J. Parallel Distrib. Comput. 117 (2018) 87–97

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

On the performance of greedy forwarding on Yao and Theta graphs

Weisheng Si [a,*], Quincy Tse [a], Guoqiang Mao [b], Albert Y. Zomaya [c]

[a] School of Computing, Engineering and Mathematics, Western Sydney University, Australia
[b] School of Computing and Communications, University of Technology Sydney, Australia
[c] School of Information Technologies, University of Sydney, Australia

HIGHLIGHTS

- When cone number is less than 6, Yao or Theta graph is not void-free on certain node sets in the plane.
- When cone number is larger than or equal to 6, both Yao and Theta graphs are void-free on any node set in the plane.
- Greedy Forwarding performs better on Yao or Theta graphs with even cone numbers than those with odd cone numbers.
- Six is probably the most suitable cone number for Yao and Theta graphs as network topologies.
- We contributed our software for graph construction to CGAL (http://www.cgal.org).

ARTICLE INFO

Article history:
Received 7 November 2016
Received in revised form 2 November 2017
Accepted 17 February 2018
Available online 27 February 2018

Keywords:
Wireless networks
Geometric routing
Greedy forwarding
Theta graph
Yao graph
Voids

ABSTRACT

Greedy Forwarding algorithm is an important geometric routing algorithm for wireless networks. However, it can fail if the network topologies contain voids, in which a packet cannot be moved closer to destination. Since Yao and Theta graphs are two types of geometric graphs exploited to construct wireless network topologies, this paper first gives theoretical results on whether these two types of graphs can contain voids with respect to their cone numbers. Then, this paper examines the performance of Greedy Forwarding on Yao and Theta graphs in terms of stretch (the ratio between the path length found by Greedy Forwarding and the shortest path length from a source to a destination). Both worst-case and average-case stretches are studied. For the worst case, this paper shows that the stretches of Greedy Forwarding on both Yao and Theta graphs do not have a constant upper bound (i.e., not competitive) in hop metric. For the average case, this paper studies the stretch experimentally by running Greedy Forwarding on a large number of Yao and Theta graphs with randomly generated node sets. The average-case stretches in both hop and Euclidean metrics are measured, and several interesting findings are revealed.

1. Introduction

When studying geometric (or geographic) routing algorithms [20] in wireless networks, the network topology is usually modeled 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 this geometric graph model is the Greedy Forwarding (also known as Greedy Routing) algorithm [11]: 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 the edge uv is present in the graph.

However, Greedy Forwarding (GF) does not succeed on a graph that contains *void* [11,14], in which for a 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 void becomes an important property to study. For the convenience of discussion, 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 [33], Gabriel Graph [17], and Delaunay Triangulation [2]. Specifically, Relative Neighborhood Graphs and Gabriel Graphs have been shown not void-free for certain node sets [27]; and Delaunay Triangulations have been shown void-free on any node set on the plane [8]. However, for Yao graphs [38]

* Corresponding author.
E-mail addresses: w.si@westernsydney.edu.au (W. Si), quincy.tse@gmail.com (Q. Tse), guoqiang.mao@uts.edu.au (G. Mao), albert.zomaya@sydney.edu.au (A.Y. Zomaya).

https://doi.org/10.1016/j.jpdc.2018.02.006
0743-7315/© 2018 Elsevier Inc. All rights reserved.

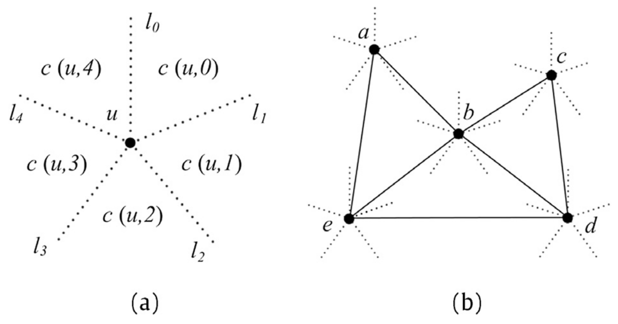


Fig. 1. Cones and an example of Yao graph for $k = 5$.

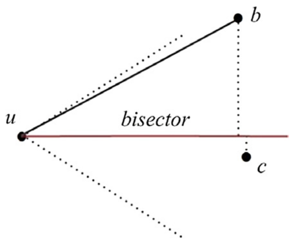


Fig. 2. Bisector in a cone of a Theta graph.

and Theta graphs (or Θ-graphs) [12], which are also leveraged in several works [21,23,25,32] to construct network topologies, no results exist on the void-free property yet. Moreover, how well GF performs on Yao and Theta graphs in terms of stretch [15] has not been studied to date.

