ONLINE ROUTING IN TRIANGULATIONS*

PROSENJIT BOSE† AND PAT MORIN†

Abstract. We consider online routing algorithms for routing between the vertices of embedded planar straight line graphs. Our results include (1) two deterministic memoryless routing algorithms, one that works for all Delaunay triangulations and the other that works for all regular triangulations, (2) a randomized memoryless algorithm that works for all triangulations, (3) an $O(1)$ memory algorithm that works for all convex subdivisions, (4) an $O(1)$ memory algorithm that approximates the shortest path in Delaunay triangulations, and (5) theoretical and experimental results on the competitiveness of these algorithms.

Key words. Routing, Online algorithms, Delaunay triangulations, shortest path, spanning path

1. Introduction. Path finding, or routing, is central to a number of fields including geographic information systems, urban planning, robotics, and communication networks. In many cases, knowledge about the environment in which routing takes place is not available beforehand, and the vehicle/robot/packet must learn this information through exploration. Algorithms for routing in these types of environments are referred to as *online* [3] routing algorithms.

In this paper we consider online routing in the following abstract setting: The environment is a planar straight line graph [17], T, with n vertices, whose edges are weighted by the Euclidean distance between their endpoints, the source v_{src} and destination v_{dst} are vertices of T, and a packet can only move on edges of T. Initially, a packet only knows v_{src}, v_{dst}, and $N(v_{\mathrm{src}})$, where $N(v)$ denotes the set of vertices adjacent to v.

We classify online routing algorithms based on their use of memory and/or randomization. Define v_{cur} as the vertex at which the packet is currently stored. A routing algorithm is called *memoryless* if the next step taken by a packet depends only on v_{cur}, v_{dst}, and $N(v_{\mathrm{cur}})$. An algorithm is *randomized* if the next step taken by a packet is chosen randomly from $N(v_{\mathrm{cur}})$. A randomized algorithm is memoryless if the distribution used to choose from $N(v_{\mathrm{cur}})$ is a function only of v_{cur}, v_{dst}, and $N(v_{\mathrm{cur}})$.

The justification for studying the memory requirements of routing algorithms comes from communication networks, in which memory used by an algorithm results in header information that travels with a packet. Since this information is only used for routing purposes and is of no use to the sender or receiver, it effectively results produces a decrease in communication bandwidth.

For an algorithm $\mathcal{A}$ we say that a graph *defeats* $\mathcal{A}$ if there is a source/destination pair such that a packet never reaches the destination when beginning at the source. If $\mathcal{A}$ finds a path P from v_{src} to v_{dst} we call P the $\mathcal{A}$ path from v_{src} to v_{dst}. Here we use the term *path* in an intuitive sense rather than a strict graph theoretic sense, since P may visit the same vertex more than once.

In this paper we also consider, as a special case, a class of "well-behaved" triangulations. The *Voronoi diagram* [16] of S is a partitioning of space into cells such that all points within a Voronoi cell are closer to the same element $p \in S$ than any other point in S. The *Delaunay triangulation* is the straight-line face dual of the

*This research was supported by the Natural Sciences and Engineering Research Council of Canada.

†School of Computer Science, Carleton University, 1125 Colonel By Dr., Ottawa, Canada, K1S 5B6 ({jit,morin}@scs.carleton.ca)

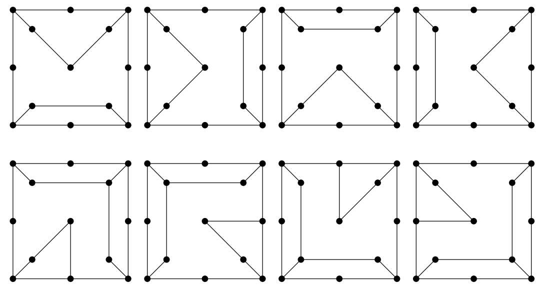


FIG. 2.1. *Any deterministic memoryless algorithm is defeated by at least one of these graphs.*

Voronoi diagram, i.e., two points in S have an edge between them in the Delaunay triangulation if their Voronoi regions have an edge in common.

In this paper we consider several different routing algorithms and compare their performance empirically. In particular, we describe

1. a memoryless algorithm that is not defeated by any Delaunay triangulation,
2. a memoryless algorithm that is not defeated by any regular triangulation,
3. a memoryless randomized algorithm that uses 1 random bit per step and is not defeated by any triangulation,
4. an algorithm that uses $O(1)$ memory that is not defeated by any convex subdivision,
5. an algorithm for Delaunay triangulations that uses $O(1)$ memory in which a packet never travels more than a constant times the Euclidean distance between $v_{\rm src}$ and $v_{\rm dst}$, and
6. a theoretical and empirical study of the quality (length) of the paths found by these algorithms.

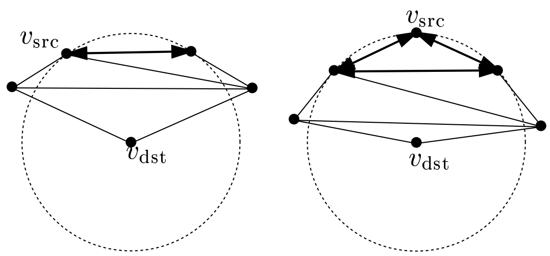
The first four routing algorithms are described in Section 2. Section 3 presents theoretical and empirical results on the length of the paths found by these algorithms and describes our algorithm for Delaunay triangulations. A discussion of related work is provided in Section 4. Finally, Section 5 summarizes and describes directions for future research.

2. Four Simple Algorithms. In this section we describe four online routing algorithms and prove theorems about which types of graphs never defeat them. We begin with the simplest (memoryless) algorithms and proceed to the more complex algorithms.

However, before beginning we should note that deterministic memoryless algorithms have some inherent limitations. This can be seen by considering the 8 graphs in Fig. 2.1 with the center vertex as the destination. Since the corresponding corner vertices of all graphs look the same, a deterministic memoryless algorithm must make the same decision at the corner vertices of each graph. But for any possible combination of decisions, at least one of the graphs will defeat the algorithm.

