

Midpoint Routing Algorithms for Delaunay Triangulations

Weisheng Si and Albert Y. Zomaya

Centre for Distributed and High Performance Computing
School of Information Technologies, University of Sydney
Sydney, NSW, Australia
{weisheng, zomaya}@it.usyd.edu.au

Abstract— Memoryless online routing (MOR) algorithms are important for the applications with only local information available to make routing decisions. This paper gives two new MOR algorithms for a class of geometric graphs called Delaunay triangulations (DTs): the Midpoint Routing algorithm and the Compass Midpoint algorithm. More meaningfully, the former is generalized into a set of MOR algorithms that use the Euclidean distance as the reference and work for DTs, and the latter is generalized into a set of MOR algorithms that use the direction as the reference and work for DTs. Many other existing MOR algorithms can also be covered by these two sets. Finally, the two new algorithms are evaluated and compared with other existing MOR algorithms, and the experimental results give new findings on the performances of these algorithms in average and general cases.

Keywords— *geometric graph; shortest paths; Delaunay triangulations; memoryless online routing.*

I. INTRODUCTION

In many application scenarios such as communication networks, transportation planning, and robotics, it often occurs that a packet/vehicle/robot only has local information available to find out its routes. Routing algorithms designed for such scenarios are called online routing algorithms [1]. To be specific, this paper considers *online routing* in the same settings as described in [1]:

- The environment is modeled by a geometric graph $G(V, E)$, where V is the set of nodes with known (x, y) coordinates and E is the set of straight line links connecting the nodes.
- When a packet travels from a source node s to a destination node t , it can remember the coordinates of s and t , and at each node v being visited, can learn the coordinates of the nodes $\in N(v)$, where $N(v)$ denotes the set of v 's neighbors.

Also, if an online routing algorithm \mathcal{A} can route a packet from any source s to any destination t in G , \mathcal{A} is said to *work* for G . If at each node v traversed by a packet, \mathcal{A} makes the routing decision for this packet only according to the coordinates of v , t , and the nodes $\in N(v)$, \mathcal{A} is said to be *memoryless* (or *oblivious*) [2], meaning that a packet remembers no information learned during the traversal of a

graph. Because the memoryless online routing (MOR) algorithms have low complexity in both space and time, they are efficient; and because they only use local information to make routing decisions, they are also scalable. Therefore, the MOR algorithms have received extensive attentions until now [1-3].

For a source/destination pair (s, t) in G , we define the *deviation ratio* of (s, t) by \mathcal{A} as the length of the path found by \mathcal{A} from s to t versus the length of the shortest path from s to t in G . Moreover, for a graph G , we define the *deviation ratio* of G by \mathcal{A} as the average deviation ratio of all (s, t) pairs in G . In practice, the path length generally has two metrics: link distance (the number of links in this path) and Euclidean distance (the sum of the Euclidean distances of all links in this path). Corresponding to the metric used for the path length, we obtain two types of deviation ratios called the *link deviation ratio* and the *Euclidean deviation ratio*. In this paper, we evaluate the performance of a routing algorithm on a graph by measuring these two types of deviation ratios. Note that the deviation ratio of a graph G defined here concerns the average performance of a routing algorithm on G , thus it is different from the *competitive* concept defined in [2], which concerns the upper bound of all path deviation ratios by a routing algorithm on a graph. Also, the deviation ratio concept is different from the *dilation* concept in [4] and the *stretch factor* concept in [5], both of which are defined to measure the path quality of a subgraph G' with respect to a graph G , and hence are not intended for evaluating routing algorithms.

A Delaunay triangulation (DT) is a triangulation graph in which no node lies in the interior of the circumcircle of any of its triangles [6]. DTs have been widely used to model the network topologies [5, 7-10]. Some of the DTs' notable properties desired by routing are as follows:

- Let n denotes the number of nodes, e the number of edges, k the number of convex hull edges, $e = 3n - 3 - k$ holds for any triangulation graph [11]. Therefore, the total number of links in a DT is less than $3n$ and the average node degree is less than 6, thus simplifying the operation of routing.
- In a DT, the Euclidean length of the shortest path between any two nodes u and v is no more than C times

the Euclidean distance between u and v , where C is proved to be between 1.5846 and 2.42 so far [12-13]. As an aside, determining C exactly is one of the most challenging problems in computational geometry.

