How Well Do Yao Graph and Theta Graph Support Greedy Forwarding?

Weisheng Si [1], Quincy Tse [1,3], Guoqiang Mao [2,4], Albert Zomaya [3]
School of Computing, Engineering and Mathematics, University of Western Sydney, Australia [1]
School of Computing and Communications, University of Technology Sydney, Australia [2]
School of IT, University of Sydney, Australia [3]
National ICT Australia Ltd., Australia [4]
w.si@uws.edu.au, guoqiang.mao@uts.edu.au, (quincy.tse, albert.zomaya)@sydney.edu.au

***Abstract*— Greedy Forwarding algorithm is a widely-used routing algorithm for wireless networks. However, it can fail if the wireless network topologies contain *voids*, where a packet cannot be moved closer to destination. Since Yao graph and Theta graph are two types of geometric graphs exploited to construct wireless network topologies, this paper firstly studied whether these two types of graphs can contain voids, showing that when the number of cones in a Yao graph or Theta graph is less than six, Yao graph and Theta graph can have voids, and when the number of cones equals or exceeds six, Yao graph and Theta graph are free of voids. Secondly, this paper experimented on how well Greedy Forwarding is supported on Yao graphs and Theta graphs in terms of *stretch*, i.e., the ratio between the path length found by Greedy Forwarding and the shortest path length in a graph. The experiments also included comparison with the stretch on Delaunay triangulation, another well-known geometric graph exploited in constructing wireless networks. Overall, our experiments revealed several interesting results.**

***Index Terms*— geometric routing, greedy forwarding, void, Yao graph, Theta graph, Delaunay triangulation.**

I. INTRODUCTION

In the study of geometric (or geographic) routing [1] problems, a wireless network is modelled by a geometric graph $G(V, E)$, in which each node in V is assigned a pair of (x, y)-coordinates, and each edge in E represents a connection between two nodes and has a weight equal to the Euclidean distance between these two nodes. An important routing algorithm under the above geometric graph model is the Greedy Forwarding algorithm [2]: when a node u forwards a packet with destination node t, u sends this packet to its *neighbor* that has the smallest Euclidean distance to t. Here, two nodes u and v are said to be each other's *neighbor* if and only if the edge uv is present in the graph.

However, Greedy Forwarding does not succeed on a graph that contains the *void* [2], in which for a certain destination t, a node does not have a neighbor with a smaller distance to t than its own distance to t. Thus, whether a geometric graph contains voids becomes an important property to study. To the convenience of study, we formally define the concept *void-free* as follows. Let $d(a, b)$ denote the Euclidean distance between node a and node b; if for any node pair (u, v) in a geometric graph G, u always has a neighbor w such that $d(w, v) < d(u, v)$, G is said to be *void-free*.

The void-free property has been studied for several types of geometric graphs used in wireless networks such as Relative Neighborhood Graph [3], Gabriel Graph [4], and Delaunay Triangulation [5]. Specifically, counter-examples are given to show that Relative Neighborhood Graphs and Gabriel Graphs are not void-free for certain node sets in [6]; and Delaunay Triangulations are shown to be *void-free* on any node set on the plane in [7]. However, for Yao graphs [8] and Theta graphs (or Θ-graphs) [9], which are also leveraged in several works [10-13] to construct network topologies, no results exist on the void-free property yet.

The importance of Yao graph and Theta graph mainly lie in the wireless networks that use directional antennas. In such networks, each wireless node is equipped with multiple directional antennas and each directional antenna's coverage area is roughly a cone/sector with certain angle [14]. As seen below, the construction of Yao graph and Theta graph are also based on cones. For later reference in this paper, the definitions of Yao graph and Theta graph are stated below.

Given a set V of nodes on the plane, the directed Yao graph with an integer parameter k ($k \geq 2$) on V is obtained as follows. For each node $u \in V$, starting from a given direction (e.g., the direction of positive y-axis), draw k equally-spaced rays l_1, l_2, …, l_k originating from u in clockwise order (see Fig. 1 (a) below). These rays divide the plane into k cones, denoted by $c(u, 1)$, $c(u, 2)$, …, $c(u, k)$ respectively in clockwise order. To avoid overlapping at boundaries, it is required that the area of $c(u, i)$, where i=1, …, k, excludes the ray $l_{i \% k}$ but includes the ray $l_{(i+1) \% k}$. In each cone of u, construct a directed edge from u to its closest node by Euclidean distance in that cone. Ties are broken arbitrarily. These directed edges will form the edge set of the directed Yao graph on V. The undirected Yao graph (or simply Yao graph) on V is obtained by ignoring the directions

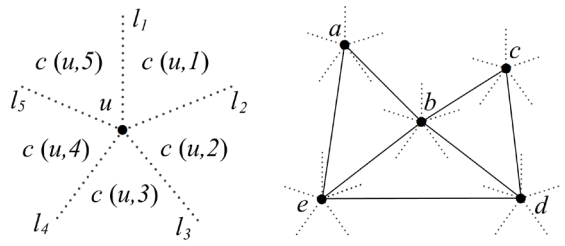


Fig. 1. Cones and an example of Yao graph for k=5.