The importance of Yao and Theta graphs mainly lie in the wireless networks (e.g., wireless mesh 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* with certain angle [39]. Note that the term *cone* in the literature of Yao and Theta graphs [12,38] means a sector (i.e., a two dimensional concept). Considering directional antennas, this paper does not assume the Unit Disk Graph (UDG) model [36] commonly used in wireless networks. In the UDG model, each node's coverage area is a unit disk, thus greatly limiting the link length possible. Instead, this paper assumes that each directional antenna on a wireless node can adjust its power to achieve desired link length, which allows us to study the complete Yao and Theta graphs without the link length constraint. This assumption is reasonable because in the real-world wireless networks, we can deploy wireless nodes in a way such that the links needed among them are realizable.

1.1. Definitions of Yao and Theta graphs

Since rigorous definitions of Yao and Theta graphs are needed by later proofs, we present them here.

Given a set of nodes V on the plane, the *directed Yao graph* with k cones (k: denoting the number of cones throughout this paper, $k \geq 2$) 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 $l_0, l_1, \ldots, l_{k-1}$ originating from u in clockwise order (see Fig. 1(a)). These rays divide the plane into k cones, denoted by $c(u, 0)$, $c(u, 1)$, ..., $c(u, k-1)$ respectively in clockwise order. To prevent the overlapping at boundaries, it is assumed here that the area of $c(u, i)$, where $i = 0, \ldots, k-1$, includes the ray l_i but excludes 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 following the same procedure but ignoring the directions of edges. Note that if both edge uv and vu are in the *directed Yao graph*, only one edge uv exists in the resulting *Yao graph*, thus removing duplicate edges. 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 and Theta graphs with k cones as Y_k and Θ_k hereafter.

Note that this paper studies undirected Yao and Theta graphs, since a link is typically bidirectional in wireless networks. That is, if node u selects node v as its neighbor under the Yao/Theta graph definition, there will be a bidirectional link between u and v, even when v does not select u as its neighbor. This complies with real-world scenarios because the multiple directional antennas of node v will cover the entire surrounding of node v and if node u selects node v as neighbor, one of v's directional antenna can reach u as well.

1.2. Contributions of this paper

First, for the void-free property of Yao and Theta graphs, this paper[1] shows that:

- When $2 \leq k \leq 5$, Y_k and Θ_k are not void-free for certain node sets on the plane.
- When $k \geq 6$, Y_k and Θ_k are void-free for any node set on the plane.

Second, this paper further studies the performance of GF on Yao and Theta graphs in terms of *stretch* [15,19,29] (formally defined in Section 4). Both worst-case and average-case stretches were studied, and for both worst and average cases, the study considered both hop and Euclidean metrics. Note that when the path length is measured by the number of hops, we obtain the *stretch in hop metric*; and when the path length is measured by the sum of the Euclidean distances of the edges, we obtain the *stretch in Euclidean metric*. While hop metric is important for wireless networks, Euclidean metric is important for areas such as transport planning and robotics where the Euclidean path length is more meaningful than the number of hops.

Specifically, for the worst-case stretch, this paper proves that a constant upper bound does not exist for GF on Yao or Theta graph in hop metric. Note that in the literature, if the stretch by a routing algorithm on a type of graph has a constant upper bound, this routing algorithm is said to be *competitive* [5] on this type of graph. Thus, in other words, this paper shows that GF is not *competitive* on Yao or Theta graph in hop metric. In Euclidean metric, we conjecture that GF is competitive on both Yao and Theta graphs. Unfortunately, we are unable to prove this in this paper.

For the average-case stretches, the results in both hop and Euclidean metrics were obtained by running GF on a large number of randomly generated Yao and Theta graphs. Moreover, for comparison purpose, we also measured the stretches in both hop and Euclidean metrics by GF on Delaunay triangulations (DTs) [2], since DT is another type of geometric graphs exploited in constructing wireless networks in several research efforts [22,31,34]. In brief,

[1] A preliminary version of this paper [19] appears in IEEE GLOBECOM 2014.

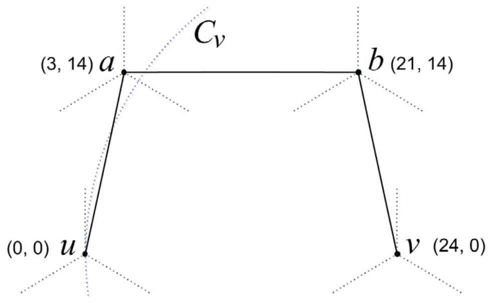


Fig. 3. Counter example node set V_0 on which Yao graphs are not void-free for $k = 2, 3$.

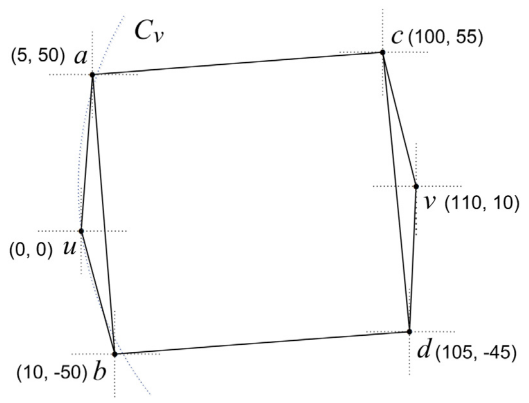


Fig. 4. The counter example node set V_1 on which Yao graph is not void-free for $k = 4$.

