Compass Routing on Geometric Networks

Evangelos Kranakis, School of Computer Science,
Carleton University, Ottawa, Canada K1S 5B6,
Harvinder Singh, and Jorge Urrutia,
School of Information Technology and Engineering,
University of Ottawa, Ottawa, Canada K1N 6N5

1 Introduction.

Suppose that a traveler arrives to the City of Toronto, and wants to walk to the famous CN-Tower, one of the tallest free-standing structures in the world. Assume now that our visitor, lacking a map of Toronto, is standing at a crossing from which he can see the CN-tower, and several streets $S_1, \ldots, S_m$ that he can choose to start his walk. A natural (and most likely safe assumption), is that our visitor must choose to walk first along the road that points closest in the direction of the CN-tower, see Figure 1.

A close look at maps of numerous cities around the world, show us that the previous way to explore a new, and unknown city will in general yield walks that will be close enough to the optimal ones to travel from one location to another.

In mathematical terms, we can model the map of many cities by geometric graphs in which street intersections are represented by the vertices of our graphs, and streets by straight line segments. Compass routing on geometric networks, in its most elemental form yields the following algorithm:

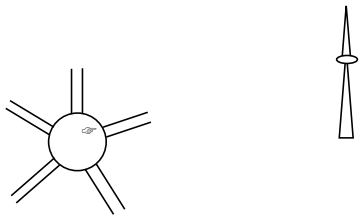

Figure 1: Finding our way to the CN-tower.

Compass Routing *Suppose that we want to travel from an initial vertex s to a destination vertex t, and that all the information available to us at any point in time is the coordinates of our destination, our current position, and the directions of the edges incident with the vertex we are located at. Starting at s, we will in a recursive way choose and traverse the edge of our geometric graph incident to our current position and with the closest slope to that of the line segment connecting the vertex we are standing at to t. Ties are broken randomly.*

For the graph shown in Figure 2, compass routing will produce the path s, a, b, c, t from s to t.

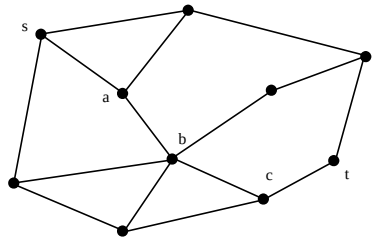

Figure 2: Traveling from s to t using compass routing.

In this paper we study *local routing algorithms* on geometric networks. Formally speaking, suppose that we want to travel from a vertex s to a vertex t of a geometric network. A routing algorithm is called a *local routing algorithm* if it satisfies the following conditions:

1. At each point in time, we know the coordinates of our starting position, as well as those of our destination. In addition, we have at our disposal a finite amount of storage where we can keep a constant number of identifiers of vertices of our network. Notice that this implies that at no point in time do we have full knowledge of the topology of our entire network.

2. Upon arrival to a vertex v (starting at s), we can use local information stored in v regarding its neighbors, and the edges connecting v to them.

Using this information plus that stored in our local memory, we choose an edge incident to v, and traverse it until we reach its second end vertex, unless that is $v = t$, in which case we stop.

3. We are not allowed to change the local information stored at v. Notice that in particular, once we have left a vertex, if we return to it, we will not be able to determine that we have visited it, unless its identifier is one of the ones we carry in our local memory. Recall that we can remember only a constant number of them.

The motivation for the last condition imposed on our algorithms, is that we don't want to leave markers, or garbage at the vertices we have already visited while trying to reach our destination. This condition arises naturally while sending information between different nodes of a network. For example if a server is connected to the web, we would like to avoid keeping track of those messages that passed through our server, this would easily use an enormous amount of memory at heavily used nodes that could quickly overload the memory available at those sites.

The problem we study here, has some similarity with problems that explore graphs, except that in most of them, i.e. finding our way out of a maze, we leave markers indicating sections of our network we have already visited. In this respect, our algorithms are *ecologically sound* in the sense that we don't leave a trail of garbage during our walks. It is interesting to point out now, that our assumption on having geometric networks is essential to our study, as it is straightforward to prove that for arbitrary networks in which all we have at each node is the list of the vertices adjacent to it, no local *deterministic* routing algorithms exist.

An approach to obtain local routing algorithms has been studied for distributed networks for which *compact routing* algorithms such as interval routing [10], boolean routing [9] etc. have been developed. In these models, we store at each node of a network a copy of a distributed algorithm. Such schemes, however can be worst-case storage intensive in the sense that large amounts of information may be required to store per node in order to achieve all-pair shortest path routing, see [3, 8]. From our point of view, another perhaps more serious drawback of the previous approach, is that the topology of the networks for which these algorithms have been developed, are assumed to be of a specific type, e.g. Cayley graphs. Our goal here is that of developing routing algorithms that can be applied to existing communication networks whose only restriction is that they are planar networks. It is interesting to note that, some of the best network topology maps used by Internet Service Providers and Internet Backbone Networks, such as TEN-34, EuropaNET, Eunet, Qwest Nationwide Network, etc. can be modeled as planar or almost planar graphs; see [1].