978-1-4799-3512-3/14/$31.00 ©2014 IEEE

of the edges. Note that if both edge *uv* and *vu* are in the directed Yao graph, only one edge *uv* exists in the Yao graph. Fig. 1 (b) gives an example of Yao graph with k=5.

Similar to Yao graph, the undirected Theta graph (or simply Theta graph) is also obtained by letting each node $u \in V$ select a 'closest' node in each of its cones to establish an edge. The only difference is that 'closest' in Theta graph means the smallest projection distance onto the bisector of that cone, not the direct Euclidean distance. For instance, in Fig. 2, node u's 'closest' node will be node b. For convenience, we denote Yao graph and Theta graph with parameter k as Y_k and Θ_k hereafter.

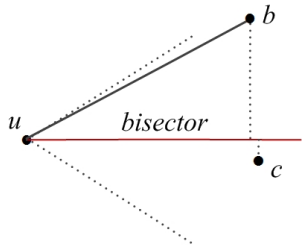


Fig. 2. The bisector in a cone of a Theta graph.

Note that this paper focuses on the study of undirected Yao and Theta graphs, since a link is typically bidirectional in wireless networks. That is, this paper assumes that if node u selects node v as its neighbor under the Yao/Theta graph definition, node v can send message back to node u even though v may not select node u as its neighbor under the Yao/Theta graph definition. The rationale for this assumption is that the multiple directional antennas of a node will cover the entire surrounding of this node and if node u selects node v as neighbor under the Yao/Theta graph definition, one of v's directional antenna can reach u as well.

In brief, this paper made both theoretical and empirical contributions. Theoretically, this paper proved the following results.

- When $2 \le k \le 5$, Y_k and Θ_k are shown to be not void-free for certain node sets on the plane.
- When $k \ge 6$, Y_k and Θ_k are shown to be void-free for any node set on the plane.

Empirically, this paper is the first to conduct experiments on the stretches [15] (detailed in Section IV) of Greedy Forwarding on Yao and Theta graphs, and also compare the results with another well-known graph Delaunay triangulation (DT) [5]. The experiments mainly showed the follows.

- In both hop and Euclidean stretches, the Greedy Forwarding algorithm performs well on both Yao and Theta graphs, and slightly better on Yao graphs than on Theta graphs.
- In both hop and Euclidean stretches, on both Yao and Theta graphs, the Greedy Forwarding algorithm performs generally better when k is even than when k is odd, and performs even better when k is a multiple of six.
- The Greedy Forwarding algorithm performs worse in Euclidean stretch on Yao and Theta graphs than on DTs, but better in hop stretch.

The rest of this paper is organized as follows. Section II shows by counter-examples that Y_k and Θ_k may not be void-free when $2 \le k \le 5$; Section III proves that Y_k and Θ_k are always void-free when $k \ge 6$; Section IV presents our experimental studies; Section V concludes this paper.

II. Counter Examples When $2 \le K \le 5$

When $2 \le k \le 5$, we give counter-example node sets to show that Y_k and Θ_k are not always void-free. For each node set, we describe how it is designed and give applicable (x, y)-coordinates for each node in that set. Using the software developed by us for constructing Yao graph and Theta graph (to be described in Section IV), we verified that each counter-example node set given by us indeed gives Y_k or Θ_k that is not void-free.

Below, we first present counter-example node sets for Y_k in the proof of the following proposition.

Proposition 1. When $2 \le k \le 5$, some node sets exist such that Y_k on them are not void-free.

Proof: When $2 \le k \le 3$, we give a counter-example node set V_0 with four nodes u, v, a, and b as shown in Fig. 3, where the dotted auxiliary circle C_v is centered at v and has radius $d(u, v)$, and the dotted auxiliary lines emanating from each node are only drawn for the case of $k = 3$.

When k = 2 and 3, Y_2 and Y_3 on V_0 are the same and depicted by solid lines in Fig. 3. As node a lies outside the circle C_v, node u does not have a neighbor with a shorter distance to v. So Y_2 and Y_3 are not void-free on V_0.

