

Two-step and Apex-angle Routing Algorithms for Delaunay Triangulations

Weisheng Si and Albert Y. Zomaya

ABSTRACT

Memoryless online routing (MOR) algorithms are suitable for the applications only using local information to discover paths, and Delaunay triangulations are the class of geometric graphs widely proposed as wireless network topologies. Motivated by these two facts, this paper reports two new MOR algorithms for the Delaunay triangulations: the Two-step Routing algorithm and the Apex-angle Routing algorithm, thus greatly enriching the family of such algorithms. This paper also evaluates and compares these two new algorithms with three existing MOR algorithms. The experimental results shed light on their performance in terms of both Euclidean and link metrics, and also reveal some intrinsic properties of Delaunay triangulations. Finally, this paper poses two open problems, with their importance explained.

Keywords

Memoryless online routing; shortest paths; geometric graphs; Delaunay triangulations.

1. INTRODUCTION

In wireless communication environments such as sensor networks, mesh networks, and vehicular networks, it often occurs that a packet/vehicle only has local information available to find out its routes. Routing algorithms designed for such a scenario are called online routing algorithms [4]. To be specific, this paper considers *online routing* in the same settings as those described in [4]:

- 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 one-hop neighbors.

Moreover, 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 processing node v , \mathcal{A} makes the routing decision for a packet only according to the coordinates of v , the destination t , and the nodes $\in N(v)$, \mathcal{A} is said to be *memoryless* (or *oblivious*) [3], meaning that a packet records no information learned during the traversal of a graph. Because the memoryless online routing (MOR) algorithms have low complexity in both space and time for both nodes and packets, 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 attention in the literature [3-5, 7-8, 13].

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 . 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 . The implication of deviation ratio is that when the ratio is closer to 1, the paths found by \mathcal{A} achieve less deviation from the actual shortest paths. 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). In relation 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* respectively. 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 concept defined here is different from the *c-competitive* concept in [4], which defines that a routing algorithm is *c-competitive* for a graph G , if for any (s, t) pair $\in G$, its deviation ratio is not greater than a constant c . Therefore, the *c-competitive* concept measures the worst-case performance of a routing algorithm on G , concerning the theoretical aspect of the existence of an upper bound c . In contrast, the deviation ratio concept measures the average performance of a routing algorithm on G ,

concerning the practical aspect of applying this routing algorithm in general. Moreover, the deviation ratio concept is different from the *dilation* concept in [9] and the *stretch factor* concept in [11], both of which are defined to measure the shortest path quality of a subgraph G' with respect to a graph G , and hence are not intended for evaluating routing algorithms.

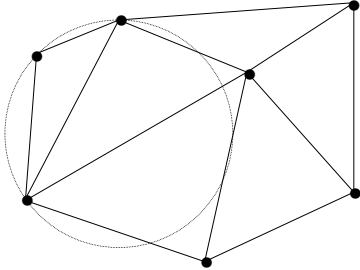


Figure 1. An example of Delaunay triangulations

A Delaunay triangulation (DT) is a triangulation graph in which no node lies in the interior of the circumcircle of any of its triangles [16]. Figure 1 illustrates an example DT. DTs have been widely proposed as the network topologies [11, 14-15, 18-19]. Some of the DTs' desirable properties for 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 [2]. 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 up to now [6, 12]. As an aside, determining C exactly is one of the most challenging problems in computational geometry.
- DTs are planar graphs.

In light of the above, this paper particularly focuses on the MOR algorithms for DTs. The contributions of this paper are mainly as follows:

- Two new MOR algorithms, termed the *Two-step Routing* algorithm and the *Apex-angle Routing* algorithm, are reported and proved to work for DTs, thus greatly enriching the family of MOR algorithms for DTs.
- The two new algorithms are evaluated and compared with three existing MOR algorithms. Different from most others' worst-case perspective, our

experimental results reveal new findings on the performance of these algorithms in average and most cases.

- For the different metrics of link and Euclidean distances, the shortest paths obtained in DTs are shown to be discrepant in a high percentage.

The rest of this paper is structured as follows. Section 2 surveys the literature; Sections 3 and 4 describe our two new algorithms, and prove that they work for DTs; Section 5 presents and discusses the experimental results on the MOR algorithms and the properties of DTs; Section 6 poses two open problems, with their importance explained; finally, Section 7 summarizes the main results of this paper, and then remarks on them.

2. RELATED WORK

Since the MOR algorithms exhibit simplicity and elegance, they are fascinating to pursue. To the best of our knowledge, three MOR algorithms are proved to work for DTs to date: (1) the Compass Routing algorithm [13], (2) the Greedy Routing algorithm [4], and (3) the Greedy Compass algorithm [3]. Below, we describe them briefly. Hereafter in this paper, we will use P to denote a packet, s to denote the source node of P , t to denote the destination node of P , v to denote the current processing node, $d(a, b)$ to denote the Euclidean distance between node a and node b , and $\angle avb$ to denote the angle ($\leq 180^\circ$) between 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 $\angle tvw$; and a tie is broken arbitrarily.