Thus, there is no hope of finding a deterministic memoryless algorithm that works for all embedded planar graphs (or even all 2-connected planar graphs).

2.1. Greedy Routing. The *greedy routing* (GR) algorithm always moves the packet to the neighbor $gdy(v_{\rm cur})$ of $v_{\rm cur}$ that minimizes $dist(gdy(v_{\rm cur}), v_{\rm dst})$, where $dist(p,q)$ denotes the Euclidean distance between p and q. In the case of ties, one of the vertices is chosen arbitrarily. The greedy routing algorithm can be defeated by a triangulation T in two ways: (1) the packet can get trapped moving back and forth



(a) (b)

FIG. 2.2. *Triangulations that defeat the greedy routing algorithm.*

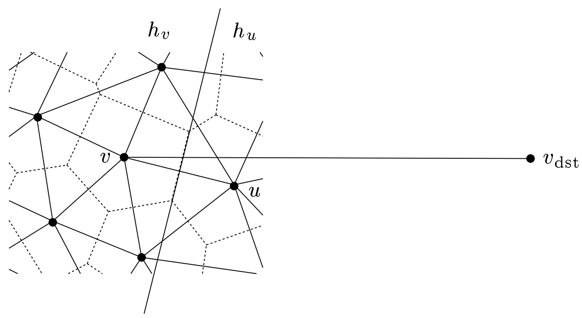


FIG. 2.3. *The proof of Theorem 2.1.*

on an edge of the triangulation (Fig. 2.2.a), or (2) the packet can get trapped on a cycle of three or more vertices (Fig. 2.2.b). However, as the following theorem shows, neither of these situations can occur if T is a Delaunay triangulation.

THEOREM 2.1. *There is no point set whose Delaunay triangulation defeats the greedy routing algorithm.*

Proof. We proceed by showing that every vertex v of T has a neighbor that is strictly closer to v_{dst} than v is. Thus, at each routing step, the packet gets closer to v_{dst} and therefore, after at most n steps, reaches v_{dst}. Refer to Fig. 2.3.

Consider the Voronoi diagram [16] $VD(T)$ of the vertices of T and let e be the first edge of $VD(T)$ intersected by the directed line segment (v, v_{dst}). Note that e is on the boundary of two Voronoi cells, one for v and one for some other vertex u, and the supporting line of e partitions the plane into two open half planes $h_v = \{p : dist(p, v) < dist(p, u)\}$ and $h_u = \{p : dist(p, u) < dist(p, v)\}$. Since the Voronoi diagram is the straight line face dual of the Delaunay triangulation, the edge $(u, v) \in T$. Also, by the choice of e, $v_{\mathrm{dst}} \in h_u$, i.e., $dist(u, v_{\mathrm{dst}}) < dist(v, v_{\mathrm{dst}})$. ∎

2.2. Compass Routing. The *compass routing* (CR) algorithm always moves the packet to the vertex $cmp(v_{\mathrm{cur}})$ that minimizes the angle $\angle v_{\mathrm{dst}}, v_{\mathrm{cur}}, cmp(v_{\mathrm{cur}})$ over all vertices adjacent to v_{cur}. Here the angle is taken to be the smaller of the two angles as measured in the clockwise and counterclockwise directions. In the case of ties, one of the (at most 2) vertices is chosen using some arbitrary deterministic rule.

One might initially believe (as we did) that compass routing can always be used to find a path between any two vertices in a triangulation. However, the triangulation in Fig. 2.4 defeats compass routing. When starting from one of the vertices on the outer face of T, and routing to v_{dst}, the compass routing algorithm gets trapped on

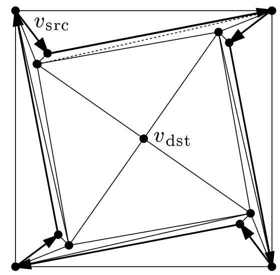


FIG. 2.4. *A triangulation that defeats the compass routing algorithm.*

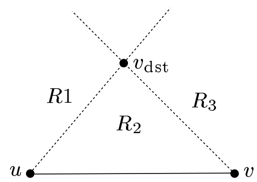


FIG. 2.5. *The proof of Lemma 2.2.*

the cycle shown in bold. The following lemma shows that any triangulation that defeats compass routing causes the packet to get trapped in a cycle.

LEMMA 2.2. *Let T be a triangulation that defeats compass routing, and let $v_{\rm dst}$ be a vertex such that compass routing fails to route a packet to $v_{\rm dst}$ when given some other vertex as the source. Then there exists a cycle $C = v_0, \ldots, v_{k-1}$ ($k \geq 3$) in T such that $cmp(v_i) = v_{i+1}$ for all $0 \leq i < k$.*[1]

Proof. Since T defeats compass routing, and the compass routing algorithm makes the same decision each time it visits a vertex, either there is an edge (u, v) such that $cmp(u) = v$ and $cmp(v) = u$, or there is the situation described in the lemma. We prove that there can be no such edge (u, v). Suppose such an edge (u, v) does exist. Then there is a triangle (u, v, w) in T such that w is in the same half-plane bounded by the line through u and v as $v_{\rm dst}$. Referring to Fig. 2.5, the vertex w must be in the one of the regions 1, 2, or 3. But this is a contradiction, since if w is in region 1, then $cmp(v) = w$, if w is in region 2, then $cmp(u) = w$ (and $cmp(v) = w$), and if w is in region 3, then $cmp(u) = w$. □

We call such a cycle, C, a *trapping cycle* in T for $v_{\rm dst}$. Next we characterize trapping cycles in terms of a visibility property of triangulations. Let t_1 and t_2 be two triangles in T. Then we say that t_1 *obscures* t_2 if there exists a ray originating at $v_{\rm dst}$ that strikes t_1 first and then t_2. Let u and v be any two vertices of T such that $cmp(u) = v$. Then define $\triangle uv$ as the triangle of T that is contained in the closed half-plane bounded by the line through uv and that contains $v_{\rm dst}$. We obtain the following useful characterization of trapping cycles.