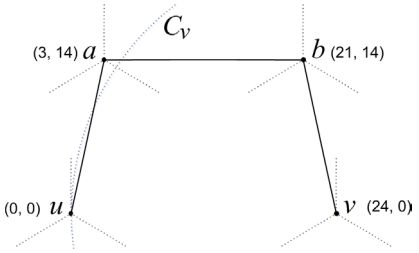


Fig. 3. The counter-example node set V_0 on which Yao graphs are not void-free for $k = 2, 3$.

When $k = 4$, we give a counter-example node set V_1 with six nodes u, v, a, b, c, and d as shown in Fig. 4. In this figure, the dotted circle C_v is centered at v and has radius $d(u, v)$. The resulting Y_4 on V_1 is depicted by solid lines. Since nodes a and b are outside the circle C_v, node u does not have a neighbor with

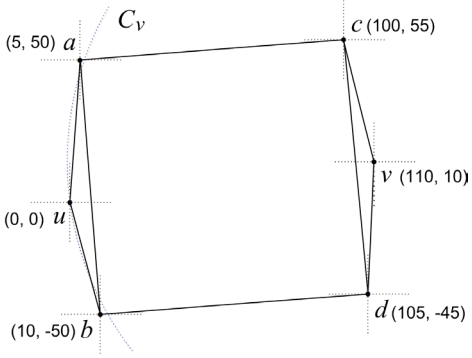


Fig. 4. The counter-example node set V_1 on which Yao graph is not void-free for $k = 4$.

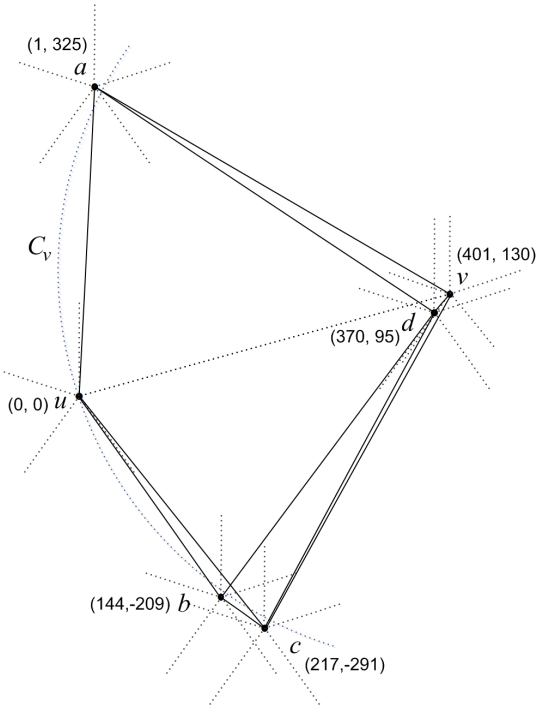


Fig. 5. The counter-example node set V_2 on which Yao graph is not void-free for $k = 5$.

shorter distance to v. So Y_4 is not void-free on V_1.

When $k = 5$, we give a counter-example node set V_2 with six nodes u, v, a, b, c, and d as shown in Fig. 5. Their positions have the following relationship: v is on the ray l_2 originating from u, so v is located inside $c(u, 1)$; d is located inside $c(v, 4)$; b is located inside $c(u, 3)$, $c(d, 4)$, and $c(v, 4)$; c is located inside $c(u, 2)$, $c(d, 3)$ and $c(v, 3)$; a, b, and c are outside the circle C_v which is centered at v and has radius $d(u, v)$. With this node placement, the resulting Y_5 on V_2 is shown by solid lines in Fig. 5. As nodes a, b, and c are outside the circle C_v, node u does not have a neighbor with a shorter distance to v. So Y_5 is not void-free on V_2. □

Next, we consider Θ_k. We verified the following by the software developed by us for constructing Yao graph and Theta graph: for $k = 2, 3$, Θ_k on V_0 are the same as Y_k on V_0; for $k = 4$, Θ_4 on V_1 is the same as Y_4 on V_1; and for $k = 5$, Θ_5 on V_2 is the same as Y_5 on V_2. Thus, the above node sets V_0, V_1, and V_2 can serve as counter examples to make Θ_k not void-free as well. Consequently, we have the following proposition.

Proposition 2. When $2 \le k \le 5$, some node sets exist such that Θ_k on them are not void-free.

III. Proofs for Void-Free When $k \ge 6$

When $k \ge 6$, we show that Y_k and Θ_k are void-free for any node set by constructing theoretical proofs. Below, we first prove the following proposition for Y_k.

Proposition 3. When $k \ge 6$, Y_k are void-free for any node set.

