# A new memroyless online routing algorithm for Delaunay triangulations

Ashkan Rezazadeh
Computer Engineering Department
Ferdowsi University Of Mashhad
Mashhad, Iran
Email: ashkan.rezazadeh@mail.um.ac.ir

Mostafa Nouri-Baygi
Computer Engineering Department
Ferdowsi University Of Mashhad
Mashhad, Iran
Email: nouribaygi@um.ac.ir

***Abstract*—We consider 1-*local* online routing on a special class of geometric graphs called *Delaunay triangulations* (DTs). A geometric graph *G=(V,E)* of a point set consists of a set of points in the plane and edges between them, where each edge weighs as the Euclidean distance between it's end-points. DTs are one of the useful classes of these graphs because of some good properties which can help during the navigation process, therefore over the years DTs have been widely proposed as network topologies for several times.**

**In this paper, we present an *MOR*[1] algorithm for DTs which is simple, elegant, and easy to implement, while having an acceptable performance.**

**The set of MOR algorithms are suitable for cases where we want to find a path using only local information, our proposed algorithm is memoryless or 1-*local*, in $k$-local routing, we find a path between a source vertex $s$ to a destination vertex $t$ while our knowledge at each step is limited to the locations of $s$ and $t$, the location of current vertex and it's $k$-neighborhood vertices.**

**We also evaluate and compare the perforamnce of our prpopsed algorithm with existing MOR algorithms. Our experimental results implied that our proposed algorithm has an acceptable performance in both Euclidean and link metrics and it outperforms all of the existing MOR algorithms in Euclidean metric, and some of them in the link metric as well. Finally, we pose two open problems to solve in the future.**

***Keywords*— Online routing, Delaunay triangulation, Geometric routing, Memoryless online routing**

## I. Introduction

Finding a path between a given source vertex $s$ to a destination vertex $t$ in a graph is a well-known and fundamental problem in computer science, it is central to several fields like urban planning, robotics, and communication systems.

In most of the cases, we have a geometric setting that can be modeled by a geometric graph, so the problem becomes the problem of finding a path from a source vertex $s$ to a destination vertex $t$ in a geometric graph $G$. Note that in geometric graphs, each vertex is identified by it's coordinates in the plane.

We focus on a particular and useful class of geometric graphs called *Delaunay triangulation*(DT). DTs are widely used in scientific computing in many diverse applications. While there are numerous algorithms for computing triangulations, it is the favorable geometric properties of DTs that make them so useful.

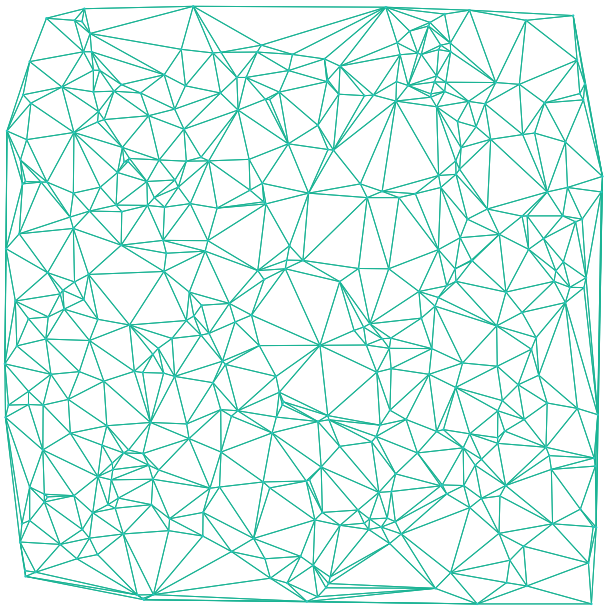

**Fig. 1:** *An example DT with 400 nodes*

The DT is the dual graph of the *Voronoi* diagram. The DT of a set of points $P$ in the plane is a triangulation of $P$ so that there are no points of $P$ in the interior of the circumcircle of any of its triangular faces. An example DT of 400 points in the plane is illustrated in Fig. 1.