LEMMA 2.3. *Let T be a triangulation that defeats compass routing and let $C = v_0, \ldots, v_{k-1}$ be a trapping cycle in T for vertex $v_{\rm dst}$. Then $\triangle v_i v_{i+1}$ is either identical to, or obscures $\triangle v_{i-1} v_i$, for all $0 \leq i < k$.*

Proof. Refer to Fig. 2.6. Assume that $\triangle v_i v_{i+1}$ and $\triangle v_{i-1} v_i$ are not identical, otherwise the lemma is trivially true. Let w be the third vertex of $\triangle v_i v_{i+1}$. Then w cannot lie in the cone defined by $v_{\rm dst}$, v_i and v_{i+1}, otherwise we would have $cmp(v_i) =$

[1]Here and henceforth, all subscripts are assumed to be taken mod k.

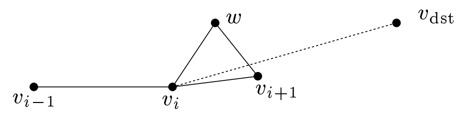


FIG. 2.6. *The proof of Lemma 2.3.*

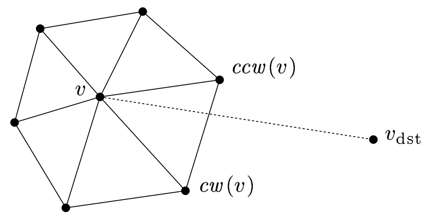


FIG. 2.7. *Definition of $cw(v)$ and $ccw(v)$.*

w. But then the line segment joining w and v_{i+1} obscures v_i and hence $\triangle v_i v_{i+1}$ obscures $\triangle v_{i-1} v_i$. ∎

A *regular triangulation* [18] is a triangulation obtained by orthogonal projection of the faces of the lower hull of a 3-dimensional polytope onto the plane. Note that the Delaunay triangulation is a special case of a regular triangulation in which the vertices of the polytope all lie on a paraboloid. Edelsbrunner [8] showed that if T is a regular triangulation, then T has no set of triangles that obscure each other cyclically from *any* viewpoint. This result, combined with Lemma 2.3, yields our main result on compass routing.

THEOREM 2.4. *There is no regular triangulation that defeats the compass routing algorithm.*

2.3. Randomized Compass Routing. In this section, we consider a randomized routing algorithm that is not defeated by any triangulation. Let $cw(v)$ be the vertex in $N(v)$ that minimizes the clockwise angle $\angle v_{\rm dst}, v, cw(v)$ and let $ccw(v)$ be the vertex in $N(v)$ that minimizes the counterclockwise angle $\angle v_{\rm dst}, v, ccw(v)$ (See Fig. 2.7). Then the *randomized compass routing* (RCR) algorithm moves the packet to one of $\{cw(v_{\rm cur}), ccw(v_{\rm cur})\}$ with equal probability.

Before we can make statements about which triangulations defeat randomized compass routing, we must define what it means for a triangulation to defeat a randomized algorithm. We say that a triangulation T defeats a (randomized) routing algorithm if there exists a pair of vertices $v_{\rm src}$ and $v_{\rm dst}$ of T such that a packet originating at $v_{\rm src}$ with destination $v_{\rm dst}$ has probability 0 of reaching $v_{\rm dst}$ in any finite number of steps.

Note that, since randomized compass routing is memoryless, proving that a triangulation T does not defeat randomized compass routing implies that a packet reaches its destination with probability 1. The following theorem shows the versatility of randomized compass routing.

THEOREM 2.5. *There is no triangulation that defeats the randomized compass routing algorithm.*

Proof. Assume, by way of contradiction that a triangulation T exists that defeats the randomized compass routing algorithm. Then there is a vertex $v_{\rm dst}$ of T and a minimal set S of vertices such that: (1) $v_{\rm dst} \notin S$, (2) the subgraph H of T induced by S is connected, and (3) for every $v \in S$, $cw(v) \in S$ and $ccw(v) \in S$.

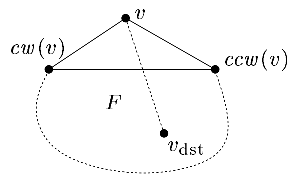


FIG. 2.8. *The proof of Theorem 2.5.*

Refer to Fig. 2.8 for what follows. The vertex v_{dst} lies in some face F of H. Let v be a vertex on the boundary of F such that the line segment (v, v_{dst}) is contained in F. Such a vertex is guaranteed to exist [5]. The two neighbours of v on the boundary of F must be $cw(v)$ and $ccw(v)$ and these cannot be the same vertex (since F contains (v, v_{dst}) in its interior). Note that, by the definition of $cw(v)$ and $ccw(v)$, and by the fact that T is a triangulation, the triangle $(cw(v), v, ccw(v))$ is in T. But this is a contradiction, since then v is not on the boundary of F. □

2.4. Right-Hand Routing. The folklore "right-hand rule" for exploring a maze states that if a player in a maze walks around never lifting her right-hand from the wall, then she will eventually visit every wall in the maze. More specifically, if the maze is the face of a connected planar straight line graph, the player will visit every edge and vertex of the face [2].

Let T be any convex subdivision. Consider the planar subdivision T' obtained by deleting from T all edges that properly intersect the line segment joining v_{src} and v_{dst}. Because of convexity, T' is connected, and v_{src} and v_{dst} are on the boundary of the same face F of T'. The *right-hand routing* (RHR) algorithm uses the right-hand rule on the face F to route from v_{src} to v_{dst}. Right-hand routing is easily implemented using only $O(1)$ additional memory by remembering v_{src}, v_{dst}, and the last vertex visited.

THEOREM 2.6. *There is no convex subdivision that defeats the right-hand routing algorithm.*

3. Competitiveness of Paths. Thus far we have considered only the question of whether routing algorithms can find *a* path between any two vertices in T. An obvious direction for research is to consider the length of the path found by a routing algorithm. We say that a routing algorithm $\mathcal{A}$ is c-competitive for T, if for any pair $(v_{\mathrm{src}}, v_{\mathrm{dst}})$ in T, the length (sum of the edge lengths) of the path between v_{src} and v_{dst} found by $\mathcal{A}$ is at most c times the length of the shortest path between v_{src} and v_{dst} in T. In the case of randomized algorithms, we use the expected length of the path. We say that $\mathcal{A}$ has a competitive ratio of c if it is c-competitive.