Proof: This proof is done by providing the following method by which, for any given node pair u and v in a Y_k with $k \ge 6$, u can always find a neighbor w in Y_k such that $d(w, v) < d(u, v)$.

We will start with node v, which must reside in one cone of u (see Fig. 6). According to the definition of Y_k, u connects to one of its closest neighbors in that cone. Denote this neighbor by w. If w is the same node as v, we have $d(w, v) = 0 < d(u, v)$, thus finishing the proof.

If w is a different node from v, we still can show that $d(w, v) < d(u, v)$ as follows. If u, w, and v are collinear, w lies inside the line segment uv, so we have $d(w, v) < d(u, v)$. Otherwise, we can draw a triangle connecting nodes u, w, and v. When $k \ge 6$, the angle of a cone is no more than $\pi/3$. Because w and v cannot fall on different boundaries of a cone at the same time (due to cone's definition), we have $\angle wuv < \pi/3$ and also $\angle uvw + \angle uwv > 2\pi/3$. Since $d(u, w) \le d(u, v)$, we have $\angle uvw \le \angle uwv$. Since we already know $\angle uvw + \angle uwv > 2\pi/3$, we have $\angle uwv > \pi/3$. Thus, $\angle uwv > \angle wuv$. Since a larger side is opposite a larger angle in a triangle, we have $d(u, v) > d(w, v)$. □

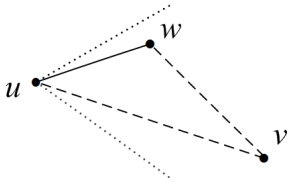


Fig. 6. Proof for Yao graph for $k \ge 6$.

Next, we give the proof for Θ_k in Proposition 4.

Proposition 4. When $k \ge 6$, Θ_k are void-free for any node set.

Proof: This proof is also done by providing a method by which for any given u, v in a Θ_k with $k \ge 6$, u can always find a neighbor w in Θ_k such that $d(w, v) < d(u, v)$.

Similar to the proof of Proposition 3, node v must reside in one cone of u. According to the definition of Θ_k, u connects to its neighbor that has the shortest projection distance on the bisector of that cone. Denote this neighbor by w, and the projection points of w and v on the bisector by w' and v' respectively. If w is the same node as v, we easily have $d(w, v) = 0 < d(u, v)$. Otherwise, we prove $d(w, v) < d(u, v)$ as follows.

If u, w, and v are collinear, w lies inside the line segment uv, so we have $d(w, v) < d(u, v)$. Otherwise, nodes u, w, and v form a triangle. There can be two cases regarding the orientation of this triangle (see Fig. 7): in Case 1, $\angle uwv$ faces the bisector; in Case 2, $\angle uwv$ does not.

For Case 1, when $k \ge 6$, the angle between the boundary and

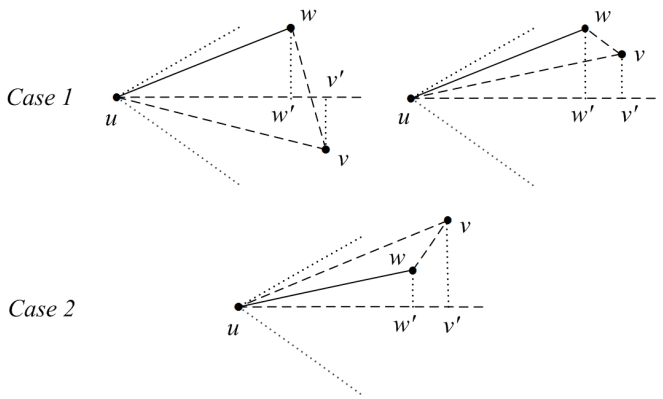
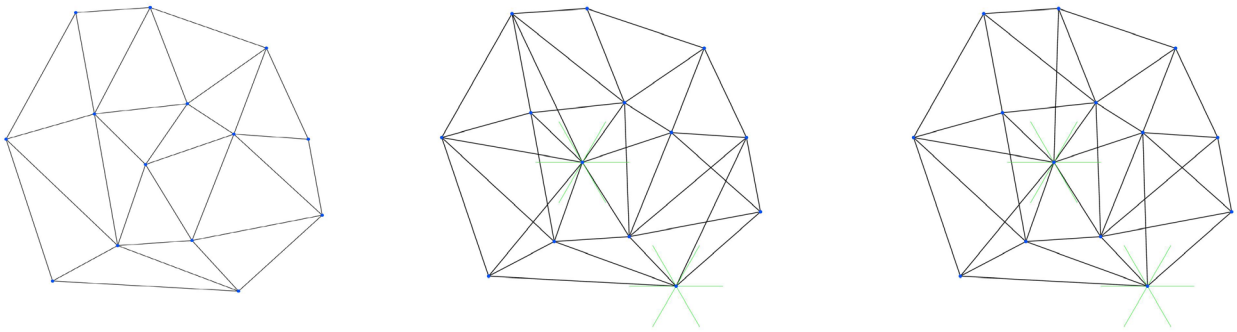


Fig. 7. Proof for Theta graph for $k \ge 6$.