- DTs are planar graphs.

In view of the above, this paper particularly focuses on the MOR algorithms for DTs. Our contributions mainly include the following:

- Two new MOR algorithms, termed the *Midpoint Routing* algorithm and the *Compass Midpoint* algorithm, are presented and proved to work for DTs.
- More meaningfully, each of the two new algorithms is generalized to a set of algorithms that work for DTs.
- The new algorithms are compared with existing MOR algorithms by experiments on DTs of random node placements. Different from the worst-case perspective of most others' work, our experimental results reveal notable knowledge on the performance of these algorithms in average and general cases.

II. RELATED WORK

In the literature, three well-known MOR algorithms are proved to work for DTs: (1) the Compass Routing algorithm [3], (2) the Greedy Routing algorithm [1], and (3) the Greedy Compass algorithm [2]. As to be described below, they are all characterized by simplicity. Hereafter in this paper, we will use t to denote the destination node of a packet P , v to denote the current processing node of P , $d(a, b)$ to denote the Euclidean distance between node a and node b , and $\angle avb$ to denote the angle ($\leq 180^\circ$) formed by the link va and the link vb .

In Compass Routing, the node v always moves P to the node $w \in N(v)$ that minimizes the angle $\angle tvw$.

In Greedy Routing, the node v always moves P to the node $w \in N(v)$ that minimizes $d(w, t)$.

In Greedy Compass, the node v first decides the two nodes $cw(v)$ and $ccw(v)$, where $cw(v)$ denotes the node w that has the smallest $\angle tvw$ clockwise from the line vt , and $ccw(v)$ denotes the node w that has the smallest $\angle tvw$ counterclockwise from the line vt ; then P is moved to one of $cw(v)$ and $ccw(v)$, whichever has a smaller Euclidean distance to t . As an aside, it is proved in [2] that in addition to DTs, the Greedy Compass algorithm actually works for arbitrary triangulations.

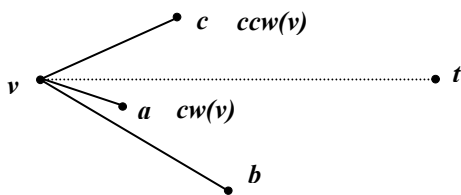


Figure 1. An example scenario for the three MOR algorithms

To illustrate these three MOR algorithms, Figure 1 gives an example network scenario in which Compass Routing moves P to a , Greedy Routing moves P to b , and Greedy Compass moves P to c .

In the evaluation section, we will compare our presented two algorithms with these three MOR algorithms in terms of link and Euclidean deviation ratios, and show that each of these five algorithms has its merits and demerits. Furthermore, we will show that all these five algorithms achieve low link and Euclidean deviation ratios in general case DTs, so they are practical for applications. As to the worst case performance, it is proved in [2] that no upper bound of link deviation ratios exists for the MOR algorithms in DTs, and no upper bound of Euclidean deviation ratios exists for the MOR algorithms in all triangulations. Fortunately, those worst case graphs given in [2] are manually constructed and almost impossible to appear in practice, so they will not affect the practicality of applying these MOR algorithms to DTs.

III. THE MIDPOINT ROUTING ALGORITHM AND ITS GENERALIZATION

Instead of minimizing the Euclidean distance to t as in the Greedy Routing algorithm, the Midpoint Routing algorithm tries to minimize the Euclidean distance to m , where m is the midpoint between the current processing node v and the destination t . The practical meaning of this algorithm is that, if a traveler only aims to reach the midpoint in each move toward the destination, he/she can still reach the destination in the end if he/she moves on a Delaunay triangulation.

The details of this algorithm are described in Figure 2, where $next(v)$ is used to denote the next-hop node decided at node v , and this notation is followed hereafter.

```

1 calculate the coordinates of midpoint  $m$  of  $vt$ ;
2 for each  $w \in N(v)$  {
3     // check whether  $t$  is a neighbor of  $v$ 
4     if (  $w$  is the same node as  $t$  ) {
5          $next(v)$  is set to  $w$ ;
6     }
7     return;
8 }

```

Figure 2. The Midpoint Routing algorithm

For the Midpoint Routing algorithm, we have the following theorem.