This section addresses questions about the competitive ratio of the algorithms described so far, as well as a new algorithm specifically targeted for Delaunay triangulations. We present theoretical as well as experimental results.

3.1. Negative Results. It is not difficult to contrive triangulations for which none of our algorithms are c-competitive for any constant c. Thus it is natural to restrict our attention to a well behaved class of triangulations. Unfortunately, even for Delaunay triangulations none of the algorithms described so far are c-competitive.

THEOREM 3.1. *There exists Delaunay triangulations for which none of the greedy, compass, randomized compass, or right-hand routing algorithms are c-competitive for any constant c.*

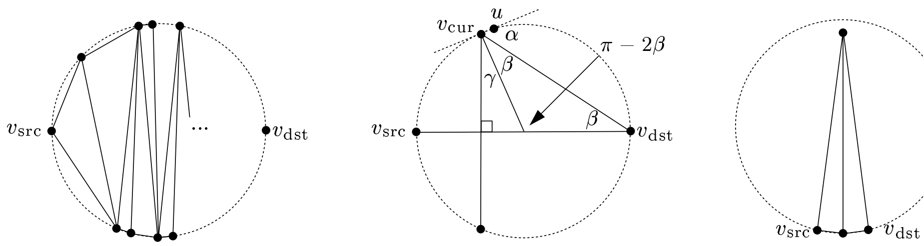


FIG. 3.1. *The proof of Theorem 3.1.*

Proof. We begin with greedy routing. Consider the set of points that are placed on a circle and then triangulated to obtain the zig-zag triangulation T shown in Fig. 3.1.a. Since the points are cocircular, this is a valid Delaunay triangulation. The points are placed so that each vertex v has a neighbor on the opposite side of the line through $v_{\rm src}$ and $v_{\rm dst}$ that is closer to $v_{\rm dst}$ than v's two neighbors on the same side of the line.

Note that there exists a path between $v_{\rm src}$ and $v_{\rm dst}$ of length approximately $(\pi/2) \cdot dist(v_{\rm src}, v_{\rm dst})$, and this is therefore an upper bound on the length of the shortest path between $v_{\rm src}$ and $v_{\rm dst}$. The length of the "zig-zag" path that uses the diagonals of T between $v_{\rm src}$ and $v_{\rm dst}$ is $\Theta(n) \cdot dist(v_{\rm src}, v_{\rm dst})$, and this is the path taken by the greedy routing algorithm. Thus, greedy routing is not c-competitive for this triangulation.

To show that compass routing is not c-competitive, we again consider a set of cocircular points and make a zig-zag triangulation. Let $v_{\rm cur}$ be any point on the circle with diameter $v_{\rm src}, v_{\rm dst}$. Consider the angle α between the tangent line passing through $v_{\rm cur}$ and the line through $v_{\rm src}$ and $v_{\rm dst}$. Compare this with the angle between the line perpendicular to $v_{\rm src}$ and $v_{\rm dst}$ that passes through $v_{\rm cur}$ and the line through $v_{\rm src}$ and $v_{\rm dst}$. Referring to Fig. 3.1.b, we have

$$\alpha = \pi/2 - \beta \tag{3.1}$$

$$\gamma = \pi/2 - 2\beta \ , \tag{3.2}$$

and therefore $\gamma + \beta = \pi/2 - \beta = \alpha$, i.e., the two angles are equal. Thus if compass routing were to choose between the tangent line and the line crossing the circle it would be a tie. Now, by placing a point u on the circle close to $v_{\rm cur}$ we can make $\angle u, v_{\rm cur}, v_{\rm dst} = \alpha - \epsilon$ for arbitrarily small $\epsilon > 0$. Similarly, by placing a point $v_{\rm nxt}$ on the opposite side of the circle we can make $\angle v_{\rm nxt}, v_{\rm cur}, v_{\rm dst} = \alpha - \epsilon - \delta$ for arbitrarily small $\delta > 0$, so that $cmp(v_{\rm cur}) = v_{\rm nxt}$. Since ϵ and δ can be arbitrarily small, we can repeat this construction as often as we like, thereby making the compass routing path arbitrarily long.

To see that randomized compass routing and right-hand routing are not c-competitive consider a configuration of points like that in Fig. 3.1.c. By making $v_{\rm src}$ and $v_{\rm dst}$ almost collinear with a third point, it is possible to produce arbitrarily long thin triangles that make the length of the path found by right-hand routing arbitrarily long. Furthermore, in this configuration the probability that the randomized compass routing path is the same as the right-hand path is $1/2$, and thus the expected length of the randomized compass path can be arbitrarily large. ∎

3.2. A c-Competitive Algorithm for Delaunay Triangulations. Since none of the algorithms described in Section 2 is competitive, even for Delaunay triangula-

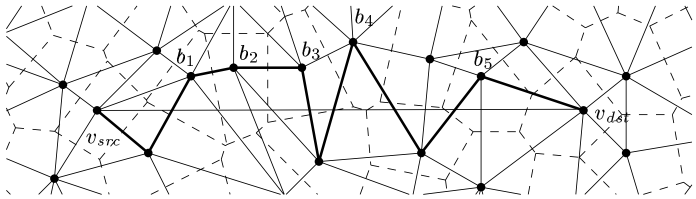


FIG. 3.2. *A path obtained by the Voronoi routing algorithm.*

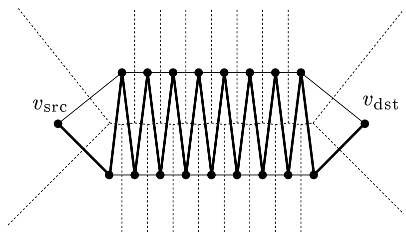


FIG. 3.3. *The Voronoi routing algorithm is not c-competitive for all Delaunay triangulations.*

tions, an obvious question is whether there exists any algorithm that is competitive for Delaunay triangulations. In this section we answer this question in the affirmative. In fact, we prove an even stronger result by giving an algorithm that finds a path whose cost is at most a constant times $dist(v_{\rm src}, v_{\rm dst})$.

