Route Discovery with Constant Memory in Oriented Planar Geometric Networks

E. Chávez[1], S. Dobrev[2], E. Kranakis[3], J. Opatrny[4], L. Stacho[5], and J. Urrutia[6]

[1] Escuela de Ciencias Fisico-Matemáticas de la Universidad Michoacana de San Nicolás de Hidalgo, México.
[2] School of Information Technology and Engineering (SITE), University of Ottawa, 800 King Eduard, Ottawa, Ontario, Canada, K1N 6N5. Research supported in part by NSERC (Natural Science and Engineering Research Council of Canada) grant.
[3] School of Computer Science, Carleton University, 1125 Colonel By Drive, Ottawa, Ontario, Canada K1S 5B6. Research supported in part by NSERC (Natural Science and Engineering Research Council of Canada) and MITACS (Mathematics of Information Technology and Complex Systems) grants.
[4] Department of Computer Science, Concordia University, 1455 de Maisonneuve Blvd West, Montréal, Québec, Canada, H3G 1M8. Research supported in part by NSERC (Natural Science and Engineering Research Council of Canada) grant.
[5] Department of Mathematics, Simon Fraser University, 8888 University Drive, Burnaby, British Columbia, Canada, V5A 1S6. Research supported in part by NSERC (Natural Science and Engineering Research Council of Canada) grant.
[6] Instituto de Matemáticas, Universidad Nacional Autónoma de México, Área de la investigación cientifica, Circuito Exterior, Ciudad Universitaria, Coyoacán 04510, México, D.F. México

Abstract. We address the problem of discovering routes in strongly connected planar geometric networks with directed links. We consider two types of directed planar geometric networks: Eulerian (in which every vertex has the same number of ingoing and outgoing edges) and outerplanar (in which a single face contains all the vertices of the network). Motivated by the necessity for establishing communication in wireless networking based only on geographic proximity, in both instances we give algorithms that use only information that is geographically local to the vertices participating in the route discovery.

1 Introduction

The most extensively used model of wireless network has bidirectional links in the sense that packets may flow in any direction between any pair of adjacent vertices. This type of connectivity arises by considering a network in which all wireless hosts have the same transmission power, thus resulting in identical reachability radii. Such a network can be represented by a unit disk graph model.

A most important consideration is how to discover a route (among the many potential candidates) that must accommodate three rather contradictory goals: (1) avoiding flooding the network, (2) efficiency of the resulting path, (3) using only geographically local information. Although it is generally accepted that flooding must be avoided in order to increase the network lifetime, limitations on the knowledge of the hosts make it necessary that in practice one may need to drop the second goal in favour of the third. Indeed, this is the case in geometric planar networks whereby algorithms are given for discovering routes (see Kranakis et al. [8]). Another important consideration is to construct a planar geometric network from a rather complicated wireless network. This question is addressed in Bose et al. [4] as well as in several subsequent papers. The basic idea is to preprocess the wireless network in order to abstract a planar geometric network over which the algorithm in Kranakis et al. [8] (compass routing, face routing, and other related routing algorithms) can be applied.

1.1 A Fundamental Issue

In principle, routing must be preceded by a radiolocation based vertex discovery process relying on an available Geographic Positioning System (or GPS) that will enable vertices to discover their neighbors. Although vertex discovery is not the purpose of study of this paper it is important to note that bidirectional communication will be rarely valid in practice. This may be due to several factors, including obstacles that may either obstruct direct view to a host and/or diminish the strength of a signal during propagation or even wireless hosts with different power capabilities.

A fundamental issue is whether geographically local routing is possible in arbitrary graphs. If the underlying network is either not planar or its links are not bidirectional the techniques outlined above may fail to route and/or traverse the network using only "geographically" local information. In general this is expected to be a difficult problem because one will never be able to avoid "loops" based only on geographically local information. Therefore there is a need for reexamination of the structure of the underlying backbone network that gives rise to our basic communication model in order to accomplish routing satisfying the previously set conditions.

1.2 Related Literature

There has been extensive literature related to discovering routes in wireless adhoc networks when the underlying graph is a undirected planar geometric network, e.g., see Bose et al. [4], Kranakis et al. [8], Kuhn et al. [9,10]. A problem related to routing is traversal which is addressed in several papers Avis et al. [1], Bose et al. [3], Chavez et al. [5], Czyczowicz et al. [6], Gold et al. [7], Peuquet et al. [11,12]. However, traversing all the vertices of a graph may be an "overkill" especially if all one requires is a single path from a source to a destination host.