Theorem 1: *The Midpoint Routing algorithm works for DTs.*

Proof: We prove this theorem by showing that in each routing step[3] a packet gets strictly closer to t . This proof also exploits that a DT is the dual graph of a Voronoi diagram.

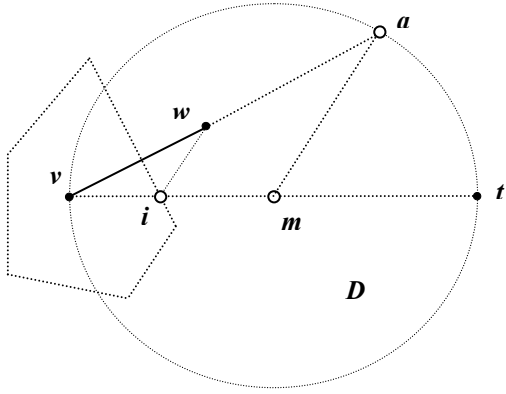


Figure 3. The proof of Theorem 1

Figure 3 depicts an example scenario for the Midpoint Routing algorithm. In this figure, draw the line segment vt , which will intersect with one of the Voronoi edges of the Voronoi region of v . Denote this intersection point i , and denote v 's neighbor corresponding to the intersected Voronoi edge w . Since i lies in the Voronoi region of v , we have $d(i, v) \leq d(i, t)$, so i must lie within the line segment vm , where m is the midpoint of the line segment vt . Consider the disk D with m as the center and mv as the radius, and assume the ray vw intersects the boundary of D at point a . Draw the supporting line segment ma . Since $\angle wit = \angle amt = 2\angle wvi$, Δviw and Δvma are similar triangles. Because we have already established that $d(v, i) \leq d(v, m)$, we have $d(v, w) \leq d(v, a)$. Thus, w must lie within the line segment va , and hence in the disk D . Note a special case is that w will lie on the boundary of D if i coincides with m . Because in the Midpoint Routing algorithm, v chooses its neighbor u that minimizes $d(u, m)$ as the next hop, and because we have shown that v at least has one neighbor w lying in the disk D , u must lie in the disk D too, which guarantees that u is strictly closer to t than v . \square

According to the above proof, if we omit the ‘if’ statement in the algorithm description in Figure 2, Theorem 1 holds as well. The negative effect of this omission is just that in some cases as illustrated in Figure 4, because the processing node v sees that $d(w, m)$ is less than $d(t, m)$, a packet will traverse one more link vw , and then reach t via wt , instead of directly reaching t via vt .

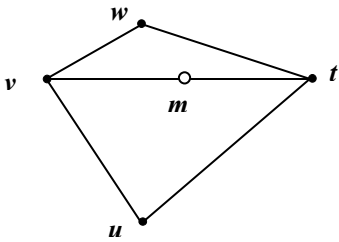


Figure 4. Why the ‘if’ statement is needed

Furthermore, we can prove the following corollary based on the proof for Theorem 1. Basically, this corollary states

that there is a set of MOR algorithms working for DTs by minimizing the distance to an arbitrary point p on the line segment mt . It is worth noting that both the Midpoint Routing algorithm and the Greedy Routing algorithm are special cases of this set of MOR algorithms.

Corollary 1: *Replace the midpoint m with any point p in the line segment mt in the Midpoint Routing algorithm, the newly obtained algorithm works for DTs.*

Proof: We prove this corollary also by showing that the algorithm referencing any p on mt moves a packet strictly closer to t in each routing step.