Finally, we mention that routing algorithms based in the location of our destination have been studied within the framework of wireless communication networks, i.e. networks in which processors represent devices similar to radio stations, two of which can communicate if they are sufficiently close to each other, see [2, 7, 11]. In these problems, as in ours, the goal is not necessarily that of finding the shortest path connecting two vertices of our network, but to make sure that our information reaches its destination.

2 Compass Routing

It is not true however that compass routing will always find a path from any starting point in a geometric graph to any other. Not even in cases when our geometric graphs are triangulations. The reader can verify that in the geometric graph shown in Figure 3 when we try to go from $s = u_0$ to t using compass routing, we travel around and around the cycle with vertex set $\{v_0, w_i; i = 0, \ldots, 3\}$.

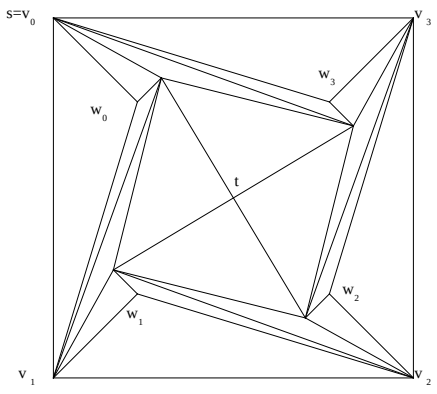

Figure 3: Compass routing will not reach t from u_i, $i = 0, \ldots, 3$.

We say that a geometric graph G supports compass routing if for every pair of its vertices s and t, compass routing (starting at s) produces a path from s to t.

The *Delaunay triangulation* $\mathcal{D}(P_n)$ of a set of n points P_n on the plane, is the partitioning of the convex hull of P_n into a set of triangles with disjoint interiors such that

- the vertices of these triangles are points in P_n
- for each triangle in our triangulation the circle passing through its vertices contains no other point of P_n in its interior.

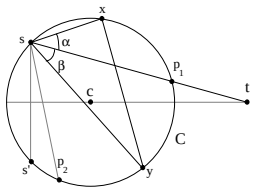

Figure 4: Routing on Delaunay triangulations.

It is well known that when the elements of P_n are in general *circular position*, i.e. no four of then are cocircular, then $\mathcal{D}(P_n)$ is well defined. For the rest of this section we will assume that P_n is in general circular position. This condition can be easily removed leaving our results unchanged. We now prove:

Theorem 2.1 *Let P_n be a set of n points on the plane, then $\mathcal{D}(P_n)$ supports compass routing.*

Proof: Suppose that we want to go from vertex s to vertex t in $\mathcal{D}(P_n)$. We will now show that if compass routing chooses to traverse edge sv of $\mathcal{D}(P_n)$ then the distance from v to t is strictly smaller than the distance from s to t. Since $\mathcal{D}(P_n)$ has a finite number of vertices, this is enough to prove that we will eventually reach t. Let $s-t$ be the line segment joining s to t, and suppose that it intersects the triangle with vertices $\{s, x, y\}$. Let C be the circle through $\{s, x, y\}$. Let c be the center of C, and s' the mirror image of s with respect to the line joining c to t, see Figure 4. Let α and β be the angles formed between $x-s$ and $s-t$, and between $t-s$ and $s-y$ respectively. Two cases arise:

1. $\alpha < \beta$. In this case, compass routing will choose edge sx, and it is straightforward to see that the distance between x and t is indeed smaller than that between s and t.
2. $\beta \leq \alpha$. Let P_1 be the intersection point between the open line segment joining s to t and C, and P_2 be the point on C such that the distance from P_1 to P_2 is the same as the distance between s and P_1. It is easy to see now that P_2 lies on the open arc C' joining s to s' in the clockwise direction. However since $\beta \leq \alpha$ y must lie on C', and thus its distance to t is smaller than the distance from s to t

■

3 Compass Routing II

We now obtain a local information routing algorithm that guarantees that any message will eventually reach its destination. We describe our algorithm first for the case in which our geometric graphs are convexly embedded, i.e. all the faces of our geometric graph are convex, except for the unbounded one which is assumed to be the complement of a convex polygon, see Figure 5. Our algorithm proceeds as follows:

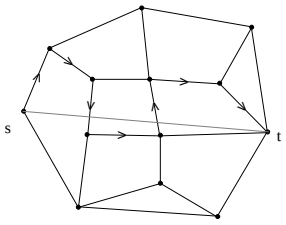

Figure 5: Routing in convexly embedded geometric graphs.