1.3 Results of the Paper

In this paper we address the problem of discovering routes in strongly connected planar geometric networks with directed links using only local information. After clarifying the model in Section 2 we consider two types of directed planar geometric networks. In Section 3 we look at Eulerian planar geometric networks in which every vertex has the same number of ingoing and outgoing edges. In Section 4 we investigate outerplanar geometric networks whereby a single face contains all the vertices of the network.

2 Model

A *planar geometric network* is a planar graph G with vertex set V, edge set E and the face set F together with its straight line embedding into the plane $\mathbb{E}^2$. In this paper we always consider only finite graphs. Furthermore, we assume that no edge passes through any vertex except its end-vertices. A geometric network is *connected* if its graph is connected. An *orientation* of a planar geometric network G is an assignment of a direction to every edge e of G. For an edge e with endpoints u and v, we write $e = (u, v)$ if its direction is from u to v. The geometric network together with its orientation is denoted by $\boldsymbol{G}$.

We assume that every vertex v is uniquely determined by the pair $[x, y]$ where x is its horizontal coordinate and y is its vertical coordinate.

Consider a connected planar geometric network $\boldsymbol{G} = (V, E)$. We say $\boldsymbol{G}$ is *Eulerian* if for every vertex $u \in V$, the size of $N^+(u) = \{x,\ (u, x) \in E\}$ equals the size of $N^-(u) = \{y,\ (y, u) \in E\}$; i.e. the number of edges outgoing from u equals the number of edges ingoing into u.

3 Route Discovery in Eulerian Planar Geometric Networks

First consider a planar geometric network G without any orientation. Given a vertex v on a face f in G, the boundary of f can be traversed in the counter-clockwise (clockwise if f is the outer face) direction using the well-known right hand rule [2] which states that it is possible to visit every wall in a maze by keeping your right hand on the wall while walking forward. Treating this face traversal technique as a subroutine, Kranakis et al. [8] give an elegant algorithm for routing in a planar geometric network from a vertex s to a vertex t.

If we impose an orientation on G, then this algorithm will not work since some edges may be directed in an opposite direction while traversing a face. In this section, we describe a simple technique on how to overcome this difficulty. In particular, we propose a method for routing a message to the other end of an oppositely directed edge in Eulerian geometric networks.

Now suppose is so that $\boldsymbol{G}$ is an Eulerian planar geometric network. For a given vertex u of $\boldsymbol{G}$, we order edges (u, x) where $x \in N^+(u)$ clockwise around u starting with the edge closest to the vertical line passing through u. Similarly we order edges (y, u) where $y \in N^-(u)$ clockwise around u; see Figure 1. Clearly, this orderings are unique and can be determined locally at each vertex.

Let $e = (y, u)$ be the i-th ingoing edge to u in $\boldsymbol{G}$. The function **succ**(e) will return a pointer to the edge (u, x) so that (u, x) is the i-th outgoing edge from

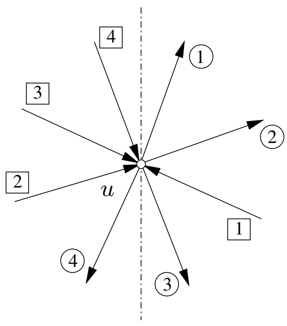

Fig. 1. Circled numbers represent the ordering on outgoing edges, squared numbers on ingoing ones.

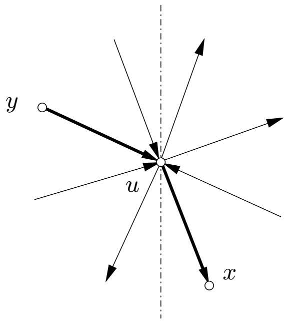

Fig. 2. In this example the ingoing edge (y, u) is third, so the chosen outgoing edge (u, x) is also third. Both these edges are depicted bold.

u. For an illustration of the function see Figure 2. Again, this function is easy to implement using only local information.

Obviously, the function $\mathbf{succ}()$ is injective, and thus, for every edge $e = (u, v)$ of $\boldsymbol{G}$, we can define a closed walk by starting from $e = (u, v)$ and then repeatedly applying the function $\mathbf{succ}()$ until we arrive at the same edge $e = (u, v)$. Since $\boldsymbol{G}$ is Eulerian, the walk is well defined and finite. We call such a walk a *quasi-face* of $\boldsymbol{G}$.