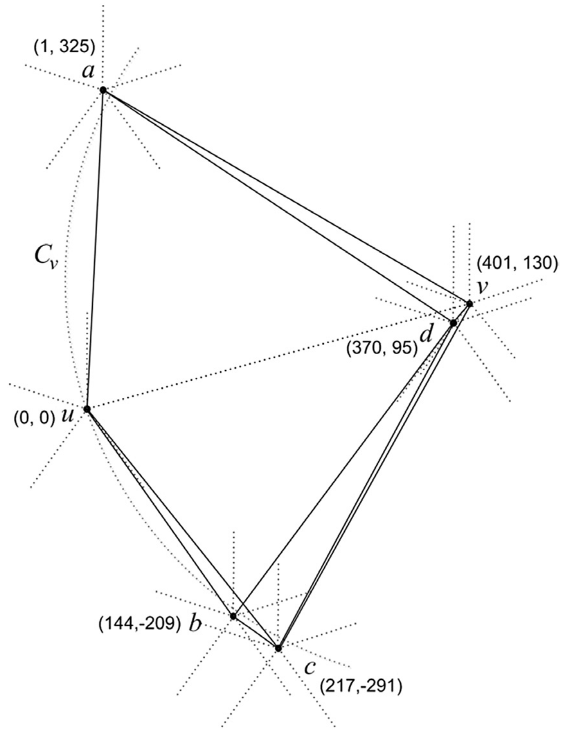


Fig. 5. The counter example node set V_2 on which Yao graph is not void-free for $k = 5$.

we are the first to experiment on the average-case stretches by GF on Yao and Theta graphs and compare them with DTs; and our experiments revealed several very interesting findings (summarized in Section 5.5).

Finally, we contributed our software package for constructing Yao and Theta graphs on a set of nodes to a prestigious open source library named CGAL (Computational Geometry Algorithm Library [10]). One significant advantage of our package is that it supports the exact construction of Yao and Theta graphs, in which the cone boundaries are calculated by using roots of polynomials, thus avoiding using the value of π, which cannot be represented exactly by computers. For how to use our package, please refer to its user manual at [28].

The rest of this paper is organized as follows. Section 2 shows by counter examples that Y_k and Θ_k are not void-free when $2 \le k \le 5$; Section 3 proves that Y_k and Θ_k are void-free when $k \ge 6$; Section 4 shows by counter examples that GF is not competitive on Y_k and Θ_k in hop metric when $k \ge 6$; Section 5 presents our experimental studies on average-case stretches; Section 6 provides a survey of related work; finally, Section 7 concludes this paper.

2. 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 (x, y)-coordinates for each node. We also prove theoretically and verify by our software that each counter example node set designed by us gives a Yao/Theta graph that is not void-free.

Below, we first present counter example node sets for Y_k $(2 \le k \le 5)$ within the proof of the following proposition. Then, we show that these node sets can be applied to Θ_k $(2 \le k \le 5)$ as well.

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.

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 a 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 located inside $c(u, 1)$ and very close to the ray l_1 originating from u; d is located inside $c(v, 3)$; b is located inside $c(u, 2)$, $c(d, 3)$ and $c(v, 3)$; c is located inside $c(u, 1)$, $c(d, 2)$ and $c(v, 2)$; 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 both manual calculation and our software for constructing Yao and Theta graphs: 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

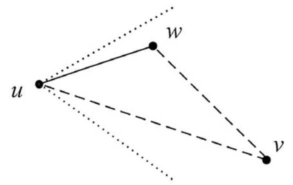


Fig. 6. Proof for Yao graph for $k \geq 6$.

as counter examples for generating $\Theta_k(2 \leq k \leq 5)$ which are not void-free as well. Consequently, we have the following proposition.

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

3. Proofs for void-freeness when $k \geq 6$

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

Proposition 3. *When $k \geq 6$, Y_k are void-free for any node set in the plane.*

Proof. This proof is done by providing a method by which, for any two different nodes u, v in a Y_k with $k \geq 6$, u can always find a neighbor w 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 Yao graph, u connects to one of its closest neighbors in that cone. We will use this neighbor connected by u as the node w. Next, we prove that $d(w, v) < d(u, v)$.