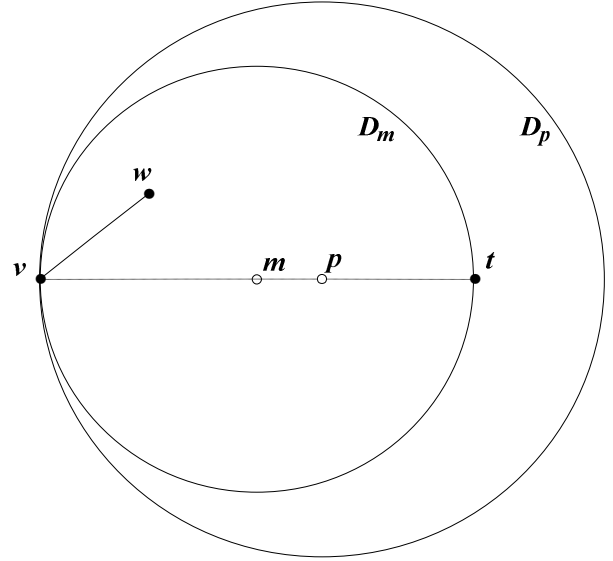


Figure 5. The proof of corollary 1

In Figure 5, D_m is the disk with m as the center and vm as the radius, and D_p is the disk with p as the center and vp as the radius. As proved previously, v must at least have one neighbor w lying in D_m . Since D_m is strictly inside D_p except the point v , w must lie in D_p . Thus, the next hop node u selected by the algorithm referencing p also lies in D_p . Since D_p is inside the disk D_t with t as the center and vt as the radius (due to the space limit, the drawing of D_t is omitted in Figure 5), u must strictly lie in D_t , thus being strictly closer to t than v . \square

IV. THE COMPASS MIDPOINT ALGORITHM AND ITS GENERALIZATION

In this section, we first present a set of algorithms called the Deterministic Compass algorithms, and prove that they work for DTs. Then, we present the Compass Midpoint algorithm, which is a special case of the Deterministic Compass algorithms, thus its working for DT follows.

Bearing a similar structure with the Greedy Compass algorithm, the set of Deterministic Compass algorithms work as follows: the processing node v first decides the two nodes $cw(v)$ and $ccw(v)$, and then selects one of them as $next(v)$ using a deterministic rule. The details of this set of

algorithms are described in Figure 6. Note that the ‘if’ statement in line 1 means that when a node v happens to have a neighbor w lying on the line segment vt , both $cw(v)$ and $ccw(v)$ are taken to be w , and hence $next(v)$ is directly set to w . And the deterministic rule in line 5 can be any rule that always selects the same node in the same situation. For example, this deterministic rule can be always selecting the $cw(v)$. More meaningfully, if this deterministic rule is to select one of the $cw(v)$ and $ccw(v)$ that has a smaller angle to t , we get the Compass Routing algorithm; and if this deterministic rule is to select one of the $cw(v)$ and $ccw(v)$ that has a smaller distance to t , we get the Greedy Compass algorithm. Note that for these two examples, we assume that the ties are also broken by a deterministic rule.

```

1 if (v has a neighbor w lying on the segment vt )
2   next(v) is set to w;
3 else {
4   decides the two nodes cw(v) and ccw(v);
5   next(v) is set to one of them with a deterministic rule;
6 }

```

Figure 6. The set of Deterministic Compass algorithms

As a note, if a fully random rule is used in line 5 to replace the deterministic rule, we get the Randomized Compass algorithm, which is more capable and proved in [1] to work for arbitrary triangulations. Unfortunately, the set of Deterministic Compass algorithms can not work for arbitrary triangulations, since Ref. [3] gives a counter-example triangulation in which the Compass Routing algorithm fails, while the Compass Routing algorithm is a special case of the Deterministic Compass algorithms.

Next, we first prove two lemmas, and then prove a theorem and a corollary stating that the set of Deterministic Compass algorithms work for the *regular* triangulations, and especially for the DTs. Note that a triangulation is *regular*, if it can be obtained by vertically projecting the faces of the lower convex hull of a 3-dimension polytope onto the plane [14]. Also note that the proof techniques here follow those used in [1] to prove that the Compass Routing algorithm works for DTs.

Lemma 1: *For a source/destination pair (s, t) in a triangulation graph T , if a Deterministic Compass algorithm cannot route a packet P from s to t , P must be trapped in a cycle, and the link distance of this cycle is larger than two.*

Proof: Since a Deterministic Compass algorithm makes the same routing decision at the same node each time, and there is limited number of nodes in T , a packet P must be trapped in a cycle if it never gets to t .