The following is the route discovery algorithm from [8] for planar geometric networks. We modify it so that it will work on Eulerian planar geometric network. For this, we only need to extend the face traversal routine as follows:

> Whenever the face traversal routine wants to traverse an edge $e = (u, v)$ that is oppositely directed, we traverse the following edges in this order:
>
> $$\mathbf{succ}(e), \mathbf{succ}(e)^2, \dots, \mathbf{succ}(e)^k,$$
>
> so that $\mathbf{succ}(e)^{k+1} = (u, v)$. After traversing $\mathbf{succ}(e)^k$, the routine resumes to the original traversal of the face.

This modification obviously guarantees (in Eulerian geometric networks) that all edges of the face will be visited.

Algorithm 1 Eulerian Geometric Network Route Discovery.
Input: Connected Eulerian geometric network $\boldsymbol{G} = (V, E)$
Starting vertex: s
Destination vertex: t

```
1: $v \leftarrow s$ {Current vertex = starting vertex.}
2: repeat
3:    Let $f$ be a face of $G$ with $v$ on its boundary that intersects the line $v$-$t$ at
      a point (not necessarily a vertex) closest to $t$.
4:    for all edges $xy$ of $f$ do
5:       if $xy \cap v\text{-}t = p$ and $\mathbf{dist}(p,t) < \mathbf{dist}(v,t)$ then
6:          $v \leftarrow p$
7:       end if
8:    end for
9:    Traverse $f$ until reaching the edge $xy$ containing the point $p$.
10: until $v = t$
```

Theorem 1. *Algorithm 1 will reach t from s in at most $O(n^2)$ steps.*

Proof. Follows from the proof of the correctness of the traversal algorithm for planar geometric networks from [8] and from the discussion above. The bound on the number of steps follows from the $O(n)$ bound on the number of steps of the algorithm from [8] and the fact that every edge on the route can be oppositely oriented and may need up to $O(n)$ steps to route through.

Examples which show that the bound cannot be improved in general are easy to construct, they typically include a large face that needs to be traversed whose boundary contains $\Theta(n)$ edges which are all oriented the opposite way.

4 Route Discovery in Strongly Connected Outerplanar Geometric Networks

A planar geometric network G is *outerplanar* if one of the elements in F contains all the vertices—the outerface. We will assume that this face is a convex polygon in $\mathbb{E}^2$. For a given triple of vertices x, y, and z, let $V_{\curvearrowleft}(x,y,z)$ [$V_{\curvearrowright}(x,y,z)$, resp.] denote the ordered set of vertices distinct from x and z that are encountered while moving from y counterclockwise [clockwise, resp.] around the outerface of G until either x or z is reached; see Figure 3.

Now consider an orientation   of the geometric network G. Let $N_{\curvearrowleft}(x,y,z) = V_{\curvearrowleft}(x,y,z) \cap N^+(x)$ and let $N_{\curvearrowright}(x,y,z) = V_{\curvearrowright}(x,y,z) \cap N^+(x)$. If $N_{\curvearrowleft}(x,y,z) \neq \emptyset$, let $v_{\curvearrowleft}(x,y,z)$ denote the first vertex in $N_{\curvearrowleft}(x,y,z)$. Similarly we define $v_{\curvearrowright}(x,y,z)$ as the first vertex in $N_{\curvearrowright}(x,y,z)$, if it exists. A geometric network with fixed orientation is *strongly connected* if for every ordered pair of its vertices, there is a (directed) path joining them.

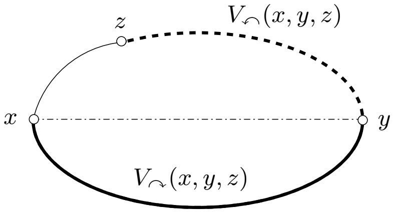

Fig. 3. The dashed part of the outer face represents the vertices in $V_{\curvearrowleft}(x,y,z)$ and the bold solid part represents vertices in $V_{\curvearrowright}(x,y,z)$, respectively. Note that y belongs to both these sets and is in fact the first element of those sets.