(a) *Delaunay triangulation (DT)* (b) *Theta graph* (c) *Yao graph*

Fig. 8. Examples of Delaunay triangulation, Theta and Yao graphs with k=6 constructed by our software on a 14-node set.

the bisector is no more than π/6. Thus we have $\angle uww' \geq \pi/2 - \pi/6 = \pi/3$. Since $d(u, w') \leq d(u, v')$, we have $\angle uww' \leq \angle uwv$. Because w and v cannot fall on different boundaries of a cone (due to cone's definition), we have $\angle wuv < \pi/3$. Thus, $\angle wuv < \angle uwv$. Since a larger side is opposite a larger angle in a triangle, we have $d(w, v) < d(u, v)$.

For Case 2, first we can easily have $d(u, w) < d(u, v)$. Then, following the proof for Proposition 3, we can obtain $d(w, v) < d(u, v)$. □

As a note, since the above two proofs for Propositions 3 and 4 do not need the assumption that no ties exist when a node selects a 'closest' neighbor in a cone in Yao graph or Theta graph, these two proofs still work when a node set can introduce such ties.

IV. Experimental Results

In the previous sections, we proved that when $k \geq 6$, Y_k and Θ_k are guaranteed to be void-free. However, it is still unclear how well these two types of graphs support the Greedy Forwarding algorithm. To evaluate on this aspect, we measured a metric called *stretch* [15]. Specifically, the *stretch* by a routing algorithm Γ for a source/destination pair (s, t) in a graph G is defined as:

$$\frac{\text{stretch}}{\text{by } \Gamma \text{ for } (s,t)} = \frac{\text{Length of path found by } \Gamma \text{ from } s \text{ to } t}{\text{Length of shortest path from } s \text{ to } t} \quad (1)$$

Further, the *average stretch* by Γ on G is the average stretch of all (s, t) pairs in G; the *maximum stretch* by Γ on G is the largest stretch of all (s, t) pairs in G. In this paper, we consider Greedy Forwarding as the routing algorithm.

Note that, to measure path length, two metrics are commonly used: number of hops and Euclidean distance. The hop metric is generally used in wireless networks, since packet forwarding in each hop involves much more significant transmission delay and queuing delay than propagation delay, which is trivial due to the light speed of wireless signal. On the other hand, the Euclidean metric is generally used in transport networks or robotics, where the propagation delay becomes significant due to the low speed of vehicles or robots. To make our work useful for different areas, we measure the stretches by Greedy Forwarding in both hop and Euclidean metrics in our experiments.

Also, to help explain the results on *stretches*, we experiment on the *average node degrees* (denoted deg_{avg} hereafter) in Theta graphs and Yao graphs as well. Note that, deg_{avg} of a graph equals $2m/n$, where m is the number of edges and n is the number of nodes in this graph.

For all the aforementioned aspects of evaluation: *stretch of Greedy Forwarding* and *average node degree* in Yao graph and Theta graph, we add a third graph type *Delaunay triangulation* (DT) [5] as a comparison, since DT is also applied as wireless network topologies in some research efforts [16, 17]. In such comparisons, we set k=6 for Yao graph and Theta graph, since 6 is the smallest k that enables those two graphs void-free.

Below, we first describe our experimental setup, and then present our results in the following order: *average node degree*, *stretch of Greedy Forwarding in hop metric*, then *in Euclidean metric*.

A. Experiments Setup

We developed software to construct Yao graph, Theta graph and DT and then calculate on the deg_{avg} and *stretch of Greedy Forwarding* in these three types of graphs. Our software is implemented using the Computational Geometry Algorithms Library (CGAL) [18], which provides us APIs for basic geometric calculations. Fig. 8 gives an example of DT, Yao graph and Theta graph with k=6 constructed by our software on a 14-node set respectively.

In our experiments, we generated nodes according to uniform distribution in a disk area. Also, in both experiments on deg_{avg} and *stretch*, we first fix k to 6 and vary the number of nodes (denoted by n hereafter) to see the influence of n, and then we fix n to 800 and vary k to see the influence of k. For each combination of n and k, we conducted 1000 experiments with each node placement randomly generated. Then, the average results of these 1000 experiments are plotted in the upcoming figures.