In Greedy Routing, the node v always moves P to the node $w \in N(v)$ that minimizes $d(w, t)$; and a tie is broken arbitrarily.

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 ; in case a node $w \in N(v)$ lies on the line segment vt , we have $cw(v) = ccw(v) = w$. Then, P is moved to one of the $cw(v)$ and $ccw(v)$ whichever has a smaller Euclidean distance to t ; and a tie is broken arbitrarily. As an aside, it is proved in [3] that the Greedy Compass algorithm actually works for arbitrary triangulations, not only DTs.

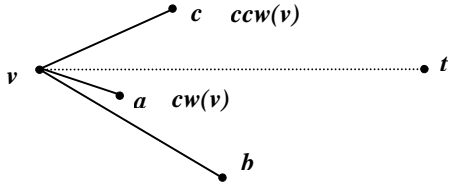


Figure 2. An example scenario for the three MOR algorithms

To illustrate these three MOR algorithms, Figure 2 gives an example network scenario where the node v moves P to node a with Compass Routing, moves P to node b with Greedy Routing, and moves P to node c with Greedy Compass.

In the evaluation section, we will compare and contrast our two new algorithms with these three existing MOR algorithms in terms of Euclidean and link deviation ratios, and also provide explanation on their performance.

3. THE TWO-STEP ROUTING ALGORITHM

While the Compass Routing minimizes the angle to t and the Greedy Routing minimizes the Euclidean distance to t , the Two-step Routing tries to minimize the sum of $d(v, w)$ and $d(w, t)$, where w is a neighbor of v . Note that a tie can be broken arbitrarily. This basic idea of Two-step Routing is illustrated in Figure 3, where node v will move a packet to node a according to this idea. Since the locus of w satisfying $d(v, w) + d(w, t) = L$ (a constant) is an ellipse with v and t as the two foci, the Two-step Routing algorithm can be viewed as restricting its searching area of a node $w \in N(v)$ with such an ellipse, and increasing this searching area gradually until the first $w \in N(v)$ is encountered.

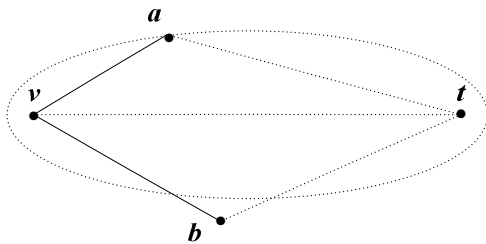


Figure 3. The basic idea of Two-step Routing

Unfortunately, if the Two-step Routing algorithm only tries to minimize the $d(v, w) + d(w, t)$, it fails for some DTs. Figure 4 gives such an example DT, where the (x, y) coordinates of its nodes are labeled. In this DT, it is easy to verify that a packet will repeatedly travel

from v to w , and then from w to v , if only $d(v, w) + d(w, t)$ is considered.

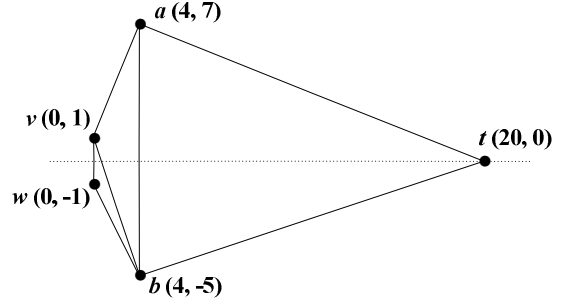


Figure 4. A DT in which the basic Two-step Routing fails

To remedy this problem, we add a checking condition that only when a $w \in N(v)$ has a smaller $d(w, t)$ than $d(v, t)$, minimizing the $d(v, w) + d(w, t)$ is used as the criteria to select the next hop. Thus, a packet is guaranteed to get strictly closer to the destination in each move, and reach the destination in the end. The feasibility for this checking condition is based on the following lemma, which is proved in [4]:

Lemma 3.1: *In a DT, for any node pair (v, t) , v at least has one neighbor w with its $d(w, t) < d(v, t)$.*

Thus, the entire Two-step Routing algorithm not only involves two comparison steps, but also involves two spacial steps $d(v, w)$ and $d(w, t)$. This explains why it is so named. Figure 5 gives the details of this algorithm, in which $next(v)$ is used to denote the next-hop node decided at node v , and this notation is followed hereafter.

```

1 for each  $w \in N(v)$  {
2   if (  $w$  has a smaller  $d(w, t)$  than  $d(v, t)$  ) {
3      $sum = d(v, w) + d(w, t)$ ;
4     if (  $w$  has a smaller  $sum$  than previous ) {
5        $next(v) = w$ ;
6     }
7   }
8 }

```

Figure 5. The Two-step Routing algorithm

For the Two-step Routing algorithm, we have the following theorem.