DTs have some good properties which help during navigation process and because of that, they have been proposed as the network topologies several times [1], [2], [3]. Some of those properties are listed below:

1) DTs are planar graphs.
2) For any triangulation, the number of edges is equal to $3n - 3 - k$ [4], while $n$ denotes the number of nodes and $k$ denotes the number of convex hull edges.
3) If the length of the shortest path between any two node in a graph $G$ is no longer than $c$ times the Euclidean distance between those two nodes, $G$ is a $c$-spanner and $c$ is it's spanning ratio, moreover, If $G$ is a geometric graph, then it is a geometric spanner. Dobkin *et al.* [5]

[1]Memoryless online routing

proved that DTs are geometric spanners, and later it has been proved that $c$ is between $1.5846$ and $1.998$ for DTs.[6], [7]

When the full knowledge of the underlying network topology is available beforehand, there exists numerous routing algorithms which can find the shortest paths in a graph, but the problem is more challenging when we try to find a route in the *Online* settings. Online routing is also called local routing which means that the full knowledge of the underlying network topology is not available beforehand and the robot/message should explore the graph as it tries to find a route to the destination using only local information available.

Bose and Morin [8] have classified online routing algorithms based on their use of memory and/or randomization, they call a routing algorithm *memoryless* if the desicion of selecting the next forwarding node is made according to the loation of the current vertex, it's 1-hop neighbors and the location of destination vertex.

The set of MOR algorithms are very useful in many scenarios like communication systems or robotics where the whole network topology is not known beforehand. The MOR algorithms are simple, elegant, scalable and energy efficient. Therefore, they have received a lot of attention in literature. [8], [9], [10], [11]

We say an online algorithm $A$ *works* for a graph $G$, if it can find a path between any two vertice of $G$.

In this paper we present a new MOR algorithm that works for DTs, our experimental results implied that it outperforms the existing MOR algorithms in Euclidean metric and most of them in link metric. Our main idea is based on the combination of two of the MOR algorithms presented by Kranakis [12], and Si and Zomaya [11].

We have organized the paper as follows. Section 2 discusses related work; Section 3 will explain our new MOR algorithm; In section 4 the experimental results will be presented and we will evaluate our algorithm's performance by comparing it with existing MOR algorithms in two common metrics(i.e. Euclidean and Link metric); finally, In section 5 we will conclude the paper and discusses about some open problems and future works.

## II. Related Work

Let us begin by introducing some notations. We will use $s$ to denote the source, $t$ to denote the destination, $p$ to denote current forwarding node, $N(p)$ to denote the 1-hop neighbors of $p$, $d(u, v)$ to denote the Euclidean distance between vertex $u$ and vertex $v$, and $\angle uvw$ to denote the angle between the line segment $vu$ and $vw$ which is less than or equal $180°$.

If the length of the path produced by an algorithm $A$ on a graph $G$ from any source vertex $s$ to any destination vertex $t$ is no longer than $c$ times the shortest path from $s$ to $t$ in $G$, then $A$ is a $c$-competitive algorithm and $c$ is it's competitive ratio.

To the best of our knowledge, There exist seven MOR and two $1 - local$ online routing schemes that have been proved to work for DTs to this date. We briefly describe them below.

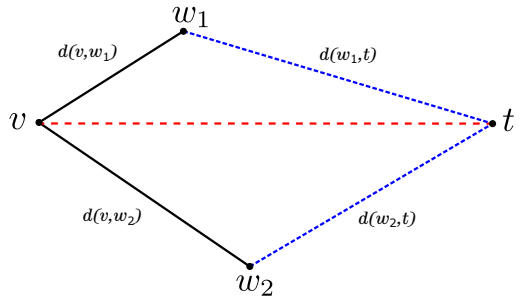


**Fig. 2:** *The basic idea of Two-step Routing. In this example $d(v, w_1) + d(w_1, t)$ is smaller than $d(v, w_2) + d(w_2, t)$, hence, the node $v$ will forward the packet to the node $w_1$.*

The first seven algorithms are in the set of MOR algorithms and their competitiveness has not been proved yet, while the others are competitive $1 - local$ online routing algorithms.