If w is the same node as v, we have $d(w, v) = 0 < d(u, v)$, thus completing the proof. Otherwise, w is a different node from v. In this case, if u, w, and v are collinear, then w must strictly lie inside the line segment uv, since w is the closest node to u in that cone. Thus, we have $d(w, v) < d(u, v)$. On the other hand, if u, w, and v are not collinear, we can draw a triangle connecting nodes u, w, and v. When $k \geq 6$, the angle of a cone is no more than $\pi/3$. Because w and v cannot fall on different boundaries of a cone due to the way a cone is defined in the Section 1.1, we have $\angle wuv < \pi/3$ and also $\angle uvw + \angle uwv > 2\pi/3$. Since $d(u, w) \leq d(u, v)$, we have $\angle uvw \leq \angle uwv$. 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 faces a larger angle in a triangle, we have $d(u, v) > d(w, v)$. ■

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

Proposition 4. *When $k \geq 6$, Θ_k are void-free for any node set in the plane.*

Proof. This proof is also done by providing a method by which, for any two different nodes u, v in a Θ_k with $k \geq 6$, u can always find a neighbor w 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 Theta graph, u connects to one of its neighbors that have the shortest projection distance on the bisector of that cone. We will use this neighbor connected by u as the node w. Next, we prove that $d(w, v) < d(u, v)$.

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 must strictly lie inside the line segment uv, since w has the smallest projection distance to u in that cone. Thus, 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

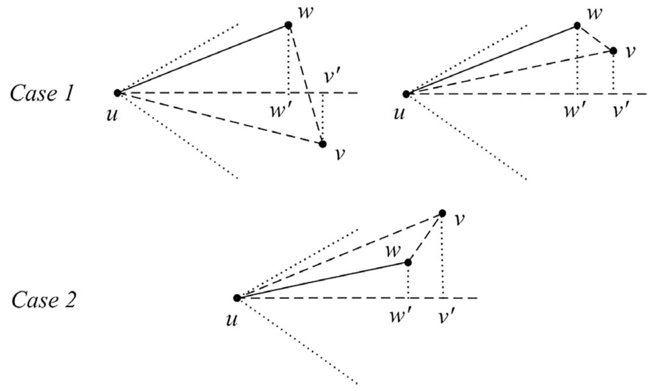


Fig. 7. Proof for Theta graph for $k \geq 6$.

Table 1
Symbols used in this and later sections.

G	A graph
s, t	Source, destination nodes in G
Γ	A geometric routing algorithm
$\Gamma(G, s, t)$	The length of path found by Γ from s to t in G
$SP\ (G, s, t)$	The shortest path length from s to t in G
$stretch_\Gamma\ (G, s, t)$	The *stretch* byΓ from s to t in G
$\Gamma_{avg}(G)$	The average $\Gamma\ (G, s, t)$ of all (s, t) pairs in G
$SP_{avg}(G)$	The average $SP\ (G, s, t)$ of all (s, t) pairs in G
$stretch_{\Gamma,avg}(G)$	The average $stretch_\Gamma\ (G, s, t)$ of all (s, t) pairs in G
$stretch_{\Gamma,max}(G)$	The maximum $stretch_\Gamma\ (G, s, t)$ of all (s, t) pairs in G
$d\ (s, t)$	The Euclidean distance between s and t

of this triangle (see Fig. 7): in Case 1, $\angle uwv$ faces the bisector; in Case 2,$\angle uwv$ does not. In the discussion of both cases below, we denote the projection points of w and v on the bisector w' and v' respectively.

In Case 1, when $k \geq 6$, the angle between the boundary and the bisector is no more than $\pi/6$. 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 the way a cone is defined in the Section 1.1, we have $\angle wuv < \pi/3$. Thus, $\angle wuv < \angle uwv$. Since a larger side faces a larger angle in a triangle, we have $d(w, v) < d(u, v)$.

In Case 2, we have $d(u, w) = d(u, w')/\cos(\angle wuw')$, and $d(u, v) = d(u, v')/\cos(\angle vuv')$. Since $d(u, w') \leq d(u, v')$ and $\cos(\angle wuw') > \cos(\angle vuv')$, we have $d(u, w) < d(u, v)$. Then, following the proof in Proposition 3 on the case where u, w and v form a triangle, we obtain $d(w, v) < d(u, v)$. ■

As a note, 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 or Theta graph, so these two proofs still work when a node set contains such ties.

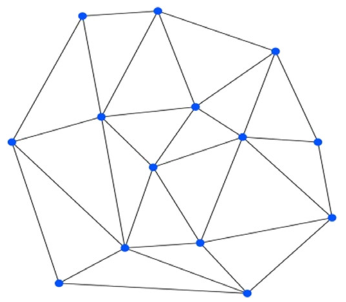
4. Competitiveness of greedy forwarding

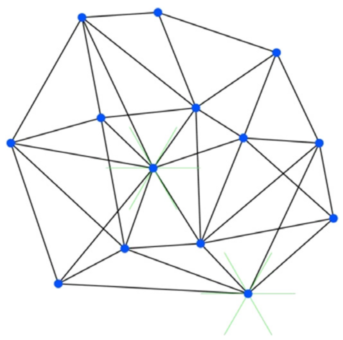
The propositions in the previous section imply that GF always succeed on Y_k and Θ_k when $k \geq 6$, which enables us to study the competitiveness of GF on Y_k and Θ_k when $k \geq 6$. Below, we first give the formal definitions of *stretch* [15] and *competitiveness* [5], which are needed by later proofs. For convenience, the symbols used in these definitions and later part of the paper are summarized in Table 1.