Algorithm 2 Outerplanar Geometric Network Route Discovery.
Input: Strongly connected outerplanar geometric network $\boldsymbol{G} = (V, E)$
Starting vertex: s
Destination vertex: t

```
1: $v \leftarrow s$ {Current vertex = starting vertex.}
2: $v_{\curvearrowleft}, v_{\curvearrowright} \leftarrow s$ {counterclockwise and clockwise bound = starting vertex.}
3: while $v \neq t$ do
4:    if $(v,t) \in E$ then
5:       $v, v_{\curvearrowleft}, v_{\curvearrowright} \leftarrow t$ {Move to $t$.}
6:    else if $N_{\curvearrowleft}(v,t,v_{\curvearrowleft}) \neq \emptyset$ and $N_{\curvearrowright}(v,t,v_{\curvearrowright}) = \emptyset$ then {No-choice vertex;
      greedily move to the only possible counterclockwise direction toward $t$.}
7:       $v, v_{\curvearrowleft} \leftarrow v_{\curvearrowleft}(v,t,v_{\curvearrowleft})$
8:    else if $N_{\curvearrowleft}(v,t,v_{\curvearrowleft}) = \emptyset$ and $N_{\curvearrowright}(v,t,v_{\curvearrowright}) \neq \emptyset$ then {No-choice vertex;
      greedily move to the only possible clockwise direction toward $t$.}
9:       $v, v_{\curvearrowright} \leftarrow v_{\curvearrowright}(v,t,v_{\curvearrowright})$
10:   else if $N_{\curvearrowleft}(v,t,v_{\curvearrowleft}) \neq \emptyset$ and $N_{\curvearrowright}(v,t,v_{\curvearrowright}) \neq \emptyset$ then {Decision vertex; first
      take the "counterclockwise" branch but remember the vertex for the backtrack
      purpose.}
11:      $b \leftarrow v$;   $v, v_{\curvearrowleft} \leftarrow v_{\curvearrowleft}(v,t,v_{\curvearrowleft})$
12:   else if $N_{\curvearrowleft}(v,t,v_{\curvearrowleft}) = \emptyset$ and $N_{\curvearrowright}(v,t,v_{\curvearrowright}) = \emptyset$ then {Dead-end vertex;
      backtrack to the last vertex where a decision has been made. No updates to $v_{\curvearrowleft}$
      and $v_{\curvearrowright}$ are necessary.}
13:      if $v \in V_{\curvearrowleft}(t,b,t)$ then
14:         while $v \neq b$ do
15:            $v \leftarrow v_{\curvearrowleft}(v,b,v)$
16:         end while
17:      end if
18:      if $v \in V_{\curvearrowright}(t,b,t)$ then
19:         while $v \neq b$ do
20:            $v \leftarrow v_{\curvearrowright}(v,b,v)$
21:         end while
22:      end if
23:      $v, v_{\curvearrowright} \leftarrow v_{\curvearrowright}(v,t,v_{\curvearrowright})$ {Take the "clockwise" branch toward $t$.}
```

```
24:     end if
25: end while
```

Note 1. The implementation of tests in lines 6, 8, 10, 12 is simple. Since the vertices of G are in convex position, one can easily (and locally—remembering only two best candidates, one for each direction) compute the first vertex in $N^+(v)$ that is in clockwise (resp. counterclockwise) direction from t or to determine that such vertex does not exist. Similarly, tests in lines 13 and 18 are simple to implement and require constant memory.

Lemma 1. *Suppose Algorithm 2 reaches a decision vertex b (line 11). Next suppose that $v_1, v_2, \dots, v_k$ are vertices reached in subsequent steps and that all are no-choice vertices, i.e. determined at line 7 or 9. Finally suppose that next vertex reached is a dead-end vertex v_{k+1} (determined at line 12). Then vertices $v_1, v_2, \dots, v_{k+1}$ are all either in $V_{\curvearrowleft}(t,b,t)$ or in $V_{\curvearrowright}(t,b,t)$.*

Proof. By way of contradiction, suppose i is maximum so that v_i and v_{i+1} are not both in $V_{\curvearrowleft}(t,b,t)$ or in $V_{\curvearrowright}(t,b,t)$. We may suppose $v_i \in V_{\curvearrowright}(t,b,t)$ and $v_{i+1} \in V_{\curvearrowleft}(t,b,t)$ (The other case is analogous.), see Figure 4.