1) **Greedy routing algorithm** [8]
   In Greedy routing, the node $w \in N(p)$ which has the smallest $d(w, t)$ will be selected as the next forwarding node and a tie is broken arbitrarily.
2) **Compass routing algorithm** [12]
   In Compass routing, the node $w \in N(p)$ which minimizes $\angle tpw$ will be selected as the next forwarding node and a tie is broken arbitrarily.
3) **Greedy-Compass routing algorithm**[13]
   $cw(p)$ denotes the node $w \in N(p)$ which minimzes $\angle tpw$ clockwise from the line $pt$, and $ccw(p)$ denotes the node $w \in N(p)$ which minimzes $\angle tpw$ counter-clockwise from the line $pt$. If a node $w \in N(p)$ lies on the line segment $pt$, then we have $cw(p) = ccw(p) = w$. In Greedy-Compass routing, $p$ first finds the two nodes $cw(p)$ and $ccw(p)$, then the node $w \in \{cw(p), ccw(p)\}$ which minimizes $d(w, t)$ will be selected as the next forwarding node, and a tie is broken arbitrarily.
4) **Two-step routing algorithm** [11]
   In Two-step routing, the node $w \in N(p)$ which has a smaller $d(w, t)$ than $d(p, t)$ and minimzes $d(p, w) + d(w, t)$ will be selected as the next forwarding node, and a tie is broken arbitrarily. Because of the condition of $d(w, t) < d(p, t)$, the robot/packet is guaranteed to reduce it's distance to $t$ in each step and reach $t$ at the end. Fig. 2 illustrates the main idea of Two-step routing, and Fig. 3 gives us further details about how the algorithm works.
5) **Apex-angle routing algorithm** [11]
   In Apex-angle routing, the node $w \in N(p)$ which has a smaller $d(w, t)$ than $d(p, t)$ and maximizes $\angle pwt$ (apex angle) will be selected as the next forwarding node, and a tie is broken arbitrarily.
   Using Lemma 2 in [11] can helps us to prove that $p$ has at least one neighbor $w$ with $d(w, t) < d(p, t)$, Si and Zomaya [11] used this to prove that Apex-angle routing algorithms works for DTs.
6) **Distance Referencing algorithms** [11]
   If we consider a point in *Cartesian* coordinate system

**Algorithm 1** The Two-step routing

```
for all w ∈ N(p) do
  if w has a smaller d(w,t) than d(p,t) then
    sum ← d(p,w) + d(w,t)
    if w has a smaller sum than previous neighbors then
      next(p) ← w
    end if
  end if
end for
```

**Fig. 3:** *The Two-step routing*

as a vector from the origin to the point, then we can express any point $l$ on the line segment $pt$ by:

$l = (1-\alpha)\cdot p + \alpha \cdot t, \alpha \in R$ (the set of real numbers).
Si and Zomaya [11] called $l$ "the reference point", and $\alpha$ "the distance coefficient". It has been shown in [11] that when $\alpha < 1/2$ or when $\alpha > 3/2$, the set of Distance Referencing algorithms do not work for DTs. Theorem 4 in [11] implies that when $1/2 \leq \alpha \leq 1$, the Distance Referencing algorithms work for DTs.
As you can see, any selection of $\alpha \in R$ gives us an MOR algorithm. In the set of Distance Referencing algorithms, we try to minimize the distance to the point $l$ at each step, hence, the node $w \in N(p)$ which minimizes the distance to the point $l$ will be selected as the next forwarding node, and a tie is broken arbitrarily. For example, when $\alpha = 1/2$, we get the Midpoint algorithm [11] and when $\alpha = 1$, we get the Greedy routing algorithm [8].