With the symbols in Table 1, the *stretch* by a routing algorithm Γ from s to t in G is defined as:

$$stretch_\Gamma(G, s, t) = \frac{\Gamma(G, s, t)}{SP(G, s, t)} \tag{1}$$

Further, $stretch_{\Gamma,avg}(G)$ is defined as the average $stretch_\Gamma(G, s, t)$ of all (s, t) pairs in G; and $stretch_{\Gamma,max}(G)$ is defined as the largest


(a) Delaunay triangulation (DT).


(b) Theta graph.

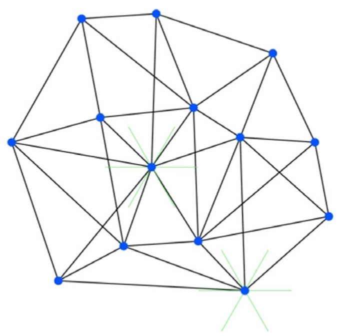

(c) Yao graph.

Fig. 8. Examples of DT, Θ_6 and Y_6 constructed by our software on a 14-node set. Light green lines in (b) and (c) show cone boundaries. (For the colour version of this figure, the reader is referred to the web version of this article.)

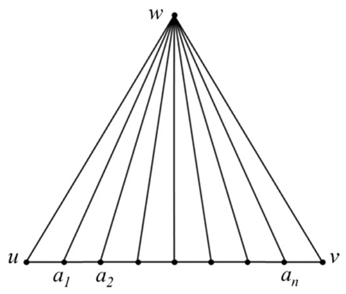


Fig. 9. The counter example node set on which Yao graph is not competitive in hop metric.

stretch$_\Gamma(G, s, t)$ of all (s, t) pairs in G. If the *stretch*$_{\Gamma,\max}(G)$ of every instance G of a graph type has a constant upper bound c, Γ is said to be *competitive* on this type of graphs and c is called the *competitive ratio* [8]. Note that c is a constant here, which is the basic case of upper bound analysis; the more complex case where the upper bound can be an expression is beyond the scope of this paper.

With the above said, we prove the following proposition.

Proposition 5. *For any $k \geq 6$, GF is not competitive on Y_k in hop metric.*

Proof. The proof is done by giving a series of counter example Yao graphs on which GF can have arbitrarily large *stretch*$_{GF,max}(G)$ in hop metric.

Consider a kind of node sets consisting of $n + 3$ nodes: $u, v, w, a_1, a_2 \dots, a_n$, in which $u, v, a_1, a_2 \dots, a_n$ are placed on a horizontal line, and w is placed above the line uv and satisfies that $\angle wuv > \pi/3$ and $\angle wvu > \pi/3$ (see Fig. 9 for an illustration). When $k \geq 6$, the angle of cones is no more than $\pi/3$, which enables w to be the only node in one of cones of $u, v, a_1, a_2 \dots, a_n$ respectively, so the edges $uw, vw, a_1w, a_2w, \dots, a_nw$ all exist in the resulting Yao graph no matter how the cones are oriented. Also, since $u, v, a_1, a_2 \dots, a_n$ are on the same line, the edges $ua_1, a_1a_2, \dots, a_nv$ exist in the resulting Yao graph.

On this Yao graph, use GF to find a path from u to v. Since $\angle wuv > \pi/3$ and $\angle wvu > \pi/3$, we have $d(w, v) > d(u, v)$. Thus, GF will find the path $ua_1a_2 \dots a_nv$ that consists of $n+1$ hops, while the shortest path is uwv that consists of 2 hops. Therefore, the maximum stretch of GF on this kind of graphs is $(n+1)/2$. Since we can increase n arbitrarily, GF is not competitive on Yao graphs. ∎

It is easy to verify that the Theta graphs generated by this kind of node sets are exactly the same as the Yao graphs generated by them, so we get a similar proposition for Theta graphs as well.

Proposition 6. *For any $k \geq 6$, GF is not competitive on Θ_k in hop metric.*

Note that the above kind of node sets is specially designed for theoretical interest and will not affect the practicality of GF in general. As seen in later experiments, the average-case stretches of GF on Yao and Theta graphs is actually only a little more than one, indicating a very good performance by GF.

5. Stretches in average case

We obtain the average-case stretches of GF on Yao graphs, Theta graphs, and DTs by experimenting with 1000 graphs and calculating the average *stretch*$_{GF,\,avg}(G)$ of all experimented graphs G for each graph type.