Our algorithm is based on the remarkable proof of Dobkin *et al* [7] that the Delaunay triangulation approximates the complete Euclidean graph to within a constant factor in terms of shortest path length. In the following we will use the notation $\mathrm{x}(p)$ (resp. $\mathrm{y}(p)$) to denote the x-coordinate (resp. y-coordinate) of the point p, and the notation $|X|$ to denote the Euclidean length of the path X.

Consider the directed line segment from $v_{\rm src}$ to $v_{\rm dst}$. This segment intersects regions of the Voronoi diagram in some order, say $R_0, \ldots, R_{m-1}$, where R_0 is the Voronoi region of $v_{\rm src}$ and R_{m-1} is the Voronoi region of $v_{\rm dst}$. The *Voronoi routing* (VR) algorithm for Delaunay triangulations moves the packet from $v_{\rm src}$ to $v_{\rm dst}$ along the path $v_0, \ldots, v_{m-1}$ where v_i is the site defining R_i. An example of a path obtained by the Voronoi routing algorithm is shown in Fig. 3.2. Since the Voronoi region of a vertex v can be computed given only the neighbours of v in the Delaunay triangulation, it follows that the Voronoi routing algorithm is an $O(1)$ memory routing algorithm.

The Voronoi routing algorithm on its own is not c-competitive for all Delaunay triangulations, as can be seen from Fig. 3.3.

However, it does have some properties that allow us to derive a c-competitive algorithm. As with right-hand routing, let T' be the graph obtained from T by removing all edges of T that properly intersect the segment $(v_{\rm src}, v_{\rm dst})$, and let F be the face of T' that contains both $v_{\rm src}$ and $v_{\rm dst}$. Assume wlog that $v_{\rm src}$ and $v_{\rm dst}$ both lie on the x-axis and that $\mathrm{x}(v_{\rm src}) < \mathrm{x}(v_{\rm dst})$. The following results follow from the work of Dobkin *et al* [7].

LEMMA 3.2. *The Voronoi path is x-monotone, i.e.,* $\mathrm{x}(v_i) < \mathrm{x}(v_j)$ *for all* $i < j$.

LEMMA 3.3. *Let P' be the collection of maximal subpaths of $v_0, \ldots, v_{m-1}$ that remain above the x-axis, i.e.,* $P' = \{v_i, \ldots, v_j : \mathrm{y}(v_{i-1}) < 0 \text{ and } \mathrm{y}(v_{j+1}) < 0 \text{ and } \mathrm{y}(v_k) \geq 0 \text{ for all } i \leq k \leq j\}$. *Then* $\sum_{X \in P'} |X| \leq (\pi/2) \cdot dist(v_{\rm src}, v_{\rm dst})$.

Let $b_0, \ldots, b_{l-1}$ be the subsequence of vertices of $v_0, \ldots, v_{m-1}$ that are above or

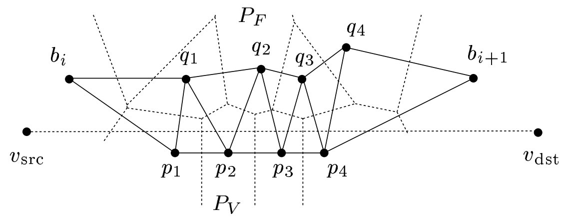


FIG. 3.4. *Definitions of P_V and P_F.*

on the segment $(v_{\rm src}, v_{\rm dst})$. (Refer to Fig. 3.2.) Consider two vertices $b_i = v_j$ and $b_{i+1} = v_k$, where $k \neq j+1$, i.e., the Voronoi path between b_i and b_{i+1} is not a direct edge. Let $P_V = (b_i = p_0, \ldots, p_x = b_{i+1})$ be the portion of the Voronoi path between b_i and b_{i+1} and let $P_F = (b_i = q_0, \ldots, q_y = b_{i+1})$ be the upper boundary of F between b_i and b_{i+1} (see Fig. 3.4). Then the following holds.

LEMMA 3.4. *Let $c_{\rm dfs} = (1+\sqrt{5})\frac{\pi}{2}$.*[2] *Then, $|P_V| \leq c_{\rm dfs} \cdot (\mathrm{x}(b_i) - \mathrm{x}(b_i))$ or $|P_F| \leq c_{\rm dfs} \cdot (\mathrm{x}(b_i) - \mathrm{x}(b_i))$.*

Proof. Let $c_0, \ldots, c_z$ be the lower convex hull of P_F, and let P_j be the Voronoi path from c_j to c_{j+1}. Dobkin *et al* prove that

$$|P_V| \leq c_{\rm dfs} \cdot (\mathrm{x}(b_{i+1} - \mathrm{x}(b_i)) \text{ or } \sum_{j=0}^{z-1} |P_j| \leq c_{\rm dfs} \cdot (\mathrm{x}(b_{i+1} - \mathrm{x}(b_i)) \ . \tag{3.3}$$

We claim that this implies Lemma 3.4 and prove this by showing that P_j visits all vertices of P_F between c_j and c_{j+1}. Thus, by the triangle inequality,

$$|P_F| \leq \sum_{j=0}^{z-1} |P_j| \ . \tag{3.4}$$

Refer to Fig. 3.5 for what follows. Assume for the sake of contradiction that there is a vertex q in P_F between c_j and c_{j+1} that is not in P_j. As part of their proof, Dobkin *et al* show that P_j remains entirely above the segment (c_j, c_{j+1}). Therefore, let Q be the polygon bounded by P_j and the segment (c_j, c_{j+1}). Since q is on P_F between c_j and c_{j+1}, it must be that q is contained Q.

Since Q is monotone in the direction from c_j to c_{j+1}, it can be partioned into trapezoids whose top sides are edges of P_j, whose bottom sides are on the line segment (c_j, c_{j+1}) and whose left and right sides are perpendicular to (c_j, c_{j+1}). Refer to Fig. 3.5.