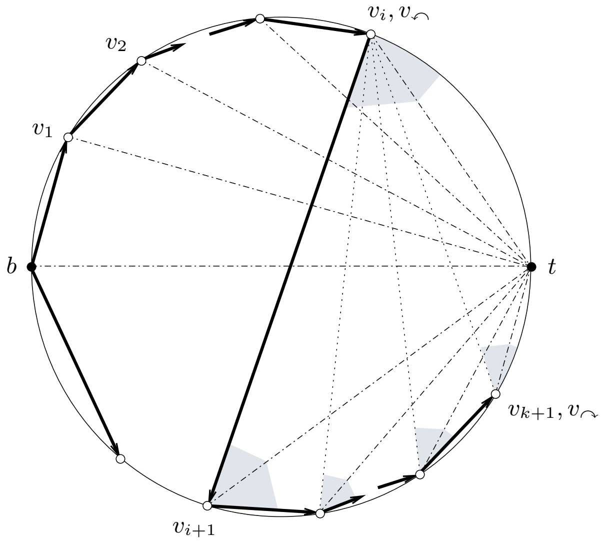

Fig. 4. Dashed parts at some vertices represent the area with no outgoing edges from the corresponding vertex in that direction. The exception are the three dotted lines which represent possible edges going into v_i. However these edges cannot help to reach t. Note that $V_{\curvearrowleft}(b,t,b) \setminus \{t\} = V_{\curvearrowright}(t,b,t) \setminus \{b\}$ and $V_{\curvearrowright}(b,t,b) \setminus \{t\} = V_{\curvearrowleft}(t,b,t) \setminus \{b\}$.

Since $\boldsymbol{G}$ is strongly connected, there must exist a path from b to t. Every such path must pass either through v_i or v_{i+1}. Since v_i is a no-choice vertex and since $v_{i+1} \in V_{\curvearrowleft}(t,b,t)$, such a path must always pass through v_{i+1}. By the choice of i, all vertices $v_{i+1}, v_{i+2}, \ldots, v_{k+1}$ are in $V_{\curvearrowleft}(t,b,t)$ and thus such a path must eventually pass through the vertex v_{k+1} and continue to a vertex in $N_{\curvearrowleft}(v_{k+1},t,v_i) \cup N_{\curvearrowright}(v_{k+1},t,v_{k+1})$. However, $N_{\curvearrowleft}(v_{k+1},t,v_i) = \emptyset$ and $N_{\curvearrowright}(v_{k+1},t,v_{k+1}) = \emptyset$, a contradiction.

Lemma 2. *Suppose Algorithm 2 reaches a dead-end vertex* d *(line 12). Then it will eventually return to the vertex* b *(last decision vertex defined in line 11).*

Proof. Suppose $v_1, v_2, \ldots, v_k$ are all vertices reached (in this order) after reaching the decision vertex b and before reaching the dead-end vertex $d = v_{k+1}$. By Lemma 1, we may assume $v_1, v_2, \ldots, v_{k+1} \in V_{\curvearrowright}(t,b,t)$. The other case is analogous. Since $\boldsymbol{G}$ is strongly connected, there must exist a (directed) path from d to b. Suppose by way of contradiction that the algorithm backtracks to some vertex $x \neq b$ for which $N_{\curvearrowright}(x,b,x) = \emptyset$. Suppose furthermore that x lies between v_i and v_{i+1} going from b to t around the outer face; see Figure 5.

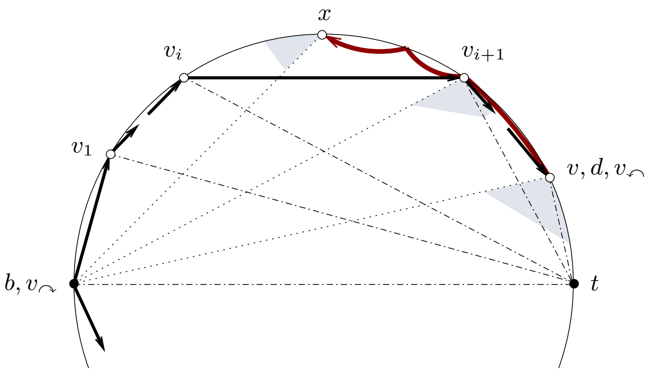

Fig. 5. Dashed parts at some vertices represent the area with no outgoing edges from the corresponding vertex in that direction. The bold curve from d to x represents the backtrack path.

By our assumption none of the edges $(v_{i+1},b), (v_{i+2},b), \ldots, (v_{k+1},b)$ exist, for otherwise the backtrack procedure would follow such an edge directly to b. Hence every path from d to b must eventually pass x and continue to a vertex in $N_{\curvearrowright}(x,b,x)$. However $N_{\curvearrowright}(x,b,x) = \emptyset$ by our assumption, a contradiction.