7) **Deterministic Compass algorithms** [11]
The Greedy-Compass routing algorithms is a special case of Deterministic Compass algorithms, Thus, this set of algorithms have a similar approach.
In this set of MOR algorithms, If $p$ does not have a neighbor $w$ that lies on the line segment $pt$, the node $p$ first finds the two node $cw(p)$ and $ccw(p)$, then will select one of them as the next forwarding node based on a deterministic rule, otherwise, $cw(p) = ccw(p) = w$, hence, $w$ will be selected directly as the next forwarding node. The deterministic rule that we mentioned above can be any rule that always chooses the same vertex in the same state. For example, if the deterministic rule is to select the node $w$ with the smallest $d(w,t)$, we will get the Greedy-Compass routing algorithm.
Si and Zomaya [11] presented the Compass Midpoint routing algorithm by selecting the node $w \in \{cw(p), ccw(p)\}$ which minimizes $d(w,m)$ as the deterministic rule, while $m$ is the midpoint of the line segment $pt$.
8) **Chew's routing algorithm on DTs** [14]
The empty region in the definition of DT that we mentioned above is a $circle$, if we replace it with $square$, the triangulation is called $L_1$-Delaunay triangulation.
If the length of any path founded by an algorithm $A$ between any two vertices is no longer than $c$ times the Euclidean distance between those two vertices, $c$ is the routing ratio of $A$.
Bonichon *et al.* [14] introduced a generalization of the Chew's deterministic 1-local routing algorithm on the $L_1$-Delaunay triangulation [15] which works on DTs with competitive and routing ratios of 5.90.
In this algorithm, we consider only the sequence of triangles that intersect the line segment $st$. $T_i$ denotes the rightmost triangle that intersects line segment $st$ and has $p$ as a vertex, $C_i$ denotes the circle circumscribing $T_i$, $w_i$ denotes the leftmost point of $C_i$, $r_i$ denotes the rightmost intersection of $C_i$ with $st$, and $x$ and $y$ denotes the other two vertices of $T_i$.
In this algorithm, the line segment $w_i r_i$ splits $C_i$ in two arcs, if $p$ lies on the upper arc we will walk clockwise on $C_i$ to reach $x$, otherwise we walk counterclockwise on $C_i$ to reach $y$, we repeat these steps until we reach $t$.
9) **MixedChordArc routing algorithm** [16]
Bonichon*et al.* [16] presented a 1-local routing algorithm on the Delaunay triangulation with a routing ratio of 3.56, improving the previous algorithm with a routing ratio of 5.9.
Note that by the routing ratio is an upper bound on the competitive ratio of a graph $G$ for any routing algorithm.
Like the previous algorithm, we find a route between a source vertex $s$ to a destination vertex $t$ along the edges of triangles that intersect the line segment $st$ using only local knowledge and the location of $s$ and $t$.
Let $T$ be the rightmost triangle that intersects line segment $st$ and has $p$ as a vertex. Let $C$ be the circumcircle of $T$, $t_C$ the rightmost intersection of $C$ on $st$, $u$ and $v$ be arbitrary points on $C$. Then $A_C(u,v)$ denotes the clockwise arc of $C$ from $u$ to $v$ and $B_C(u,v)$ denotes the counterclockwise arc of $C$ from $s$ to $t$, $q \neq p$ denotes the vertex of $T$ which is below the line segment $st$, and $r$ denotes the other vertex of $T$.
The MixedChordArc algorithm works as follows. If $p = s$, If the center of $C$ is on or below the line segment $st$, we select $r$, otherwise we select $q$ as the next forwarding node. If $p \neq s$ we repeat the following steps until we reach $t$:
If $|A_C(p,t_C)| \leq |pq| + |B_C(q,t_C)|$ we select $r$ as the next forwarding node, otherwise we set $p = q$.

## III. Two-step-Compass Routing algorithm

The experimental results and comparisons in [11] showed that the Two-step routing has the best performance when we rate algorithms by the Euclidean length of the paths founded by an algorithm. Therefore, we tried to focus on improving the performance of this algorithm while keeping simplicity. It is woth noting that analyzing the experimental results in [11]

**Algorithm 2** Two-step-Compass routing

```
for all w ∈ N(p) do
  if w has a smaller d(w,t) than d(p,t) then
    sum ← (α · (d(p,w) + d(w,t))) + (β · ∠tpw)
    if w has a smaller sum than previous neighbors then
      next(p) ← w
    end if
  end if
end for
```

**Fig. 4:** *Two-step-Compass routing*

shows that an algorithm's performance in Euclidean metric is in contrary with it's performance in link metric, hence, the Two-step routing performs worse in link metric.

In most scenarios like robotics, reducing the Euclidean length of the path is our main goal, while in some other scenarios like WSNs[2], it is more important to reduce the number of hops in a path.

Some of the algorithms presented in [11] use the Euclidean distance, some others use angles, and some uses the combination of the two to decide the next forwarding node. Our algorithm will fit into the latter set, combining these two to improve the performance of Two-step in both metrics.