Theorem 3.1: *The Two-step Routing algorithm works for DTs.*

Proof: According to Lemma 3.1, the Two-step Routing algorithm moves a packet strictly closer to the destination at each routing step, so a packet visits a different node in each move. Since there is limited

number of nodes in a DT, the packet must reach the destination eventually; otherwise, there should be infinite number of nodes in a DT. \square

As an aside, people will wonder whether the following algorithm combination can give us a new MOR algorithm for DTs: first deciding those $w \in N(v)$ with $d(w, t) < d(v, t)$ and then selecting one of them minimizing the $\angle tvw$. Unfortunately, it cannot. Since it is proved in [13] that the $next(v)$ selected by the Compass Routing algorithm always has a smaller distance to t than v , the above combination is actually equivalent to the Compass Routing algorithm.

4. THE APEX-ANGLE ROUTING ALGORITHM

We call $\angle vwt$ the apex angle for the current processing node v , a node $w \in N(v)$, and the destination t . The basic idea of the Apex-angle Routing algorithm is to maximize this apex angle in each routing step. Note that a tie can be broken arbitrarily. This idea is illustrated in Figure 6, where node v will move a packet to node a according to this idea. Since the locus of w satisfying $\angle vwt = \theta$ (a constant angle) is the two symmetric arcs with vt as the chord, the Apex-angle Routing algorithm can be viewed as restricting its searching area of a $w \in N(v)$ with these two arcs, and increasing this searching area gradually until the first $w \in N(v)$ is encountered.

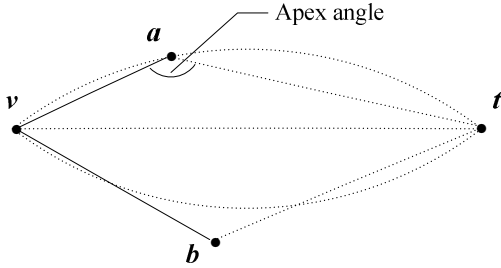


Figure 6. The idea of Apex-angle Routing

As to be proved later, simply maximizing the apex angle guarantees a packet reaching the destination in DTs. Thus, the Apex-angle Routing algorithm can be described as follows.

```

1 for each  $w \in N(v)$  {
2   if (  $w$  has a larger  $\angle vwt$  than previous ) {
3      $next(v) = w$ ;
4   }
5 }
```

Figure 7. The Apex-angle Routing algorithm

To show the Apex-angle Routing algorithm works for DTs, we first prove the following lemma.

Lemma 4.1: *In a DT, for any node pair (v, t) , v at least has one neighbor w that lies in the disk D (including the boundary) with vt as the diameter.*

Proof: This proof exploits the fact that a DT is the dual graph of a Voronoi diagram [16].

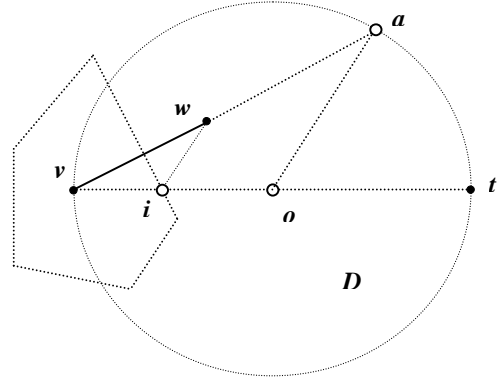


Figure 8. The proof of Lemma 4.1

If t is a neighbor of v , this lemma holds. If not, Figure 8 depicts such an example scenario, showing a node v , the Voronoi region of v , and a destination node t in a DT. In this figure, draw the line segment vt , which must intersect with one of the Voronoi edges of the Voronoi region of v . Denote this intersection point i , and denote v 's neighbor node 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 vo , where o is the center of the disk D with vt as the diameter. Suppose the ray vw intersects the boundary of disk D at point a . Draw the supporting line segment oa .

Since a Voronoi edge is the perpendicular bisector of the dual DT edge, Δviw is an isosceles triangle. While Δvoa is also an isosceles triangle, we have Δviw and Δvoa are similar triangles. Because we have already established that $d(v, i) \leq d(v, o)$, 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 o . \square

For Lemma 4.1, we have the following two notes. First, when a node $w \in N(v)$ lying in the disk D with vt as the diameter, it can be easily proved that $d(w, t) < d(v, t)$, so Lemma 4.1 is a stronger result than Lemma 3.1. Second, it is actually proved in [10] that there exists an entire path from v to t lying in the disk D , which implies Lemma 4.1. Since we believe Lemma 4.1 will have

many applications by itself, we make it standalone here, and provide a more direct proof for it than the proof in [10]. Next, we will prove the following theorem with Lemma 4.1.

Theorem 4.1: *The Apex-angle Routing algorithm works for DTs.*

Proof: We prove by showing that in each move, a packet gets strictly closer to t .