B. Average node degree

Fig. 9 plots the deg_{avg}'s for DTs, Yao graphs and Theta graphs with k=6 and n = 100, 200, …, 900 and 1000 respectively. In a DT, we have $e = 3n - c - 3$, where e is the number of edges and c is the number of convex hull edges [5]. So we know that deg_{avg} of a DT should approach 6 with the increase of n, which is confirmed in Fig. 9. For Theta graph, the

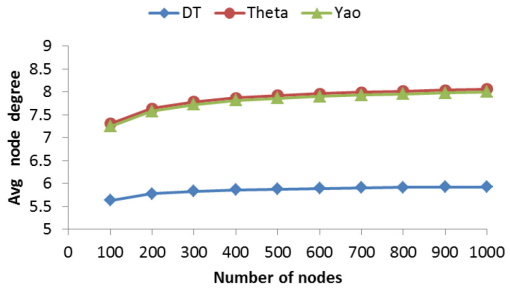


Fig. 9. Average node degrees with different n's and k=6.

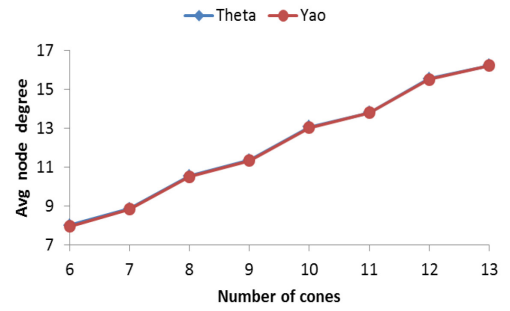


Fig. 10. Average node degrees with different k's and n=800.

following asymptotic result exists on its deg_{avg} when n is made infinite [19]:

$$deg_{avg} \approx \begin{cases} 1.3954 \cdot k & when\ k\ is\ even \\ 1.3565 \cdot k & when\ k\ is\ odd \end{cases} \quad (2)$$

For Yao graph, [19] states that its deg_{avg} is similar to Theta graph, but the closed form formula of its deg_{avg} cannot be found. According to our observation from Fig. 9, when we fix k to 6, deg_{avg} of Theta graph is roughly the same as Yao graph, and is about 1.3 times the deg_{avg} of DT.

Fig. 10 plots the deg_{avg}'s of Yao graph and Theta graph with n=800 and k = 6, 7, … , 12 and 13 respectively. We can mainly see the follows from this figure. First, the deg_{avg}'s of Yao graph and Theta graph are about the same at different k's. Second, the data lines of both graphs exhibit a zigzag shape, which conforms with formula (2) where an even k and an odd k involve different coefficients. This zigzag of deg_{avg} will be used to explain the zigzag of *stretches* in the coming figures.

C. Stretch of Greedy Forwarding in hop metric

Fig. 11 plots the average stretch of Greedy Forwarding in hop metric on graphs with n = 100, 200, …, 900 and 1000 respectively. This figure mainly shows that (1) for the node numbers experimented, Greedy Forwarding on Yao graph achieves a slightly smaller stretch than on Theta graph, and (2) Greedy Forwarding on DTs has a much larger stretch than on Yao graphs and Theta graphs, since DTs have smaller deg_{avg} than Yao graphs and Theta graphs.

Fig. 12 plots the maximum stretch of Greedy Forwarding in hop metric when n changes. The maximum stretch is used to reflect the worst case performance of Greedy Forwarding. Basically, this figure shows similar performance relationship among DT, Theta graph and Yao graph in worst case as in average case in Fig. 11.

Fig. 13 plots the average stretch of Greedy Forwarding in hop metric when k changes. This figure mainly shows the following. First, the average hop stretches on both Yao graph and Theta graph bear a zigzag shape, taking a larger value at an odd k and a smaller value at an even k. This is largely due to the zigzag shape of deg_{avg}'s of Yao graph and Theta graph as depicted in Fig. 10. Second, when k equals 6 or 12, Greedy Forwarding achieves a better hop stretch than other experimented k's on both Yao and Theta graphs. This implies that *six* is an efficient number for Yao and Theta graphs, and the underlying reason needs to be further explored.