Our algorithm is an MOR 1-local routing algorithm for DTs based on two of MOR algorithms we mentioned in the previous section: Two-step [17] and Compass [12] routing algorithms. Compass Routing outperforms Two-step in link metric while Two-step outperforms Compass in the Euclidean metric. Hence, combining them results an algorithm that has an acceptable performance on both metrics, specially in Euclidean metric.

We call the algorithm "$Two\text{-}step\text{-}Compass$" as it combines the Two-step and Compass routing scheme while adding some extra parameters so it can perform well in different scenarios.

Two-step-Compass has all the benefits of the set of MOR algorithms and works as follows: At each step, the node $w \in N(p)$ which has a smaller $d(w,t)$ than $d(p,t)$ and minimizes

$$(\alpha \cdot (d(p,w) + d(w,t))) + (\beta \cdot \angle tpw)$$

will be selected as the next forwarding node.

$\alpha$, $\beta \in [0,1]$ and $\angle tpw$ is measured in degrees. It is clear that the selection of $\alpha$ and $\beta$ is effective on the performance of the algorithm. In cases that we want to reduce the Euclidean length of the path, $\alpha$ should be increased and be higher than $\beta$, while in cases that we want to reduce the number of hops of the path, $\beta$ should be increased and be higher that $\alpha$.

Fig. 4 illustrates the details of the algorithm in form of a pseudo-code.

It has been proven in [17], [8] that the Two-step and Compass routing algorithms works for DTs and since our main idea is based on Two-step and can be considered as a combination of the Two-step and Compass routing algorithms, we can conclude that it also works for DTs.

[2]Wireless sensor network

## IV. Evaluation

A good metric for the path quality is *stretch*. The *stretch* by a routing algorithm $A$ for any pair of vertices $(u,v)$ in a graph $G$ is defined by the ratio of the length of the path found by $A$ from $u$ to $v$ to the shortest path from $u$ to $v$ which can be found by an offline algorithm like Dijkstra.

We use *average_stretch* as our metric for the path quality in order to evaluate and compare the performance of the routing algorithms. It is clear that the *average_stretch* by a routing algorithm $A$ on a geometric graph $G$ is defined by the average of the stretches of all $(u,v)$ pairs in $G$ by $A$.

The two common path length metrics that we have used are listed below:

1) Euclidean distance:
   Defined as the sum of all Euclidean distances of all edges traversed in a path.
2) Link distance:
   Defined as the number of hops in a path.

Note that in some scenarios like robotics, the smaller Euclidean distance is more important , while it is more important to reduce the number of hops in a path in some other cases like WSNs, therefore evaluating the performance of routing algorithms in these two metric can helps us to choose the better option in each case.

We used these five routing algorithms with our proposed algorithm in our experiments for comparing the performance of them in terms of *average_stretch* in both Euclidean and link metric:

- Greedy
- Midpoint
- Two-step
- Apex-angle
- MixedChordArc

The reasons that made us use these algorithms in our experiments in order to evaluate the performance of Two-step-Compass are as follows:

1) The experimental results presented in [11] showed that in terms of *average_stretch* and *max_stretch*, Two-step and Apex-angle routing algorithms have the best performance in the Euclidean metric, and in the link metric, Greedy and Midpoint routing algorithms performs better than the others.
2) The competitiveness of the algorithms presented by Si and Zomaya [11] have not been proven yet, hence, we have added the MixedChordArc [16] routing to our experiments, since it has the smallest routing ratio(3.56) on DTs to this date.

### A. Experiment setup

We have generated 100 random DTs with 100, 200, 400, 600, 800, and 1000 nodes which were distributed uniformly in a square area, then we implemented the algorithms in a

