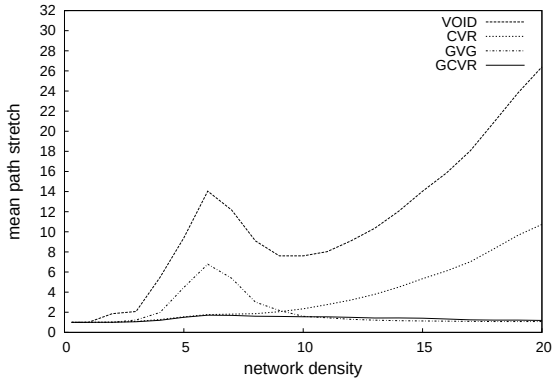
[28] K. Seada and A. Helmy. Efficient geocasting with perfect delivery in wireless networks. *IEEE Wireless Communications and Networking Conference (WNCN 2004)*, page 6, December 2004.

[29] A. Vora and M. Nesterenko. Void traversal for guaranteed delivery in geometric routing. *The 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2005)*, pages 63–67, November 2005.

[30] J. Wu and F. Dai. A generic broadcast protocol in ad hoc networks based on self-pruning. In *17th International Parallel and Distributed Processing Symposium (IPDPS)*, pages 29–29, Los Alamitos, CA, April 22–26 2003. IEEE Computer Society.



(a) Graph parameters depending on its density. Shortest path span, ratio of connected graphs rate of success of greedy routing.



(b) Mean path stretch.

Figure 13: Performance evaluation of geometric routing algorithms on non-planar quasi unit-disk graphs. The distance of definite connectivity is 0.75 unit; possible connectivity between 0.75 and 1 unit; no connectivity above 1 unit.