According to Lemma 4.1, v at least has one neighbor w lying in the disk D with vt as the diameter. Thus, the apex angle for this w (namely, $\angle vwt$) is not less than 90° . Because the Apex-angle Routing algorithm tries to maximize the apex angle, the $next(v)$ obtained by this algorithm must have an apex angle at least as large as $\angle vwt$. Thus, $next(v)$ must also lie in the disk D , implying that $next(v)$ has a smaller distance to t than v . Then, following the same arguments in proving Theorem 3.1, this theorem holds. \square

In implementing the Apex-angle Routing algorithm, we calculate $\cos(\angle vwt)$ to find out the largest $\angle vwt$. To calculate $\cos(\angle vwt)$, we use the following well-known cosine formula:

$$\cos(\angle vwt) = \frac{d^2(v, w) + d^2(w, t) - d^2(v, t)}{2 \cdot d(v, w) \cdot d(w, t)}$$

Thus, $d(w, t)$ needs to be calculated first. Moreover, because the final w selected must have a smaller $d(w, t)$ than $d(v, t)$ as established in Theorem 4.1, we can avoid the complex computation of $\cos(\angle vwt)$ by adding a checking condition as follows: only when a $w \in N(v)$ has a smaller $d(w, t)$ than $d(v, t)$, $\cos(\angle vwt)$ is calculated. With this speed-up technique, the improved Apex-angle Routing algorithm is described in Figure 9, which exactly has the same structure as the Two-step Routing algorithm described in Figure 5.

```

1 for each  $w \in N(v)$  {
2   if (  $w$  has a smaller  $d(w, t)$  than  $d(v, t)$  ) {
3     if (  $w$  has a larger  $\angle vwt$  than previous ) {
4        $next(v) = w$ ;
5     }
6   }
7 }
```

Figure 9. The improved Apex-angle Routing algorithm

5. EVALUATION

In this section, we evaluate and compare the two MOR algorithms reported by us and the three existing MOR algorithms described in the related work: (1) Two-

step Routing, (2) Apex-angle Routing, (3) Compass Routing, (4) Greedy Routing, and (5) Greedy Compass. Hereafter, they are abbreviated as TwoStp, ApxAng, ComRtg, GdyRtg, and ComGdy for brevity. Specifically, our evaluations cover the following four aspects.

- The Euclidean and link deviation ratios of these five algorithms on DTs.
- The impact of the node number of DTs on the Euclidean and link deviation ratios.
- The impact of the node density of DTs on the Euclidean and link deviation ratios.
- As an investigation of the property of DTs, we measure the possibility for the shortest paths in DTs in Euclidean metric and link metric to be discrepant.

5.1 Experiment setup

We develop a computer program that implements the above five MOR algorithms and performs measurements presented in this section. With this program, we conduct experiments on DTs with random node placement. For each DT, its nodes are uniformly distributed in a square area. We adopt the uniform distribution for node placement in our experiments because the uniform distribution is the most common one used in the research on wireless networks (e.g., [1, 11, 14, 19]). We build the DTs on the uniformly-distributed nodes by the open source software Triangle [17]. Figure 10 shows an example DT of 100 nodes used in our experiments.

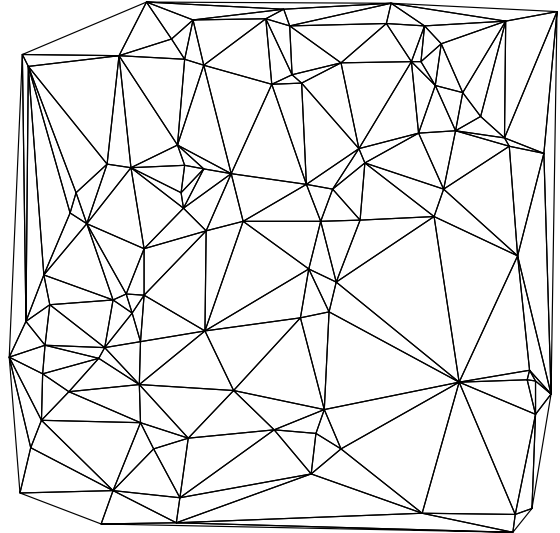


Figure 10. An example DT of 100 nodes

5.2 Euclidean and link deviation ratios on DTs

To compare the performances of these five algorithms in average and most cases, we run these five MOR algorithms totally on 1000 DTs of 100 nodes, and obtain their Euclidean and link deviation ratios for each DT. Then, we calculate the average and the 99th percentile Euclidean and link deviation ratios from these 1000 experiments for these five algorithms. The average value is used to reflect the average-case performance, and the 99th percentile value is used to reflect the performance in most cases.