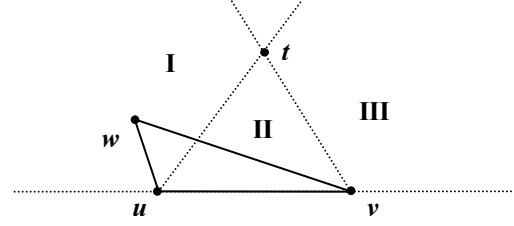


Figure 7. Proof of Lemma 1

We now prove that the link distance of this cycle is larger than two. In other words, there does not exist a link uv in T , such that $next(u) = v$ and $next(v) = u$ for a Deterministic Compass algorithm. Suppose there exists such a link uv . We depict the scenario for it in Figure 7. Since T is connected and is a triangulation, u and v must have a common neighbor w located in the one of the three regions I, II, or III, and u, v, w forms a triangle. If w is located in region I, $next(v)$ cannot be u . If w is located in region II, $next(v)$ cannot be v . Similarly, if w is located in region III, $next(u)$ cannot be v . No matter which region w lies in, contradiction occurs, so such a link uv cannot exist. \square

With Lemma 1, we know that if a Deterministic Compass algorithm does not work for a triangulation T , there must exist a trapping cycle with link distance larger than two for a certain (s, t) pair in T . Next, Lemma 2 gives a visibility property of such trapping cycles in terms of the *obscure* concept, which is defined as follows [1]. Let ΔA and ΔB be two triangles in T . ΔA is said to *obscure* ΔB with respect to a viewpoint z on the same plane, if there exists a ray from z reaching any point in ΔA first and then any point in ΔB .

Let u and v be any two nodes in T such that $next(u) = v$ by a Deterministic Compass algorithm for a destination t . Define Δuv as the triangle in T that lies in the half-plane bounded by the line through uv and containing t . Then we have the following Lemma on the visibility of Δuv s in a trapping cycle.

Lemma 2: *If a Deterministic Compass algorithm is trapped in a cycle $v_0 v_1 v_2 \dots v_{k-1} v_0$ for a source/destination pair (s, t) in a triangulation T , $\Delta v_i v_{i+1}$ is either identical to $\Delta v_{i-1} v_i$ or obscures $\Delta v_{i-1} v_i$ with respect to the viewpoint t . ($0 \leq i < k$, and all subscripts are the results of mod k).*

Proof: If $\Delta v_i v_{i+1}$ is identical to $\Delta v_{i-1} v_i$, nothing needs to be proved here. If $\Delta v_i v_{i+1}$ is not identical to $\Delta v_{i-1} v_i$, we need to prove that $\Delta v_i v_{i+1}$ obscures $\Delta v_{i-1} v_i$.

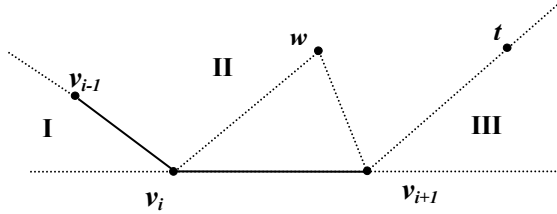


Figure 8. The proof of Lemma 2

In Figure 8, let w be the third node of $\Delta v_i v_{i+1}$, then w can only lie in the regions I, II, or III separated by the ray $v_i v_{i-1}$ and the ray $v_{i+1} t$. If w is in region I, there will be crossing links in T , contradicting to the planarity of T . Note that we assume v_{i-1} is located in the upper half plane here; otherwise, there are only regions II and III, and this proof step can be omitted. If w is in region III, $next(v_i)$ cannot be v_{i+1} . So w can only lie in region II. In this case, the ray from t to v_i reaches the link $v_{i+1} w$ first and then reaches v_i , a point in $\Delta v_{i-1} v_i$. Therefore, $\Delta v_i v_{i+1}$ obscures $\Delta v_{i-1} v_i$. \square