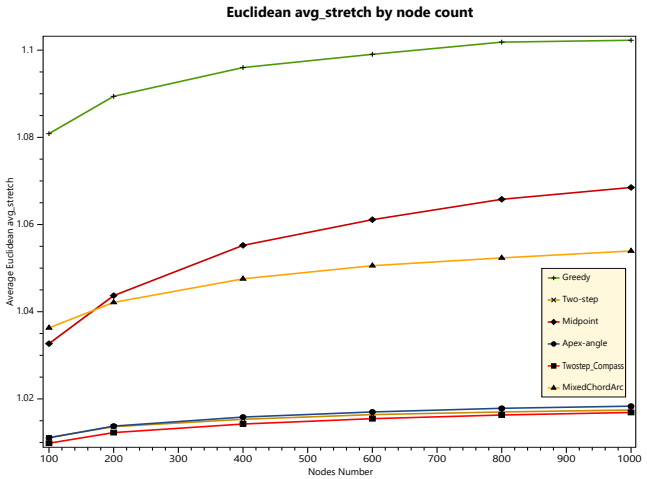


**Fig. 5:** *The average Euclidean average_stretches of the algorithms*

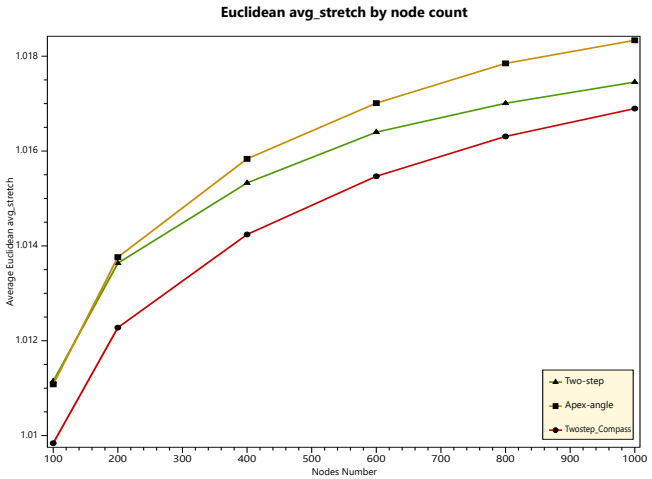


**Fig. 6:** *The average Euclidean average_stretches of the Two-step, Apex-angle, and Two-step-Compass*

program and run each of them on every DT. You can see one of our randomly generated DTs with 400 vertices which used in our experiments in figure 1.

We run the Two-step-Compass routing algorithm while setting $\alpha = \beta = 1$ in our experiments, but they can be changed in order to get the desired result base on differenct cases.

As mentioned before, we use *average_stretch* in order to evaluate and compare the quality of paths found by algorithms in both Euclidean and link metric.

In order to calculate the *stretch* and *average_stretch*, we need to have the shortest path between any two vertice in the graph, therefore, we used BFS and the famous Dijkstra's shortest path algorithm to get the shortest paths in link and Euclidean metrics, respectively.

### *B. Experimental results*

In this section, we will present the results of our experiments in order to evaluate and compare the performance of the algorithms in terms of *average_stretch* on DTs.

We can see the average Euclidean *average_stretch* of the algorithms in Fig. 5.

As you can see, our experimental results imply that while maintaining simplicity and elegance of the MOR algorithms, the Two-step-Compass routing outperforms the other algorithms in terms of average Euclidean *average_stretch*.

We can take a closer look to the comparison between these three algorithms in Fig. 6.

The average link *average_stretch* of the algorithms is illustrated in Fig. 7. It can be understood that although the Two-step-Compass has the best performance in Euclidean metric, it's performance in link metric is not as good as the Greedy, Midpoint, and Apex-angle routing algorithms.

We can summarize our experimental results as follows:

- The Two-step-Compass routing outperforms the other algorithms in Euclidean metric.
- In Euclidean metric, the Two-step-Compass routing performs better than both Two-step and Apex-angle which had the best performance in Euclidean metric to this date.