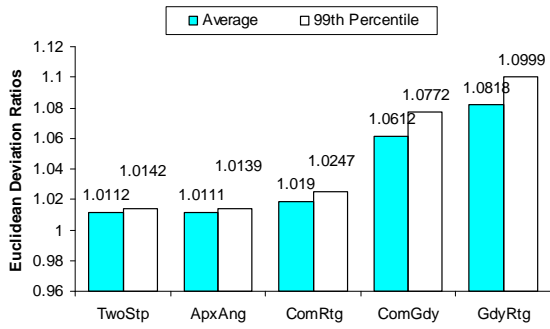


Figure 11. The average and the 99th percentile Euclidean deviation ratios of the five algorithms

Figure 11 plots the average and the 99th percentile Euclidean deviation ratios from these 1000 DTs for the five MOR algorithms. From this figure, we can draw the following two conclusions. First, 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 most cases, and hence are practical for applications.

Second, in terms of both average and 99th percentile Euclidean deviation ratios, the performance rankings of these five algorithms are: ApxAng, TwoStp, ComRtg, ComGdy, and GdyRtg. Note that the performances of TwoStp, ApxAng, and ComRtg are very close. The reason for such a performance ranking is that TwoStp, ApxAng, and ComRtg are effective in selecting a next-hop node that is close to the line vt , while GdyRtg has the tendency to select a next-hop node that is far away from the line vt . Thus, TwoStp, ApxAng, and ComRtg find paths with much less Euclidean length than GdyRtg, while ComGdy performs in between, because it is a combination of the ComRtg and the GdyRtg.

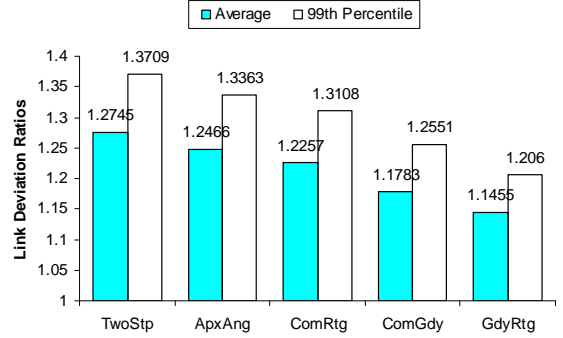


Figure 12. The average and the 99th percentile link deviation ratios of the five algorithms

Figure 12 plots the average and the 99th percentile link deviation ratios of the 1000 DTs for the five MOR algorithms. From this figure, we can draw the following three conclusions. First, both average and 99th percentile link deviation ratios of these five algorithms are very small (all below 1.371), so all of them perform well in average and most cases, and hence are practical for applications.

Second, the link deviation ratios of all these five algorithms are larger than their Euclidean counterparts. This reflects that the structure of DTs is more exploitable for a routing algorithm to reduce the Euclidean distance than to reduce the link distance. Intuitively, in DTs, many links can be traversed while the total Euclidean length is still small. This intuition is illustrated in Figure 13, where the path $vab\dots gt$ is the shortest Euclidean path from v to t , but it traverses much more links than the path vwt , which is the shortest link path from v to t .

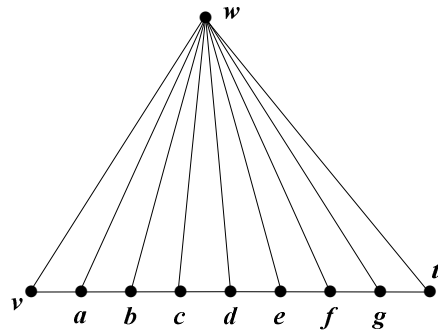


Figure 13. A DT in which a path consisting of many links has a small Euclidean length

Finally, in terms of both average and 99th percentile link deviation ratios, the performance rankings of these five algorithms are: GdyRtg, ComGdy, ComRtg, ApxAng, and TwoStp. Note that this ranking is almost exactly the reverse to ranking of the Euclidean deviation

ratio. The reason for such a performance ranking is that GdyRtg tries to get close to t as much as possible, while ComRtg, ApxAng, and TwoStp tend to progress along the line vt . And the strategy of GdyRtg is more effective in reducing the link distance of the paths.

While Figure 11 and Figure 12 present the performance of these five algorithms in average and most cases, we have the following two notes on the worst case performance of these algorithms. First, it is proved in [3] that no upper bound of the link deviation ratio for a (s, t) pair in DTs exists for all online routing algorithms, thus it follows immediately that no upper bound exists for all MOR algorithms. Also, it is proved in [4] that no upper bound of the Euclidean deviation ratio for a (s, t) pair in DTs exists for the ComRtg and GdyRtg algorithms. For other MOR algorithms, it is not known yet whether an upper bound of the Euclidean deviation ratio exists for a (s, t) pair in DTs. Fortunately, those worst case DTs manually constructed in [3-4] are almost impossible to appear in practice, so they will not affect the practicality of applying these MOR algorithms to DTs. Second, since the worst case deviation ratios are not bounded, the worst case experimental data will depend on whether an extremely bad DT is generated among the random samples of DTs. This explains why we do not record the worst case data in our experiments.