In our experiments, stretches in both hop and Euclidean metrics are measured. For convenience, we also call the stretch in hop metric the *hop stretch* and the stretch in Euclidean metric the *Euclidean stretch* later. Moreover, to help explain the results on stretches, we experiment on the *average node degrees* (denoted D_{avg} hereafter) of all three graph types. Since D_{avg} reflects how rich a graph is connected, it will shed some light on the phenomena exhibited by *stretch*. Note that, the D_{avg} of a graph equals $2e/n$, where e is the number of edges and n is the number of nodes in this graph.

Below, we first describe our experimental setup, and then present our results in the following order: *average node degree*, *average-case hop stretches*, and *average-case Euclidean stretches.*

5.1. Experiments setup

We developed software to measure and calculate the aforementioned metrics (*average node degree*, *hop stretch*, *Euclidean stretch*) on Yao graphs, Theta graphs and DTs. Our software is implemented using the Computational Geometry Algorithms Library (CGAL) [10], which provides basic APIs for geometric calculations. While CGAL provides the construction package for DTs but not Yao and Theta graphs, we developed the construction package for Yao and Theta graphs ourselves, and contributed our package to CGAL. After five rounds of reviews and revisions, our package is now integrated into CGAL and available at [28].

When comparing Yao and Theta graphs with DTs, we use $k = 6$ for Yao and Theta graphs, since six is the smallest k that makes those two graphs void-free. To provide an intuition to these three types of graphs (DT, Θ_6 and Y_6), examples of them constructed by our software on a 14-node set are illustrated in Fig. 8, in which light green lines are added to show cone boundaries at the nodes where the resulting Theta graph and Yao graph have different edges. From this figure, we can observe that (1) Θ_6 and Y_6 generally have richer

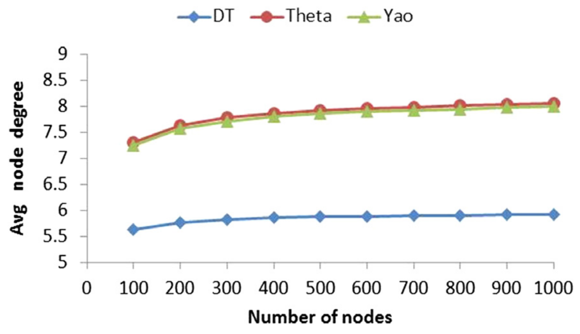


Fig. 10. Average node degrees with different n's and $k = 6$.

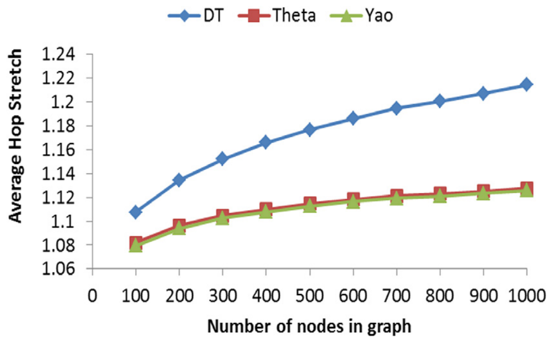
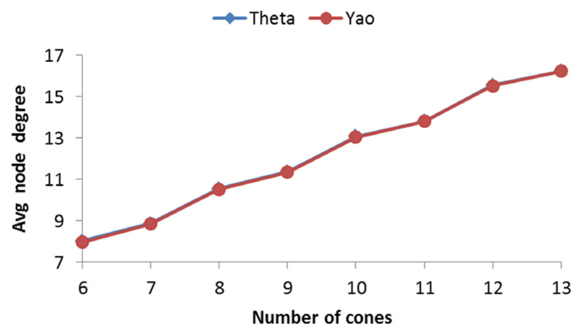


Fig. 12. Average hop stretch with different n's and $k = 6$.



	6	7	8	9	10	11	12	13
Theta	1.336	1.267	1.321	1.266	1.307	1.2564	1.296	1.2496
Yao	1.326	1.263	1.315	1.262	1.304	1.2557	1.294	1.2492

Fig. 11. Average node degrees with different k's and $n = 800$.

connectivity than a DT, and can be nonplanar graphs (i.e., having crossing edges); (2) the differences between Θ_6 and Y_6 on the same set of nodes generally only occur on a small portion of edges.

In our experiments, the nodes are generated according to uniform distribution in a square area. Note that the uniform distribution is a common distribution assumed among the research works [15,21,23,25,32,39] in this area to obtain general results. In our experiments on aforementioned metrics, we first fix k to 6 and vary the number of nodes (denoted by n hereafter) to observe the influence of n, and then we fix n to 800 and vary k to observe 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.

5.2. Average node degrees

Before presenting the empirical results, we first give the known theoretical results on the D_{avg} in DT, Theta graph and Yao graph. What is in common among these three graphs is that all their D_{avg} approach a constant with the increase of the node number in the graphs.