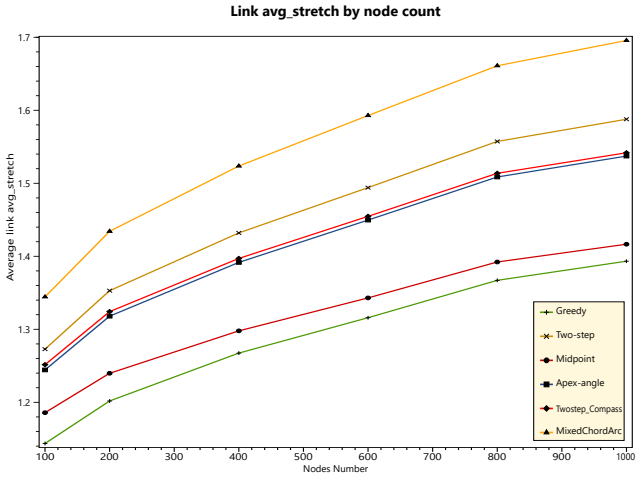


**Fig. 7:** *The average link average_stretches of the algorithms*

- The Greedy, Midpoint, and Apex-angle routing algorithms performs better than the Two-step-Compass in link metric.
- the Greedy routing algorithm outperforms the other algorithms in link metric.
- The Two-step-Compass routing algorithm improves the performance of both Two-step and MixedChordArc in both Euclidean and link metrics.
- The performance of the Two-step-Compass routing is acceptable in both metrics on DTs, it's *average_stretch* is below 1.017 in Euclidean metric, and below 1.55 in link metric, therefore, it can be applicable in various fields.
- The Two-step-Compass performs better in Euclidean metric, hence it can be more suitable choice for scenarios like robotics, where reducing the Euclidean distance is more important.

Table I and II gives us the percentage of improvement of each algorithm by the Two-step-Compass in Euclidean and link metrics, respectively.

| Algorithm \ Nodes No. | 100 | 200 | 400 | 600 | 800 | 1000 |
|---|---|---|---|---|---|---|
| Greedy | 87.82 | 86.26 | 85.16 | 84.38 | 83.98 | 83.47 |
| Midpoint | 69.89 | 71.90 | 74.21 | 74.68 | 75.21 | 75.33 |
| Two-step | 11.72 | 9.96 | 7.09 | 5.67 | 4.11 | 3.20 |
| Apex-angle | 11.20 | 10.79 | 10.07 | 9.06 | 8.63 | 7.85 |
| MixedChordArc | 72.89 | 70.86 | 70.04 | 69.40 | 68.8 | 68.67 |

**TABLE I:** *The percentage of improvement of the algorithms by the Two-step-Compass routing in Euclidean metric.*

| Algorithm \ Nodes No. | 100 | 200 | 400 | 600 | 800 | 1000 |
|---|---|---|---|---|---|---|
| Greedy | -75.17 | -60.62 | -48.48 | -43.94 | -39.94 | -37.75 |
| Midpoint | -35.38 | -35.20 | -33.30 | -32.56 | -30.97 | -30.06 |
| Two-step | 7.79 | 8.13 | 8.08 | 7.96 | 7.82 | 7.81 |
| Apex-angle | -2.94 | -1.95 | -1.38 | -1.13 | -0.99 | -0.84 |
| MixedChordArc | 26.96 | 25.34 | 24.17 | 23.29 | 22.27 | 22.11 |

**TABLE II:** *The percentage of improvement of the algorithms by the Two-step-Compass routing in link metric. It is clear that a negative value indicates deterioration of performance by Two-step-Compass.*

## V. Open Problems

In this section we will discuss two open problems which can be solved as future work:

- It remains to be seen whether the Two-step-Compass routing algorithm is *c-competitive* or not. Bose and Morin [8] proved that The Greedy and Compass routing algorithms are not c-competitive in Euclidean metric by running these two algorithm on a zig-zag triangulation.
  Although that the four algorithm presented in [11], as well as our algorithm will not trap in a zig-zag triangulation like the one constructed in [8] to prove that the Greedy and Compass algorithms are not *c-competitive*, it has not been proven yet wether these algorithms are *c-competitive* or not.
- The experimental results presented by Si and Zomaya [11] implied that the performance order of the algorithms in Euclidean metric is reversed in link metric, but our proposed algorithm is in the 1st and 3rd rank in the Euclidean and link metric, respectively, which means that it has improved the performance of two algorithms in both metrics, so the question is, *can we design an MOR algorithm that outperforms other MOR algorithms in both Euclidean and link metric?*

## VI. Conclusion

In this paper, we preseneted an MOR algorithm that will find a path from any source vertex $s$ to any destination vertex $t$ on any graph $G$ of a special class of geometric graphs, called Delaunay triangulations. This algorithm is inspired by two of the previous presented MOR algorithms, the Two-step [17] and Compass [12], hence we call it "Two-step-Compass".

We also evaluated our proposed algorithm by compare it's performance with four existing MOR algorithms (Greedy, Midpoint, Two-step, and Apex-angle) and an $O(1)$-memory competitive routing algorithm (MixedChordArc) in terms of the average Euclidean and link *average_stretches*. Our experimental results implied that while maintaining simplicity, the Two-step-Compass routing improves the performance of previous MOR algorithms in Euclidean metric, but not all of them in link metric, so it can be practically used in different scenarios.