5.3 The impact of the node number of DTs on the Euclidean and link deviation ratios

To examine the impact of the node number of DTs on the Euclidean and link deviation ratios, we run these five algorithms on DTs with 50, 100, 200, 400, 600, 800, and 1000 nodes respectively. We conduct experiments on 1000 DTs for each node number.

Table 1. Average Euclidean deviation ratios of the five algorithms on DTs with different node numbers

	TwoStp	ApxAng	ComRtg	ComGdy	GdyRtg
50	1.0085	1.0083	1.0171	1.0529	1.0723
100	1.0112	1.0111	1.0190	1.0612	1.0818
200	1.0135	1.0137	1.0206	1.0688	1.0896
400	1.0155	1.0160	1.0219	1.0755	1.0964
600	1.0164	1.0171	1.0223	1.0786	1.0993
800	1.0171	1.0178	1.0226	1.0803	1.1009
1000	1.0175	1.0184	1.0229	1.0824	1.1029

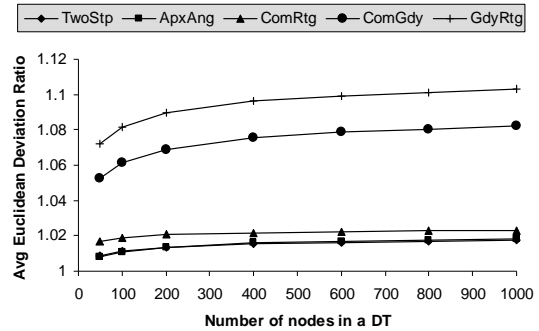


Figure 14. Average Euclidean deviation ratios of the five algorithms on DTs with different node numbers

Table 1 gives the average Euclidean deviation ratios for the five algorithms from those 1000 experiments, and Figure 14 plots Table 1. If our experiments are repeated by others, almost the same values as presented in Table 1 should be obtained. Thus, for the purpose of others' check, Table 1 is not omitted here. From Table 1 and Figure 14, we can conclude the following:

- For all these five algorithms, the Euclidean deviation ratios increase slowly with the node number, reflecting that when the DT has a larger node number, the paths found by these five algorithms deviate more from the actual shortest paths in Euclidean metric.
- For all node numbers of DTs evaluated, the performance rankings of these five algorithms are the same as the ranking presented in Figure 11, only except that TwoStp and ApxAng switch their ranks when the node number of DTs reaches 200.

Table 2. Average link deviation ratios of the five algorithms on DTs with different node numbers

	TwoStp	ApxAng	ComRtg	ComGdy	GdyRtg
50	1.2132	1.1915	1.1705	1.1332	1.1034
100	1.2737	1.2456	1.2249	1.1774	1.1447
200	1.3471	1.3126	1.2928	1.2343	1.1994
400	1.4390	1.3977	1.3796	1.3093	1.2722
600	1.5001	1.4548	1.4376	1.3597	1.3212
800	1.5479	1.4998	1.4835	1.4000	1.3605
1000	1.5928	1.5422	1.5264	1.4383	1.3980

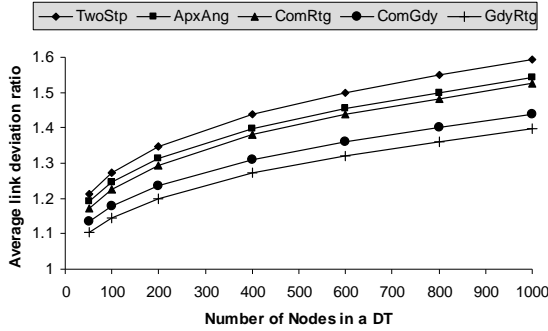


Figure 15. Average link deviation ratios of the five algorithms on DTs with different node numbers

Table 2 gives the average link deviation ratios for the five algorithms from those 1000 experiments, and Figure 15 plots Table 2. For the purpose of others' check, Table 2 is not omitted here. From Table 2 and Figure 15, we can conclude the following:

- For all these five algorithms, the link deviation ratio increases slowly with the node number, reflecting that when the DT has a larger node number, the paths found by these five algorithms deviate more from the actual shortest paths in link metric.
- For all node numbers of DTs evaluated, the performance rankings of these five algorithms are the same as the ranking presented in Figure 12.

5.4 The impact of the node density of DTs on the Euclidean and link deviation ratios

In the experiments for above evaluations, we set the node density to 10 nodes/km² when generating the DT nodes in a square area. For instance, if the number of nodes is 20, the side length of the square area is approximately 1414m. We also vary the node densities (e.g., using 50 nodes/km², 100 nodes/km²) to conduct the same set of experiments. Interestingly, we get almost identical experimental data as described above. This reflects that *DTs have the property that in both Euclidean metric and link metric, their node densities will not affect the ratios between the lengths of two different paths, if the link sets of these two different paths are not changed.*

Since the change of node density on the plane is a dilation transformation (i.e., stretching or shrinking with respect to a center point) on the DTs, the above property can be explained by the following fact: the dilation transformation in Euclidean space preserves the ratios between the distances of two line segments [20].