Let a and b be the two vertices of P_j that define the trapezoid containing q. We claim that a and b cannot be consecutive on P_j because their Voronoi regions do not share an edge that intersects (c_j, c_{j+1}). We will prove this by showing that in the Voronoi diagram of q, a and b the bisector of a and b does not intersect the segment (c_j, c_{j+1}). This is sufficient, since this bisector contains the bisector of a and b in the entire Voronoi diagram.

Let C be the circle with center on (c_j, c_{j+1}) and with a and b on its boundary. If the bisector of a and b in the Voronoi diagram of q, a and b intersects the segment (c_j, c_{j+1}) then C must not contain q. However, C does contain the top, left, right, and bottom sides of the trapezoid containing q. But this can't be, since then C contains

[2] We call $c_{\rm dfs}$ the Dobkin–Friedman–Supowit constant [7].

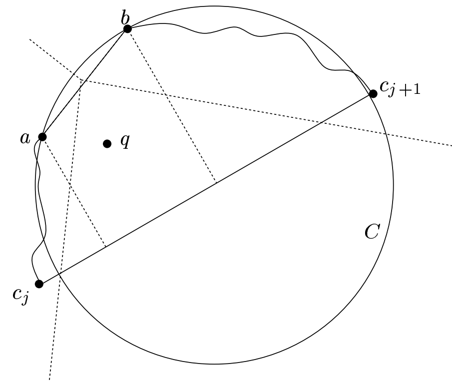


FIG. 3.5. *The proof of Lemma 3.4.*

the entire trapezoid and contains q. We conclude that there is no point q on the boundary of F between c_j and c_{j+1} that is not on P_j. ∎

Our c-competitive routing algorithm will visit all the vertices $b_0, \ldots, b_{l-1}$ in order. If b_i and b_{i+1} are consecutive on the Voronoi path (i.e., $b_i = v_j$ and $b_{i+1} = v_{j+1}$ for some j) then our algorithm will use the Voronoi path (i.e, the direct edge) from b_i to b_{i+1}. On the other hand, if b_i and b_{i+1} are not consecutive on the Voronoi path then by Lemma 3.4, there exists a path from b_i to b_{i+1} of length at most $c_{\mathrm{dfs}} \cdot (\mathrm{x}(b_{i+1}) - \mathrm{x}(b_i))$.

The difficulty occurs because the algorithm does not know beforehand which path to take. The solution is to simulate exploring both paths "in parallel" and stopping when the first one reaches b_{i+1}.[3]

More formally, let P_V and P_F be defined as in Lemma 3.4. The algorithm for finding a path from b_i to b_{i+1} is described by the following pseudocode.

1: $j \leftarrow 0$, $l_0 \leftarrow \min\{dist(p_0, p_1), dist(q_0, q_1)\}$.
2: **repeat**
3: Explore P_F until reaching b_{i+1} or until reaching a vertex q_x such that $|q_0, \ldots, q_{x+1}| > 2l_j$. If b_{i+1} is reached quit, otherwise return to b_i.
4: $j \leftarrow j + 1$, $l_j \leftarrow |q_0, \ldots, q_{y+1}|$.
5: Explore P_V until reaching b_{i+1} or until reaching a vertex p_y such that $|p_0, \ldots, p_{y+1}| > 2l_j$. If b_{i+1} is reached then quit, otherwise return to b_i.
6: $j \leftarrow j + 1$, $l_j \leftarrow |p_0, \ldots, p_{y+1}|$.
7: **until** b_{i+1} is reached

LEMMA 3.5. *Using the parallel search algorithm described above, a packet reaches* b_{i+1} *after traveling a distance of at most* $9 \cdot c_{\mathrm{dfs}} \cdot (\mathrm{x}(b_{i+1}) - \mathrm{x}(b_i)) \sim 45.75 \cdot (\mathrm{x}(b_{i+1}) - \mathrm{x}(b_i))$.

Proof. Clearly the algorithm reaches b_{i+1} in a finite number of steps, since lines 4 and 6 ensure that both paths advance by at least one edge at each iteration. Let k be the maximum value of j, and let d_j be the distance traveled during the jth exploration step of the algorithm. Thus, the total distance d traveled by the packet is given by $d = \sum_{j=0}^{k} d_j$.

Since the algorithm did not terminate with $j = k - 1$, by Lemma 3.4 we have

$$d_k < 2 \cdot c_{\mathrm{dfs}} \cdot (\mathrm{x}(b_{i+1}) - \mathrm{x}(b_i)) \ . \tag{3.5}$$

[3] A similar algorithm for finding an unknown target point on a line is given by Baeza-Yates *et al* [1]. See also Klein [12].

Similarly, since the algorithm did not terminate with $j = k-1$ or $j = k-2$ we have

$$l_{k-1} < 2 \cdot c_{\mathrm{dfs}} \cdot (\mathrm{x}(b_{i+1}) - \mathrm{x}(b_i)) \ . \tag{3.6}$$

Since $l_j \geq 2l_{j-1}$ for each $j > 0$ we have

$$d \leq \sum_{j=0}^{k-1} 2l_j + d_k \tag{3.7}$$

$$\leq \sum_{j=0}^{k-1} 2l_{k-1}/2^j + d_k \ , \tag{3.8}$$

which immediately yields a bound of $10 \cdot c_{\mathrm{dfs}} \cdot (\mathrm{x}(b_{i+1}) - \mathrm{x}(b_i))$. To obtain a tighter bound we note that $d_k > c_{\mathrm{dfs}} \cdot (\mathrm{x}(b_{i+1}) - \mathrm{x}(b_i)))$ implies $l_{k-1} < c_{\mathrm{dfs}} \cdot (\mathrm{x}(b_{i+1}) - \mathrm{x}(b_i))$. Subject to this constraint, (3.8) is maximized when $l_{k-1} = 2 \cdot c_{\mathrm{dfs}} \cdot (\mathrm{x}(b_{i+1}) - \mathrm{x}(b_i))$, yielding

$$d \leq \sum_{j=0}^{k-1} 4 \cdot c_{\mathrm{dfs}} \cdot (\mathrm{x}(b_{i+1}) - \mathrm{x}(b_i))/2^j + c_{\mathrm{dfs}} \cdot (\mathrm{x}(b_{i+1}) - \mathrm{x}(b_i)) \tag{3.9}$$

$$< 9 \cdot c_{\mathrm{dfs}} \cdot (\mathrm{x}(b_{i+1}) - \mathrm{x}(b_i)) \tag{3.10}$$

□

Given the positions of v_{dst} and v_{src} the parallel search algorithm described above is easily implemented as part of an $O(1)$ memory routing algorithm. We refer to the combination of the Voronoi routing algorithm with this parallel search algorithm as the *parallel Voronoi routing* (PVR) algorithm.