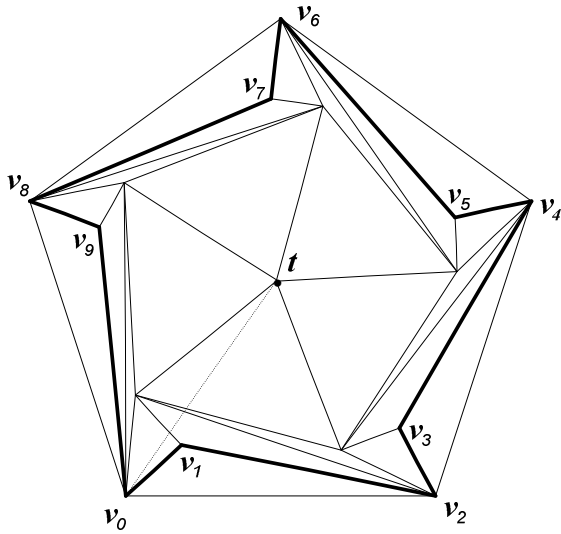


Figure 9. An example of the trapping cycles

To visualize the visibility property of the trapping cycles, Figure 9 shows a triangulation in which the Compass Routing algorithm falls into the cycle $v_0 v_1 v_2 \dots v_9 v_0$ (depicted in bold). In this example, with the dotted supporting line tv_0 , it is easy to see that $\Delta v_0 v_1$ obscures $\Delta v_9 v_0$, $\Delta v_1 v_2$ obscures $\Delta v_0 v_1$, and so on.

Given the above two lemmas, we are ready to prove the following theorem.

Theorem 2: *Any Deterministic Compass algorithm works for regular triangulations.*

Proof: In [15], Edelsbrunner proved that if a triangulation T is a regular triangulation, T has no set of triangles that can form an obscuring cycle with respect to any viewpoint in T . Given Lemma 1 and Lemma 2, if a Deterministic Compass algorithm does not work for T , there must exist a set of triangles forming an obscuring cycle, thus causing contradictions. Therefore, this theorem holds. \square

Since a DT is the projection onto the plane of the lower convex hull of a set of points that all lie on a paraboloid, a DT is a special case of the regular triangulations [6]. Thus, the following corollary holds.

Corollary 2: *Any Deterministic Compass algorithm works for DTs.*

We now describe the Compass Midpoint algorithm. It is obtained by simply setting the deterministic rule at line 5 in Figure 6 to the following: select the $cw(v)$ and $ccw(v)$ whichever has a smaller Euclidean distance to m , where m is the midpoint of the line segment vt . Since the Compass Midpoint algorithm belongs to the Deterministic Compass algorithms, we have the following corollary.

Corollary 3: *The Compass Midpoint algorithm works for the regular triangulations, especially the DTs.*

V. EVALUATION

In this section, we evaluate and compare the following five MOR algorithms: (1) Compass Routing, (2) Greedy Routing, (3) Greedy Compass, (4) Midpoint Routing, and (5) Compass Midpoint. The former three are existing MOR algorithms discussed in Section II, and the latter two are the MOR algorithms given by us. Hereafter, they are abbreviated as ComRtg, GdyRtg, ComGdy, MidRtg, and ComMid for convenience.

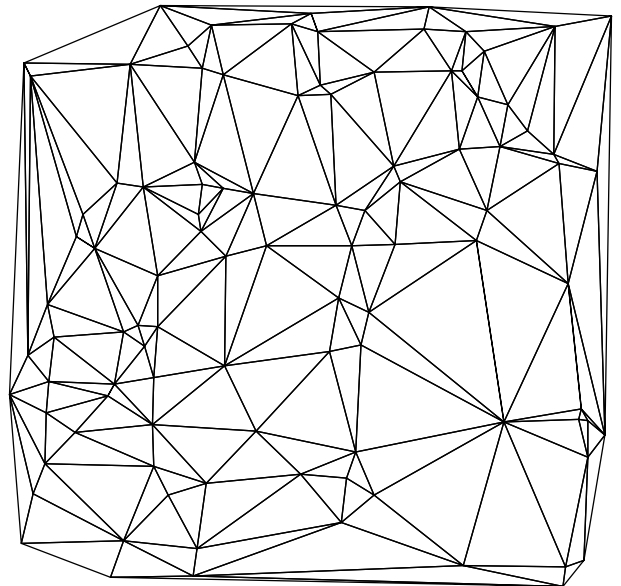


Figure 10. An example DT of 100 nodes

We develop a computer program that implements the above five MOR algorithms on geometric graphs and calculates their link deviation ratios and Euclidean deviation ratios. With this program, we totally conduct experiments on 1000 DTs of 100 nodes. For each DT, the positions of its 100 nodes are randomly generated with a uniform distribution in a square area, and then the DT is built on these nodes by the open source software Triangle [16]. Figure 10 shows an example DT of 100 nodes used in our experiments. In our experiment on a DT, we run our computer program and obtain the link deviation ratios and Euclidean deviation ratios by the five MOR algorithms on this DT. At last, we calculate the average and 99th percentile results from the experiments on these 1000 DTs to reflect the performances of these five algorithms in average and general cases.