Based on our experiments, the Two-step-Compass routing algorithm performs well in both Euclidean and link metric in terms of average $average\_stretch$, but it performs better in Euclidean metric, therefore it can be more suitable choice in scenarios that we would like to reduce the Euclidean length of the path rather than the number of hops.

## References


[1] M. B. Haider and K. Sugihara, "Almost delaunay triangulation routing in wireless sensor networks," in *2007 10th international conference on computer and information technology*, Dec 2007, pp. 1–7.

[2] Jie Gao, L. J. Guibas, J. Hershberger, Li Zhang, and An Zhu, "Geometric spanners for routing in mobile networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 1, pp. 174–185, Jan 2005.

[3] D. Satyanarayana, R. SV *et al.*, "Constrained delaunay triangulation for ad hoc networks," *Journal of Computer Systems, Networks, and Communications*, vol. 2008, 2008.

[4] M. Berg, M. Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications, second ed.Springer*, 2000.

[5] D. P. Dobkin, S. J. Friedman, and K. J. Supowit, "Delaunay graphs are almost as good as complete graphs," *Discrete & Computational Geometry*, vol. 5, no. 4, pp. 399–407, 1990.

[6] P. Bose, L. Devroye, M. Löffler, J. Snoeyink, and V. Verma, "The spanning ratio of the delaunay triangulation is greater than pi/2," 01 2009, pp. 165–167.

[7] G. Xia, "The stretch factor of the delaunay triangulation is less than 1.998," *SIAM Journal on Computing (SICOMP)*, vol. 42, no. 4, pp. 1620–1659, 2013.

[8] P. Bose and P. Morin, "Online routing in triangulations," *SIAM Journal on Computing (SICOMP)*, vol. 33, no. 4, pp. 937–951, 2004.

[9] P. Bose, P. Carmi, and S. Durocher, "Bounding the locality of distributed routing algorithms," in *Proceedings of the 28th ACM Symposium on Principles of Distributed Computing*, ser. PODC '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 250–259. [Online]. Available: https://doi.org/10.1145/1582716.1582756

[10] D. Chen, L. Devroye, V. Dujmovic, and P. Morin, "Memoryless routing in convex subdivisions: Random walks are optimal," *CoRR*, vol. abs/0911.2484, 2009. [Online]. Available: http://arxiv.org/abs/0911.2484

[11] W. Si and A. Y. Zomaya, "New memoryless online routing algorithms for delaunay triangulations," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 8, 2012.

[12] E. Kranakis, H. Singh, and J. Urrutia, "Compass routing on geometric networks," in *Proceedings of the Canadian Conference on Computational Geometry (CCCG)*, Conference Proceedings, pp. 51–54.

[13] P. Bose, A. Brodnik, S. Carlsson, E. D. Demaine, R. Fleischer, A. López-Ortiz, P. Morin, and J. I. Munro, "Online routing in convex subdivisions," *International Journal of Computational Geometry & Applications*, vol. 12, no. 4, pp. 283–296, 2002.

[14] N. Bonichon, P. Bose, J.-L. D. Carufel, L. Perković, and A. v. Renssen, "Upper and lower bounds for online routing on delaunay triangulations," *In: Bansal N., Finocchi I. (eds) Algorithms - (ESA), Lecture Notes in Computer Science (Springer)*, vol. 9294, pp. 203–214, 2015.

[15] L. P. Chew, "There is a planar graph almost as good as the complete graph," in *Proceedings of 2nd Annual Symposium on Computational Geometry (SOCG)*, Conference Proceedings, pp. 169–177.

[16] N. Bonichon, P. Bose, J. D. Carufel, V. Despré, D. Hill, and M. H. M. Smid, "Improved routing on the delaunay triangulation," *In 26th Annual European Symposium on Algorithms (ESA)*, p. 22:1–22:13, 2018.

[17] W. Si, A. Y. Zomaya, and S. Selvakennedy, "A geometric deployment and routing scheme for directional wireless mesh networks," *IEEE Transactions on Computers*, vol. 63, no. 6, pp. 1323–1335, June 2011.