THEOREM 3.6. *The parallel Voronoi routing algorithm produces a path whose length is at most* $(9 \cdot c_{\mathrm{dfs}} + \pi/2) \cdot dist(v_{\mathrm{src}}, v_{\mathrm{dst}})$.

Proof. The algorithm incurs two costs: (1) the cost of traveling on subpaths of the Voronoi path that remain above the y-axis, and (2) the cost of applications of the parallel search algorithm. By Lemma 3.3, the first cost is at most $(\pi/2) \cdot dist(v_{\mathrm{src}}, v_{\mathrm{dst}})$. By Lemma 3.5 and the fact that $b_0, \ldots, b_{l-1}$ is x-monotone (Lemma 3.2), the cost of the second is at most $9 \cdot c_{\mathrm{dfs}} \cdot dist(v_{\mathrm{src}}, v_{\mathrm{dst}})$. □

3.3. Empirical Results. While it is sometimes possible to come up with pathological examples of triangulations for which an algorithm is not competitive, it is often more reasonable to use the competitive ratio of an algorithm on average or random inputs as an indicator of how it will perform in practice. In this section we describe some experimental results about the competitiveness of our algorithms. All experiments were performed on sets of random points uniformly distributed in the unit square, and each data point is the maximum of 50 independent trials.

The first set of experiments, shown in Fig. 3.6, involved measuring the performance of all six routing algorithms on Delaunay triangulations. Compass routing, greedy routing, and Voronoi routing consistently achieve better competitive ratios, with greedy routing slightly worse than the other two. Randomized compass routing, right-hand routing, and parallel Voronoi routing had significantly higher competitive ratios. The results for randomized compass routing and right-hand routing show a significant amount of jitter. This is due to the fact that relatively simple configurations (see Fig. 3.1.b) that can easily occur in random point sets, result in high competitive ratios for these algorithms. On the other hand, parallel Voronoi routing seems much

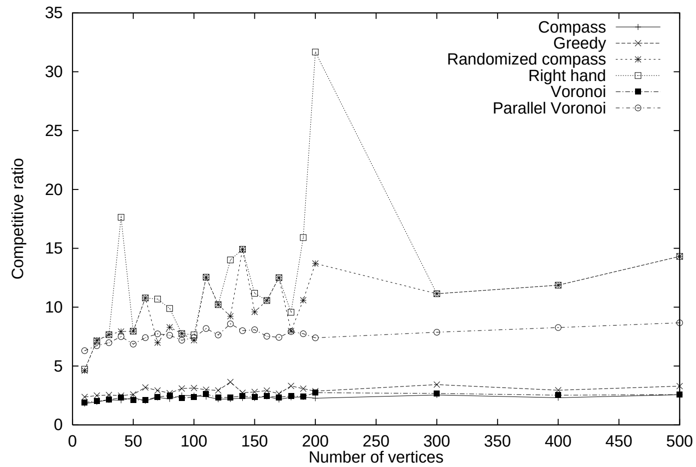


FIG. 3.6. *Empirical competitive ratios for Delaunay triangulations.*

more stable, and achieves better competitive ratios in practice than its worst case analysis would indicate.

The most important conclusion drawn from these experiments is that there are no simple configurations (i.e., that occur often in random point sets) that result in extremely high competitive ratios for greedy, compass, Voronoi, or parallel Voronoi routing in Delaunay triangulations. This suggests that any of these algorithms would work well in practice.

The four simple routing algorithms of Section 2 were also tested on Graham triangulations. These are obtained by first sorting the points by x-coordinate and then triangulating the resulting monotone chain using a linear time algorithm for computing the convex hull of a monotone polygonal chain [17]. The results are shown in Fig. 3.7. In these tests it was always the case that at least one of the 50 independent triangulations defeated greedy routing. Thus, there are no results shown for greedy routing. The relative performance of the compass, randomized compass, and right hand routing algorithms was the same as for Delaunay triangulations. However, unlike the results for Delaunay triangulations, the competitive ratio appears to be increasing linearly with the number of vertices.

4. Comparison with Related Work. In this section we survey related work in the area of geometric online routing, and compare our results with this work. We restrict our attention to work directly related to routing between the vertices of geometric graphs in which the source and destination are inputs, and do not consider routing in other geometric settings such as polygons (c.f. [9, 10, 12]).

Keil and Gutwin [11] give an algorithm for the construction of a geometric graph called the θ-graph for which a memoryless routing algorithm similar to compass routing always results in a path whose length is at most a constant (dependent only on θ) times the Euclidean distance between v_{src} and v_{dst}.

Kranakis *et al* [13] study compass routing, and provide a proof that no Delaunay triangulation defeats compass routing. The current paper makes use of a very different proof technique to show that compass routing works for a larger class of triangulations. They also describes an $O(1)$ memory routing algorithm that is not defeated by any

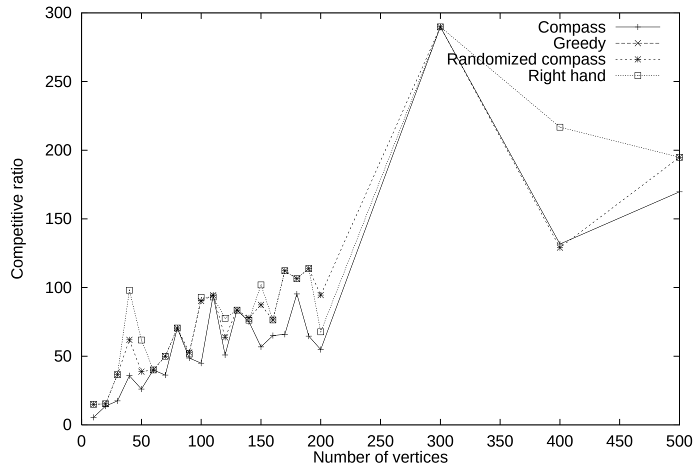


FIG. 3.7. *Empirical competitive ratios for Graham triangulations.*

connected planar graph, thus proving a stronger result than Theorem 2.6.