- In particular, for a DT, we first have the following formula holds: $e = 3n - h - 3$ [2], where e is the number of edges and h is the number of convex hull edges in this DT. Then, since $D_{avg} = 2e/n$, the D_{avg} of a DT approaches 6 with the increase of n.
- For Theta graph, Morin and Verdonschot proved the following two asymptotic formulae when n approaches infinite [24]. These two formulae establish the approximate linear relationship between D_{avg} and k, showing that D_{avg} approaches a constant for a fixed k. Moreover, they interestingly reveal that the coefficient for an even k is larger than that for an odd k. Since in the Theta graph definition presented in Section 1.1, the only way that makes D_{avg} less than $2k$ is the removal of duplicate edges, this difference between even and odd k's reflects that an even k sees less duplicate edges than an odd k in Theta graph.

$$D_{avg} \approx (2 - \pi\sqrt{3}/9) \cdot k \approx 1.40 \cdot k \quad \textit{for an even } k \geq 4 \tag{2}$$

$$D_{avg} \leq (2 - 2\arctan(\frac{1}{3})) \cdot k \approx 1.36 \cdot k \quad \textit{for an odd } k \geq 5 \tag{3}$$

- For Yao graph, Morin and Verdonschot [24] showed that its D_{avg} is similar to that of Theta graph, but were unable to give the closed-form expression.

Our experiments aim to confirm and demonstrate the above theoretical results, especially for Yao graph in which the D_{avg} has no known closed-form expression. Specifically, Fig. 10 plots the D_{avg} for DT, Θ_6 and Y_6 with $n = 100, 200, \ldots, 900$ and 1000 respectively. From this figure, we do observe that (1) the D_{avg} of DT approaches 6; (2) for $k = 6$, the D_{avg} of Yao graph is almost the same as that of Theta graph, and both are roughly 1.4 times the D_{avg} of DT, thus complying with the formula (2).

Fig. 11 plots the D_{avg} of Yao and Theta graphs under different k's, mainly showing the following. First, the data lines of Yao and Theta graphs are almost identical, reflecting the D_{avg} of Yao and Theta graphs are very close under different k's. Since the data line of Theta graph is almost hidden by that of Yao graph, we add a table inside Fig. 11 to present the actual data behind the data lines. Second, the data lines of both graphs exhibit a zigzag shape, which is consistent with formulae (2) and (3). This reflects that *the parity of k has an impact on the connectivity of Yao and Theta graphs, and this zigzag of D_{avg} will help explain the zigzag of the average-case stretches in the coming plots.*

5.3. Average-case hop stretches

This subsection presents experimental results on average-case hop stretches by GF on three types of graphs: Theta graph, Yao graph and DT. Specifically, Fig. 12 plots the results with $k = 6$ and $n = 100, 200, \ldots, 900$ and 1000 respectively. This figure mainly shows that (1) for all experimented n, GF achieves a slightly smaller stretch on Yao graphs than on Theta graphs; and (2) GF has a much larger stretch on DTs than on Yao and Theta graphs; this can be

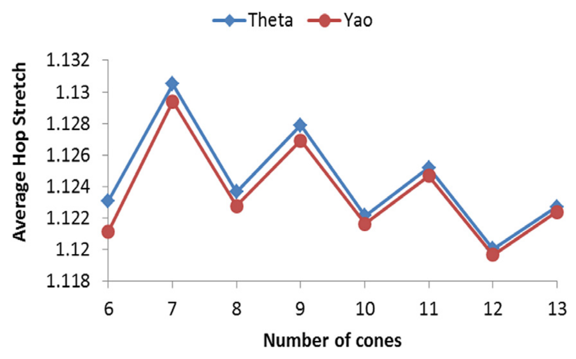


Fig. 13. Average hop stretch with different k's and $n = 800$.

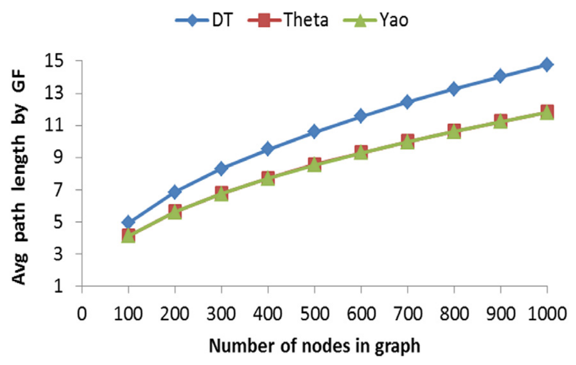


Fig. 14. Average hop path length by GF with different n's and $k = 6$.

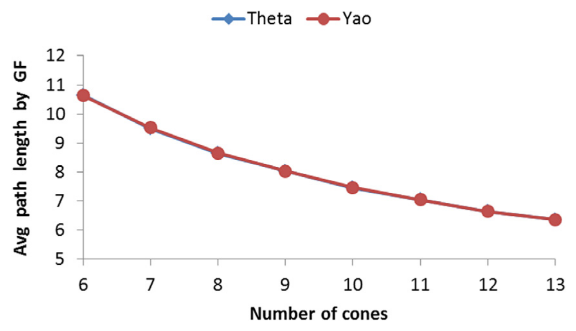


Fig. 15. Average hop path length by GF with different k's and $n = 800$.