D. Stretch of Greedy Forwarding in Euclidean metric

Fig. 14 plots the average stretch of Greedy Forwarding in Euclidean metric on graphs with n = 100, 200, …, 900 and 1000 respectively. This figure shows very different phenomena from those in hop metric shown in Fig. 11. First, Greedy Forwarding achieves almost identical average Euclidean stretch on Yao graph and Theta graph, with sometimes a slightly smaller stretch on Yao and sometimes a slightly smaller stretch on Theta, so it is hard to tell on which graph Greedy Forwarding performs better. Second, though DTs have smaller deg_{avg}'s than Yao and Theta graphs, Greedy Forwarding achieves better Euclidean stretch on DTs than on Yao and Theta graphs. This implies the graph structure of DTs has a much better support on Euclidean stretch than on hop stretch.

Fig. 15 plots the maximum stretch of Greedy Forwarding in Euclidean metric when n changes. This figure basically shows the same phenomena as Fig. 14, except that Greedy Forwarding consistently achieves smaller maximum Euclidean stretches on Yao graph than on Theta graph, reflecting that Greedy Forwarding performs more stably on Yao graph than on Theta graph.

Fig. 16 plots the average stretch of Greedy Forwarding in Euclidean metric when k changes. This figure shows similar phenomena as those in hop metric shown in Fig. 13. The explanation will also resort to the zigzag shape of deg_{avg}'s of Yao graph and Theta graph shown in Fig. 10.

Since this section presented many experimental results, *a detailed summary of these results is provided in the Conclusions section to enable a quick overview*.

V. Conclusions

Since the Greedy Forwarding algorithm is an important routing algorithm used in wireless networks, this paper studied how well Yao graph and Theta graph support Greedy Forwarding, and presented both theoretical and empirical results. Theoretically, this paper proved that (1) when the number of cones is less than 6, Yao graphs or Theta graphs are not void-free on some node sets on the plane, and (2) when the number of cones equals or exceeds 6, Yao graphs and Theta graphs are void-free for any node set on the plane. Experimentally, this paper mainly showed the following results.

- The Greedy Forwarding algorithm performs well on both Yao and Theta graphs, with average stretch less than 1.14 in both hop and Euclidean metrics on all experimented parameters in this paper.

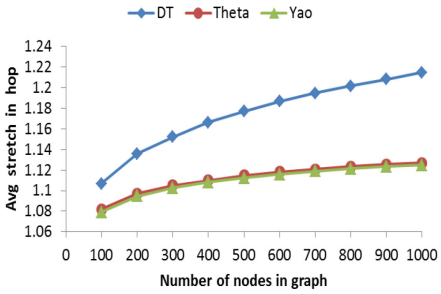


Fig. 11. Average stretch in hop with different n's and k=6.

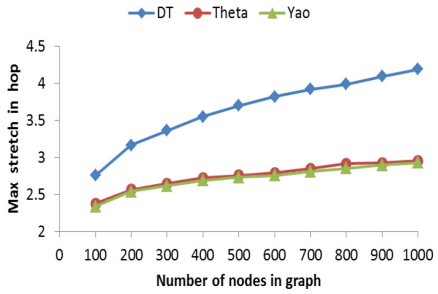


Fig. 12. Maximum stretch in hop with different n's and k=6.

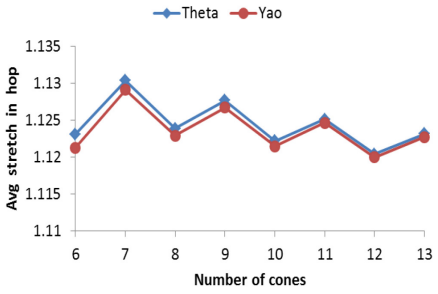


Fig. 13. Average stretch in hop with different k's and n=800.

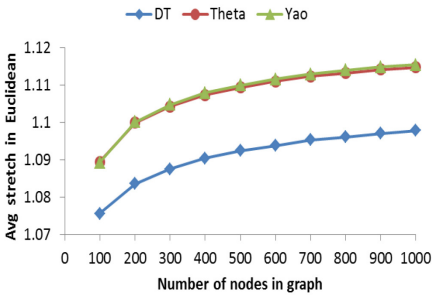


Fig. 14. Average stretch in Euclidean with different n's and k=6.

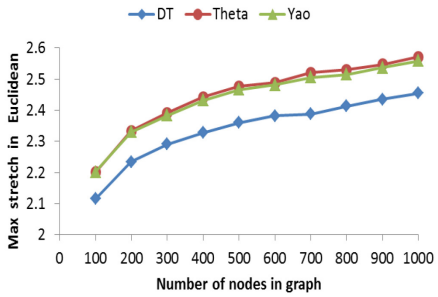


Fig. 15. Maximum stretch in Euclidean with different n's and k=6.