Lin and Stojmenović [15] and Bose *et al* [4] consider online routing in the context of *ad hoc* wireless networks modeled by unit disk graphs. They provide simulation results for a variety of algorithms that measure success rates (how often a packet never reaches its destination) as well as hop-counts of these algorithms on unit graphs of random point sets.

Lawson's oriented walk [14] is a simple algorithm for point location in Delaunay triangulations without preprocessing. The algorithm can be converted to an $O(1)$ memory routing algorithm that is not defeated by any Delaunay triangulation. The results of the current paper improve on this algorithm by providing two *memoryless* routing algorithms that are not defeated by any Delaunay triangulation.

De Berg *et al* [6] describe an algorithm for enumerating all the vertices of a connected planar subdivision using only $O(1)$ additional memory. This algorithm can also be viewed as an $O(1)$ memory routing algorithm. Similarly, in any connected graph with a finite number of vertices, a random walk will eventually visit every vertex. Thus, random walking can be viewed as a randomized memoryless routing algorithm that is not defeated by any graph. Unfortunately paths found by these techniques will usually be much longer than the shortest path, since they are general traversal techniques. In contrast, the right hand routing and randomized compass routing algorithms make use of information about the source and destination to find more direct paths.

To the best of our knowledge, no literature currently exists on the competitiveness of geometric routing algorithms in our abstract setting, and our parallel Voronoi routing algorithm is the first theoretical result in this area.

5. Conclusions. We have studied the problem of online routing in geometric graphs. Our theoretical results show which types of graphs our algorithms are guaranteed to work on, while our simulation results rank the performance of the algorithms on two types of random triangulations. These results are summarized in the Table 5.1.

We conclude with an open problem. In Section 2 we showed that no deterministic memoryless routing algorithm works for every 2-connected embedded planar graph.

Algorithm	Mem.	Rand.	Class of graphs	Rank 1	Rank 2	Competitive
GR	None	No	Delaunay $\triangle$'s	3	–	No
CR	None	No	Regular $\triangle$'s	1	1	No
RCR	None	Yes	All $\triangle$'s	5	2	No
RHR	$O(1)$	No	Convex subd.	6	3	No
VR	$O(1)$	No	Delaunay $\triangle$'s	1	–	No
PVR	$O(1)$	No	Delaunay $\triangle$'s	4	–	Yes

TABLE 5.1
Summary of Results

Can a similar argument be made for triangulations, thus proving that randomization or memory is necessary for a algorithm that is not defeated by any triangulation?

Acknowledgement. The authors would like to thank Silvia Götz for reading and commenting on an earlier version of this paper.

REFERENCES

[1] R. Baeza-Yates, J. Culberson, and G. Rawlins, *Searching in the plane*, Inform. and Comput., 106 (1993), pp. 234–252.
[2] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, Elsevier North-Holland, 1976.
[3] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*, Cambridge University Press, 1998.
[4] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia, *Routing with guaranteed delivery in* ad hoc *wireless networks*, in Proc. of Discrete Algorithms and Methods for Mobility (DIALM'99), 1999, pp. 48–55.
[5] V. Chvátal, *A combinatorial theorem in plane geometry*, J. of Combin. Theory Ser. B, 18 (1975), pp. 39–41.
[6] M. de Berg, M. van Kreveld, R. van Oostrum, and M. Overmars, *Simple traversal of a subdivision without extra storage*, Internat. J. GIS, 11 (1997), pp. 359–373.
[7] D. Dobkin, S. J. Friedman, and K. J. Supowit, *Delaunay graphs are almost as good as complete graphs*, Discrete Comput. Geom., 5 (1990), pp. 399–407. See also *28th Symp. Found. Comp. Sci.*, 1987, pp. 20–26.
[8] H. Edelsbrunner, *An acyclicity theorem for cell complexes in d dimension*, Combinatorica, 10 (1988), pp. 251–260.
[9] S. K. Ghosh and S. Saluja, *Optimal on-line algorithms for walking with minimum number of turns in unknown streets*, Computational Geometry Theory and Applications, 8 (1997), pp. 241–266.
[10] C. Icking and R. Klein, *Searching for the kernel of a polygon: A competitive strategy*, in Proc. of the 11th Annual ACM Symposium on Computational Geometry, 1995, pp. 258–266.
[11] J. M. Keil and C. A. Gutwin, *Classes of graphs which approximate the complete Euclidean graph*, Discrete Comput. Geom., 7 (1992), pp. 13–28.
[12] R. Klein, *Walking an unknown street with bounded detour*, Computational Geometry Theory and Applications, 1 (1992), pp. 325–351.
[13] E. Kranakis, H. Singh, and J. Urrutia, *Compass routing on geometric networks*, in Proc. of the 11th Canadian Conference on Computational Geometry (CCCG'99), 1999. To appear.
[14] C. L. Lawson, *Software for C^1 surface interpolation*, Mathematical Software III, Academic Press, New York, 1977, pp. 161–194.
[15] X. Lin and I. Stojmenović, *Geographic distance routing in* ad hoc *wireless networks*, Tech. Report TR-98-10, SITE, University of Ottawa, December 1998.
[16] A. Okabe, B. Boots, and K. Sugihara, *Spatial Tesselations: Concepts and Applications of Voronoi Diagrams*, John Wiley and Sons, 1992.
[17] F. P. Preparata and M. I. Shamos, *Computational Geometry*, Springer-Verlag, New York, 1985.
[18] G. M. Ziegler, *Lectures on Polytopes*, no. 154 in Graduate Texts in Mathematics, Springer-Verlag, New York, 1994.