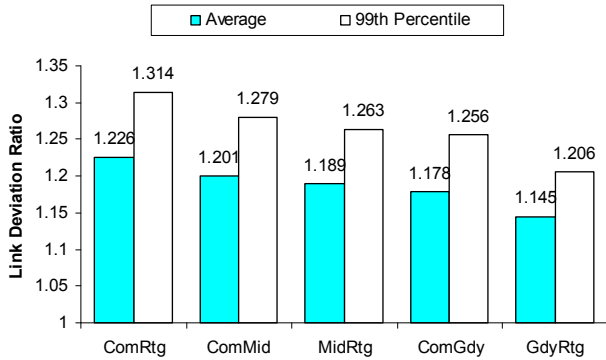


Figure 11. The average and 99th percentile link deviation ratios of the five algorithms

Figure 11 plots the average and 99th percentile link deviation ratios of these 1000 DTs for the five MOR algorithms. From this figure, we can conclude the following.

- Both average and 99th percentile link deviation ratios of these five algorithms are very small (all below 1.32), so all of them perform well in average and general cases, and hence are practical for applications.
- In terms of both average and 99th percentile link deviation ratios, GdyRtg performs the best, and the next four ones are in turn ComGdy, MidRtg, ComMid, and ComRtg. This reflects that minimizing the Euclidean distance to the destination at each routing step is very effective in reducing the link distances of the paths, while minimizing the angle to the destination is less effective.
- The two new algorithms MidRtg and ComMid perform in the middle among these five algorithms.

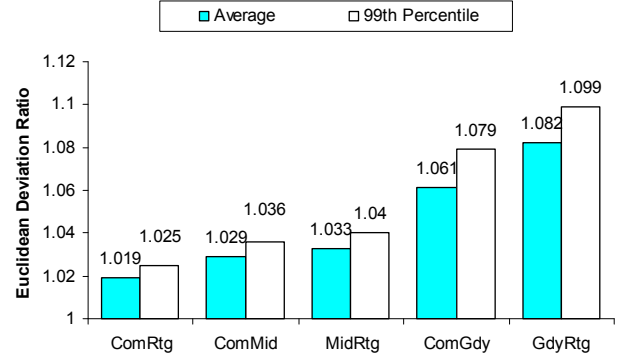


Figure 12. The average and 99th percentile Euclidean deviation ratios of the five algorithms

Figure 12 plots the average and 99th percentile Euclidean deviation ratios of these 1000 DTs for the five MOR algorithms. From this figure, we can conclude the following.

- Both average and 99th percentile Euclidean deviation ratios of these five algorithms are very small (all below 1.1), so all of them perform well in average and general cases, and hence are practical for applications.
- In terms of both average and 99th percentile Euclidean deviation ratios, ComRtg performs the best, and the next four ones are in turn ComMid, MidRtg, ComGdy, and GdyRtg, which is exactly in the reverse order of the link deviation ratios. This reflects that minimizing the angle to the destination at each routing step is very effective in reducing the Euclidean distances of the paths, while minimizing the Euclidean distance to the destination is less effective.
- The two new algorithms MidRtg and ComMid perform in the middle among these five algorithms.

VI. CONCLUSIONS AND OPEN PROBLEMS

In this paper, we presented and proved two new MOR algorithms that work for DTs: the Midpoint Routing algorithm and the Compass Midpoint algorithm. More significantly, we generalized the former into a set of MOR algorithms that use the Euclidean distance as the reference and work for DTs, and generalized the latter into a set of MOR algorithms that use the direction as the reference and work for DTs. Other existing MOR algorithms discussed in this paper can also be covered by these two sets. These two generalizations reflect that DTs are the class of graphs for which a broad range of MOR algorithms work.