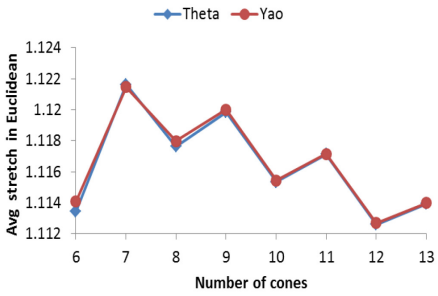


Fig. 16. Average stretch in Euclidean with different k's and n=800.

- In hop metric, Greedy Forwarding achieves slightly smaller stretches in both average and worst cases on Yao graphs than on Theta graphs. Moreover, both Yao and Theta graphs with k=6 give smaller stretches in both average and worst cases than DTs.
- In Euclidean metric, Greedy Forwarding achieves slightly smaller stretch in worst case on Yao graph than on Theta graph. In average case, it is hard to tell which graph performs better. Surprisingly, though Yao graph and Theta graph with k=6 have larger deg_{avg}'s than DTs, both of them incur larger stretches than DTs, reflecting that DTs are very suitable in finding short paths in Euclidean metric.
- In both hop and Euclidean metrics, for both Yao and Theta graphs, Greedy Forwarding generally has smaller average stretches when k is even than when k is odd; especially when k is a multiple of six, Greedy Forwarding sees the smallest average stretches, showing that six is an efficient number for Yao and Theta graphs.

REFERENCES

[1] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, "Geometric ad-hoc routing: of theory and practice," in *ACM PODC*, pp. 63--72, 2003.

[2] D. Chen and P. K. Varshney, "A Survey Of Void Handling Techniques For Geographic Routing In Wireless Networks," *IEEE Communications Surveys & Tutorials,* vol. 9, pp. 50-67, 2007.

[3] G. T. Toussaint, "The relative neighborhood graph of a finite planar set," *Pattern Recognition,* vol. 12, pp. 261-268, 1980.

[4] K. R. Gabriel and R. R. Sokal, "A new statistical approach to geographic variation analysis," *Systematic Zoology,* vol. 18, pp. 259-278, 1969.

[5] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars, *Computational geometry: algorithms and applications*, 3rd ed. New York: Springer-Verlag, 2008.

[6] W. Si, B. Scholz, J. Gudmundsson, G. Mao, R. Boreli, and A. Y. Zomaya, "On Graphs Supporting Greedy Forwarding for Directional Wireless Networks," in *IEEE International Conference on Communications (ICC)*, 2012.

[7] P. Bose and P. Morin, "Online Routing in Triangulations," *Siam Journal of Computing,* vol. 33, pp. 937-951, 2004.

[8] A. C. Yao, "On constructing minimum spanning trees in k-dimensional spaces and related problems," *SIAM Journal on Computing,* 1982.

[9] K. Clarkson, "Approximation algorithms for shortest path motion planning," in *ACM Symposium on Theory of Computing*, pp. 56-65, 1987.

[10] F. Li, Z. Chen, and Y. Wang, "Localized Topologies with Bounded Node Degree for Three Dimensional Wireless Sensor Networks," in *International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, 2011.

[11] S. Poduri, S. Pattem, B. Krishnamachari, and G. S. Sukhatme, "Using Local Geometry for Tunable Topology Control in Sensor Networks," *IEEE Transactions on Mobile Computing,* vol. 8, pp. 218-230, 2009.

[12] W.-Z. Song, Y. Wang, X.-Y. Li, and O. Frieder, "Localized algorithms for energy efficient topology in wireless ad hoc networks," in *ACM Int. Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2004.

[13] X.-Y. Li, W.-Z. Song, and W. Wang, "A unified energy-efficient topology for unicast and broadcast," in *ACM Int. Conference on Mobile Computing and Networking (MobiCom)*, 2005.

[14] Z. Yu, J. Teng, X. Bai, D. Xuan, and W. Jia, "Connected coverage in wireless networks with directional antennas," in *IEEE INFOCOM* pp. 2264-2272, 2011.

[15] R. Flury, S. V. Pemmaraju, and R. Wattenhofer, "Greedy Routing with Bounded Stretch," in *IEEE INFOCOM*, pp. 1737-1745, 2009.

[16] 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.

[17] W. Si and A. Y. Zomaya, "New Memoryless Online Routing Algorithms for Delaunay Triangulations," *IEEE Transactions on Parallel and Distributed Systems,* vol. 23, pp. 1520-1527, 2012.

[18] CGAL, *Computational Geometry Algorithms Library,* http://www.cgal.org/, 2013.

[19] P. Morin and S. Verdonschot, "On the average number of edges in Theta graphs " in *SIAM Meeting on Analytic Algorithmics & Combinatorics (ANALCO)*, pp. 1-20, 2014.