Lemma 3. *If the "counterclockwise" branch taken at a decision vertex leads to a dead-end vertex, then no dead-end vertex is reached on the "clockwise" branch at that vertex before reaching a new decision vertex.*

Proof. The proof is similar to the two previous proofs. If $b, s_1, s_2, \ldots, s_k$ where s_k is a dead-end vertex is the counterclockwise branch at b and $b, n_1, n_2, \ldots, n_l$

where n_l is a dead-end vertex is the clockwise branch at b, then by Lemma 1, vertices $s_1, s_2, \ldots, s_k \in V_{\curvearrowleft}(b,t,b)$ and vertices $n_1, n_2, \ldots, n_l \in V_{\curvearrowright}(b,t,b)$. Now it is clear that every (directed) path from b to t must pass either through s_k or n_l and then continue to a vertex either in $N_{\curvearrowleft}(s_k,t,b) \cup N_{\curvearrowright}(s_k,t,b)$ or in $N_{\curvearrowleft}(n_l,t,b) \cup N_{\curvearrowright}(n_l,t,b)$. However all these sets are empty by our assumption, a contradiction.

Lemma 4. *At each step of Algorithm 2 which is not the backtracking step, either* $v_{\curvearrowleft}$ *or* $v_{\curvearrowright}$ *is moved closer to* t *(measured as the graph distance on the outer face of* G.

Proof. This follows directly from the definition of $N_{\curvearrowleft}(v,t,v_{\curvearrowleft})$ and $N_{\curvearrowright}(v,t,v_{\curvearrowright})$ and the update performed at lines 5, 7, 9, 11, and 23, respectively.

Theorem 2. *Algorithm 2 will reach* t *from* s *in at most* $2n-1$ *steps.*

Proof. The proof that the algorithm will reach the destination vertex t follows from the above lemmas. The fact that no more than $2n-1$ steps are needed follows from the fact that G has at most $2n-1$ edges, that the algorithm process an edge at each step, and the fact that no edge is processed twice.

5 Conclusion

Routing in oriented ad-hoc networks is much more difficult than routing in non-oriented networks. Except for flooding, there seems to be no simple extension of the known routing algorithms that would be applicable to general oriented networks. In this paper we give routing algorithms for two cases of oriented planar networks: Eulerian and Outerplanar. No doubt, routing in oriented ad-hoc networks is far from being settled and further progress in this area is needed.

References

1. D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. In *Proc. of 7th Annu. ACM Sympos. Comput. Geom.*, pages 98–104, 1991.
2. J. A. Bondy and U. S. R. Murty. *Graph theory with applications.* American Elsevier Publishing Co., Inc., New York, 1976.
3. P. Bose and P. Morin. An improved algorithm for subdivision traversal without extra storage. *Internat. J. Comput. Geom. Appl.*, 12(4):297–308, 2002. Annual International Symposium on Algorithms and Computation (Taipei, 2000).
4. P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7:609–616, 2001.
5. E. Chavez, S. Dobrev, E. Kranakis, J. Opatrny, L. Stacho, and J. Urrutia. Traversal of a quasi-planar subdivision without using mark bits. accepted for 4th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN'04), Fanta Fe, New Mexico, 2004.

6. J. Czyczowicz, E. Kranakis, N. Santoro, and J. Urrutia. Traversal of geometric planar networks using a mobile agent with constant memory. in preparation.
7. C. Gold, U. Maydell, and J. Ramsden. Automated contour mapping using triangular element data structures and an interpolant over each irregular triangular domain. *Computer Graphic*, 11(2):170–175, 1977.
8. E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proc. of 11th Canadian Conference on Computational Geometry*, pages 51–54, August 1999.
9. F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric ad-hoc routing: Of theory and practice. In *Proc. of the 22nd ACM Symposium on the Principles of Distributed Computing (PODC)*, July 2003.
10. F. Kuhn, R. Wattenhofer, and A. Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Proc. of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, June 2003.
11. D. Peuquet and D. Marble. Arc/info: an example of a contemporary geographic information system. In *Introductory Readings in Geographic Information Systems*, pages 90–99. Taylor & Francis, 1990.
12. D. Peuquet and D. Marble. Technical description of the dime system. In *Introductory Readings in Geographic Information Systems*, pages 100–111. Taylor & Francis, 1990.