We also evaluated and compared our presented two algorithms with other three well-known MOR algorithms. Our experimental results mainly show the following. First, all these five algorithms can find paths with low link and Euclidean deviation ratios on DTs in average and general cases, so they are practical for applications. Second, among them, the Greedy Routing algorithm performs the best in terms of link deviation ratios on DTs, and the worst in terms

of Euclidean deviation ratios on DTs, while the Compass Routing algorithm does the reverse. This phenomenon implies that minimizing the Euclidean distance to the destination is effective in reducing the link deviation ratio on DTs, while minimizing the angle to the destination is effective in reducing the Euclidean deviation ratio on DTs. Third, the presented two algorithms perform in the middle among these five algorithms in terms of both link and Euclidean deviation ratios, so they are suitable for the applications requiring satisfactory performance in both link and Euclidean metrics.

As mentioned previously, the Greedy Compass algorithm works for arbitrary triangulations, suggesting that combining the references to angles and to Euclidean distances can generate more capable MOR algorithms. This leads to the conjecture that *the Compass Midpoint algorithm presented in this paper also works for arbitrary triangulations*. Right now, we cannot prove or disprove this conjecture. If this conjecture is proved true, the Compass Midpoint algorithm will have more applications.

ACKNOWLEDGMENT

We thank Dr Pat Morin and Dr Joachim Gudmundsson for discussion during this work.

REFERENCES

- [1] P. Bose and P. Morin, "Online Routing in Triangulations," *Siam Journal of Computing*, vol. 33, pp. 937-951, 2004.
- [2] P. Bose, A. Brodnik, S. Carlsson, E. D. Demaine, R. Fleischer, A. Lopez-ortiz, P. Morin, and J. I. Munro, "Online routing in convex subdivisions," *International Journal of Computational Geometry*, vol. 12, pp. 283-295, 2002.
- [3] E. Kranakis, H. Singh, and J. Urrutia, "Compass routing on geometric networks," in *Canadian Conference on Computational Geometry*, 1999, pp. 51-54.
- [4] D. Eppstein, "Spanning trees and spanners," Technical Report 96-16, Dept. of Information and Computer Science, Univ. of California, Irvine, 1996.
- [5] J. Gao, L. J. Guibas, J. Hershberger, L. Zhang, and A. Zhu, "Geometric spanners for routing in mobile networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, pp. 174-185, 2005.
- [6] J. O'Rourke, *Computational Geometry in C*, 2nd ed. Cambridge, UK: Cambridge University Press, 1998.
- [7] X.-Y. Li, G. Calinescu, P.-J. Wan, and Y. Wang, "Localized Delaunay Triangulation with Applications in Wireless Ad Hoc Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, pp. 1035-1047, 2003.
- [8] J. Liebeherr, M. Nahas, and W. Si, "Application-Layer Multicasting with Delaunay Triangulation Overlays," *IEEE Journal on Selected Areas in Communications*, vol. 20, pp. 1472-1488, 2002.
- [9] W. Si and S. Selvakennedy, "A Position-Based Deployment and Routing Approach for Directional Wireless Mesh Networks," in *International Conference on Computer Communications and Networks (ICCCN)*, 2008, pp. 1-8.
- [10] Y. Wang and X.-Y. Li, "Efficient Delaunay-based Localized Routing for Wireless Sensor Networks," *International Journal of Communication Systems*, vol. 20, pp. 767 - 789, 2007.
- [11] M. d. Berg, M. v. Kreveld, M. Overmars, and O. Schwarzkopf, *Computational geometry: algorithms and applications*, 2nd ed. New York: Springer, 2000.
- [12] P. Bose, L. Devroye, M. Löffler, J. Snoeyink, and V. Verma, "The spanning ratio of the Delaunay triangulation is greater than $\pi/2$," in *Canadian Conference on Computational Geometry*, 2009, pp. 1-3.
- [13] J. M. Keil and C. A. Gutwin, "Classes of graphs which approximate the complete euclidean graph," *Discrete and Computational Geometry*, vol. 7, pp. 13-28, 1992.
- [14] G. M. Ziegler, *Lectures on polytopes*. New York: Springer-verlag, 1994.
- [15] H. Edelsbrunner, "An acyclicity theorem for cell complexes in d dimension " *Combinatorica*, vol. 10, pp. 251-260, 1990.
- [16] J. R. Shewchuk. *Triangle version 1.6*. Available: <http://www.cs.cmu.edu/~quake/triangle.html>, 2005.