5.5 Discrepancy Ratio

Combining the conclusions obtained in the previous subsection, we see the tendency that when an MOR algorithm performs better in terms of Euclidean deviation ratio, it performs worse in terms of link deviation ratio, and vice versa. This leads to the conjecture that *in a DT with random node placement, the shortest paths in the Euclidean metric are discrepant with those in the link metric in a large proportion.* Thus, when an MOR algorithm performs well in finding the shortest paths in one metric, it fails in the other metric. To verify this conjecture, we measure the *discrepancy ratio* of the shortest paths in a DT. The discrepancy ratio is defined as the number of the (s, t) pairs for which the shortest paths under the Euclidean and link metrics are discrepant versus the total number of (s, t) pairs considered in a DT.

Note that, for both Euclidean and link metrics, the shortest paths for a (s, t) pair in DTs can have multiple instances with identical length. To be clear, we mean by 'discrepant' that no instance of the shortest paths in Euclidean metric has an identical link set with any instance of the shortest paths in link metric for (s, t) . In implementation, our way of determining whether these two kinds of shortest paths for (s, t) are discrepant is as follows. We first determine the multiple shortest paths in link metric for (s, t) , and then select the shortest one in Euclidean metric among them and denote its Euclidean length EL1. Next, we determine the multiple paths in Euclidean metric for (s, t) and denote their Euclidean length EL2. Finally, we examine whether EL1 equals EL2. If they are equal, it means that these two kinds of shortest paths can have the same link set for (s, t) ; otherwise, we deem these two kinds of shortest paths discrepant for (s, t) .

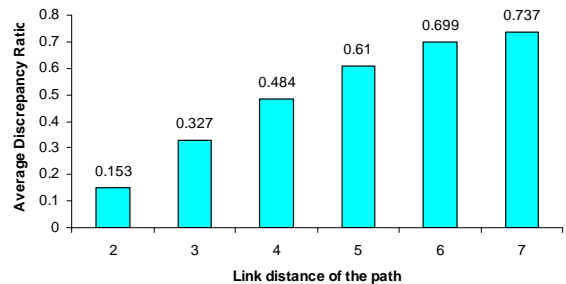


Figure 16. The discrepancy ratio of the shortest paths between link metric and Euclidean metric

We conduct experiments on 1000 DTs of 100 nodes, and measure the discrepancy ratios for the (s, t) pairs with the shortest link distances 2, 3, ..., 7 respectively.

Figure 16 plots the average values from these 1000 experiments. Note that we do not measure the discrepancy ratios for the (s, t) pairs with a link distance larger than 7, because such (s, t) pairs seldom exist for 100-node DTs. For your information, in our experiments, we obtain that the average link distance for all (s, t) pairs in 100-node DTs is about 4.24. From Figure 16, we can see the following:

- The discrepancy ratios are significantly large for all the evaluated link distances 2-7, ranging from 0.153 to 0.737, thus verifying our conjecture.
- The discrepancy ratio increases with the link distance of the paths, reflecting that the longer the path, the larger the possibility is for the shortest paths in these two metrics to be discrepant.

We also evaluate the impact of the node number of DTs on the discrepancy ratios. In this evaluation, we measure the discrepancy ratio for all the (s, t) pairs with link distances ≥ 2 in a DT, and the experiments are conducted on 1000 DTs of 50, 100, 200, 400, 600, 800, and 1000 nodes respectively. Figure 17 plots the average values obtained from those 1000 DTs for each node number, showing that the discrepancy ratio increases steadily with the growth of the node number of DTs.

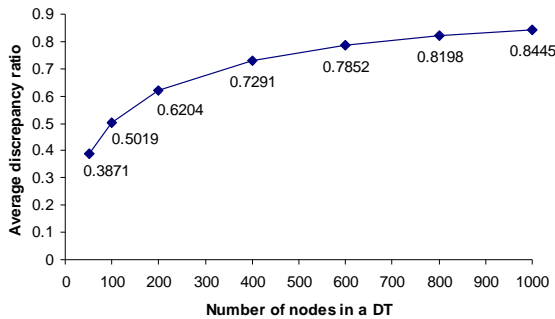


Figure 17. Average discrepancy ratios of DTs with different node numbers

6. OPEN PROBLEMS

This section raises two open problems based on the observations from our experiments. First, it was shown in [4] that there exist some DTs for which both Compass Routing and Greedy Routing are not c -competitive in the Euclidean metric. As observed in our experiments, both Two-step Routing and Apex-angle Routing perform better than Compass Routing and Greedy Routing in Euclidean metric. Moreover, the Two-step Routing and the Apex-angle Routing will not fall in the zig-zag paths that are constructed in [4] to make Compass Routing and

Greedy Routing not c -competitive. Hence, we conjecture that *the Two-step Routing and the Apex-angle Routing are c -competitive for all DTs*. Up to now, no MOR algorithms have been proved to be c -competitive for all DTs. If this conjecture is proved true, more meanings will be added to these two algorithms.