Compass Routing II:

1. Starting at s determine the face $F = F_0$ incident to s intersected by the line segment st joining s to t. Pick any of the two edges of F_0 incident to s, and start traversing the edges of S_0 until we find the second edge, say $u-v$ on the boundary of F_0 intersected by st.
2. At this point, we update F to be the second face of our geometric graph containing $u-v$ on its boundary. We now traverse the edges of our new F until we find a second edge $x-y$ intersected by st. At this point we update F again as in the previous point. We iterate our current step until we reach t

Let $F_0, F_1, \ldots \mathrm{F}_r$ be the faces intersected by st. Observe that initially $F = F_0$, and that each time we update F, we change its value from F_i to F_{i+1}, so eventually we will reach F_k, the face containing t, and when we traverse its boundary we will arrive at t.

We now show how to modify our algorithm so that it will also work for arbitrary geometric graphs. Observe first that the vertices and edges of any geometric graph G induce a partitioning of the plane into a set of connected regions with disjoint interiors, not necessarily convex, called the faces of G. The boundary $\mathcal{B}_i$ of each of these faces, is a closed polygonal in which we admit some edge of G

to appear twice. For example in the graph shown in Figure 6, the polygonal bounding the external face is $\{v_1, v_2, v_3, v_4, v_5, v_6, v_5, v_7, v_8, v_9, v_{10}, v_4, v_3, v_{11}, v_1\}$, notice that edge v_3v_4 is traversed twice, once from v_3 to v_4, and again from v_4 to v_3. Suppose now that we want to travel from a vertex s to a vertex t of G. As before calculate the line segment joining s to t, and determine the face $F = F_0$ incident to s intersected by st. We now traverse the polygonal determined by F_0, checking if the last line traversed intersects st. If it does, we calculate the distance from the intersection point to s. Upon returning to s, (unless we reach t in which case we stop) all we need to recall is the point p_0 at which the polygonal bounding F_0 intersects st, which maximizes its distance to s. We now travel the boundary of F_0 until we reach p_0, at which point we update F to be the second face whose face contains p_0. It is straightforward to see that we eventually reach t. Furthermore, we notice that each edge $e$of G is traversed at most twice, regardless of whether e belongs to the polygonals determined by one or two faces of G.

For the graph shown in Figure 6, we first traverse the edges of the face bounded the polygonal $v_1v_2v_3v_{11}v_1$, we then traverse the edges of the external face, and finally the polygonal $v_4v_5v_7v_8v_4v_{12}$ at which point we stop. Summarizing we have:

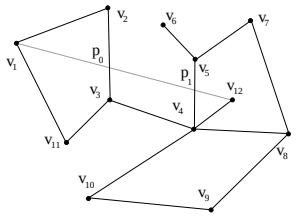

Figure 6: Traveling from v_1 to v_{12}.

Theorem 3.1 *There exists a local information routing algorithm on geometric graphs which guarantees that we reach our destination. Moreover, our algorithm is such that we traverse a linear number of edges.*

4 Further work

The problem of deciding which planar graphs admit geometric embeddings for which compass routing produces the shortest path between any pair of vertices is interesting. In this direction we have proved that trees always have such embeddings, and that some outerplanar graphs do not. The more general question of developing shortest path local routing algorithms for specific subfamilies of planar graphs remains open.

References

[1] An Atlas of Cyberspaces, *http : //www.geog. ucl.ac.uk/casa/martin/atlas/isp_maps.html.*

[2] Basagni, S., I. Chlamatac, V.R. Syrotiuk, and B.A. Woodward, "A distance Routing Effect Algorithm for Mobility", Proc. MOBICOM, 1998, 76-84.

[3] P. Fraigniau and C, Gavoille, "Universal Routing Schemes", *Journal of Distributed Computing*, **10** (1997), pp. 65-78.

[4] G. N. Frederickson and R. Janardan, "Designing Networks with Compact Routing Tables", *Algorithmica*, **3** (1988), pp. 171-190.

[5] G. N. Frederickson and R. Janardan, "Efficient Message Routing in Planar Networks", *SIAM Journal of Computing*, **18** (1989), 843-857.

[6] C, Gavoille and S. Perennes, "Lower Bounds for Shortest Path Interval Routing", Proceedings of SIROCCO96, pp. 88-103, N. Santoro and P. Spirakis, eds, Carleton University Press, 1997.

[7] Ko, Y.B., and N.H. Vaidya, "Location-aided Routing in Mobile ad hoc Networks", Proc. MOBICOM, 1998, 66-75.

[8] E. Kranakis and D. Krizanc, Proceedings of STACS96, pp. 529-540, C. Puech and R. Reischuk, eds, SVLNCS vol. 1046, 1996.

[9] E. Kranakis and D. Krizanc,"Boolean Routing on Cayley Networks", Proceedings of SIROCCO96, 119-124, N. Santoro and P. Spirakis, eds, Carleton University Press, 1997.

[10] Santoro, N., and R. Khatib, "Labeling and implicit routing in networks" *The Computer Journal*, **28**, 1 (1985), 5-8.

[11] Stojmenovic, I., and X. Liu, "Geographic Distance Routing in ad hoc Wireless Networks", Preprint, SITE, University of Ottawa, 1999.