Second, our experiments show that in DTs, the shortest paths in Euclidean metric are discrepant with those in link metric in a large proportion. So we see that DTs are not suitable if a routing algorithm desires that the path found in a graph be shortest in both Euclidean and link metrics. Thus, the following problem can be meaningful: *what kind of graphs has the property that the shortest paths in Euclidean metric and in link metric are identical for any pair of nodes?* For example, we know that the grid graph (composed of rectangles) and the hexagonal graph (composed of regular hexagons) possess such a property. Even more, *can we give a low-complexity testing method for this property?*

7. CONCLUSIONS

In this paper, we reported and proved two new MOR algorithms that work for DTs: the Two-step Routing algorithm and the Apex-angle Routing algorithm. We also evaluated and compared these two algorithms with other three existing MOR algorithms discussed in related work. Our experimental results mainly show the following.

- All the five algorithms can find paths with low Euclidean and link deviation ratios on DTs in average and most cases, so they are practical for applications.
- All the five algorithms perform better in Euclidean deviation ratio than in link deviation ratio, reflecting that the structure of DTs helps more for the MOR algorithms to reduce the Euclidean deviation ratio than to reduce the link deviation ratio.
- For all the five algorithms, the Euclidean and link deviation ratios increase with the growth of the node number of DTs.
- DTs have the property that their node densities do not affect the Euclidean and link deviation ratios achieved by a routing algorithm.
- DTs with random node placement have the property that the shortest paths in link metric and in Euclidean metric are discrepant in a large proportion.

Furthermore, we have the following three remarks on our work. First, since the MOR algorithms are interesting due to its simplicity and elegance, we feel

that the key contribution of our work lies in enriching the family of MOR algorithms that works for DTs. Second, our evaluation is not intended to show that our new algorithms perform better than the existing algorithms, but to shed light on the performance of this family of MOR algorithms. Since we have shown that the shortest paths in Euclidean metric and link metric are highly discrepant in DTs, and also because a given MOR algorithm can only find a single path for a (s, t) pair, it is not possible for an MOR algorithm to perform the best in both Euclidean metric and link metric among this family of algorithms. Third, although the algorithms and proofs given by our work are simple, they possess an elementary nature, so they help gaining insight into the MOR algorithms as well as the DTs, and also spark further research.

ACKNOWLEDGMENTS

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

REFERENCES

- [1] K. Alzoubi, X. Y. Li, Y. Wang, P. J. Wan, and O. Frieder, "Geometric spanners for wireless ad hoc networks", *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, pp. 408-421, 2003.
- [2] M. d. Berg, M. v. Kreveld, M. Overmars, and O. Schwarzkopf, *Computational geometry: algorithms and applications*, 2nd ed. New York: Springer, 2000.
- [3] 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.
- [4] P. Bose and P. Morin, "Online Routing in Triangulations", *Siam Journal of Computing*, vol. 33, pp. 937-951, 2004.
- [5] P. Bose, P. Carmi, and S. Durocher, "Bounding the Locality of Distributed Routing Algorithms", in *The 28th ACM symposium on principles of distributed computing (PODC) 2009*, pp. 250-259.
- [6] 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.
- [7] D. Chen, L. Devroye, V. Dujmovic, and P. Morin, "Memoryless Routing in Convex Subdivisions: Random Walks are Optimal", <http://arxiv.org/abs/0911.2484>, 2009.
- [8] P. Cucka, N. S. Netanyahu, and A. Rosenfeld, "Learning in navigation: Goal finding in graphs", *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 10, pp. 429-446, 1996.
- [9] S. Cui, I. A. Kanj, and G. Xia, "On the dilation of Delaunay triangulations of points in convex position", in *Canadian Conference on Computational Geometry*, 2009, pp. 51-54.
- [10] D. P. Dobkin, S. J. Friedman, and K. J. Supowit, "Delaunay graphs are almost as good as complete graphs", *Discrete and Computational Geometry*, vol. 5, pp. 399-407, 1990.
- [11] 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.
- [12] 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.
- [13] E. Kranakis, H. Singh, and J. Urrutia, "Compass routing on geometric networks", in *Canadian Conference on Computational Geometry*, 1999, pp. 51-54.
- [14] 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.
- [15] 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.
- [16] J. O'Rourke, *Computational Geometry in C*, 2nd ed. Cambridge, UK: Cambridge University Press, 1998.
- [17] J. R. Shewchuk, *Triangle version 1.6*, Available: <http://www.cs.cmu.edu/~quake/triangle.html>, 2005.
- [18] 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.
- [19] 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.
- [20] WolframMathworld, *Dilation transformation*, Available: <http://mathworld.wolfram.com/Dilation.html>, 2010.