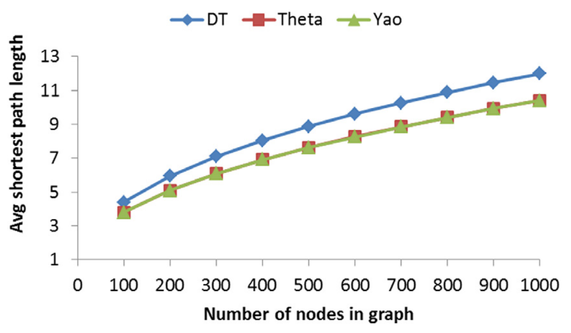


Fig. 16. Average hop shortest path length with different n's and $k = 6$.

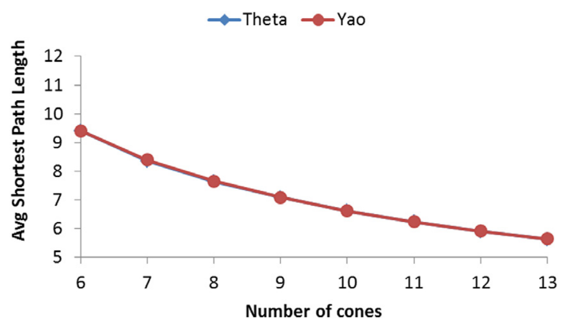


Fig. 17. Average hop shortest path length with different k's and $n = 800$.

explained by that DTs have smaller D_{avg} than Yao and Theta graphs as shown in the previous subsection.

The average-case hop stretches of GF on Theta and Yao graphs with $k = 6, 7, \ldots, 13$ are plotted in Fig. 13. This figure mainly shows the following.

First, the average hop stretches on both Yao and Theta graphs exhibit a zigzag shape, bearing a relatively large value at an odd k and a relatively small value at an even k. This is contradictory to the intuition that when k increases, the connectivity of Yao and Theta graphs will get richer and hence the average stretch should decrease monotonically. So this reflects that *the parity of k has a significant impact on the performance of GF on Yao and Theta graphs.* This impact links to the zigzag shape of D_{avg} depicted in 0, where an even k generally gives rise to a larger D_{avg} and hence richer connectivity than an odd k.

Second, when k equals 6 or 12, GF achieves a better hop stretch than other experimented k's on both Yao and Theta graphs. This implies that *six is an efficient cone number for Yao and Theta graphs.* To generalize the discussion in a broader context, *this is similar in principle to the phenomenon that bees build honeycomb cells with a hexagonal shape to minimize the consumption of construction materials* [35]. Considering that $k = 12$ involves too much cost in establishing edges in Yao and Theta graphs, $k = 6$ should be a suitable choice for using Yao and Theta graphs to establish network topologies.

According to formula (1), $stretch_{GF}(G, s, t)$ is a ratio between $GF(G, s, t)$ and $SP(G, s, t)$, so a question arising naturally is: whether the average path length by GF ($GF_{avg}(G)$) and the average shortest path length ($SP_{avg}(G)$) on Yao and Theta graphs also exhibit a zigzag shape when k increases. To investigate this, we measured the $GF_{avg}(G)$ and the $SP_{avg}(G)$ for Yao and Theta graphs in our experiments, and also measured them for DTs for comparison purpose. The measurement results on $GF_{avg}(G)$ are shown in Fig. 14 ($k = 6$ and varying n) and Fig. 15 ($n = 800$ and varying k), and the results on $SP_{avg}(G)$ are shown in Fig. 16 ($k = 6$ and varying n) and Fig. 17 ($n = 800$ and varying k).

From Figs. 14 and 16, we can observe that both the $GF_{avg}(G)$ and the $SP_{avg}(G)$ of all three graph types increase when n grows from 100 to 1000. This is consistent with the intuition that more nodes result in longer paths in graphs. Also from these two figures, we see that both Yao and Theta graphs have smaller $GF_{avg}(G)$ and $SP_{avg}(G)$ than DTs, showing that richer connectivity contributes to shorter path lengths.

From Figs. 15 and 17, we can observe that both the $GF_{avg}(G)$ and the $SP_{avg}(G)$ decrease smoothly when k increases from 6 to 13, thus no zigzag shape is spotted. This reflects that the parity of k does not have a noticeable influence on $GF_{avg}(G)$ and the $SP_{avg}(G)$, although it does have a very apparent influence on $stretch_{GF,avg}(G)$ according to Fig. 13.

5.4. Average-case Euclidean stretches

This subsection presents experimental results on average-case Euclidean stretches by GF on Theta graph, Yao graph and DT.

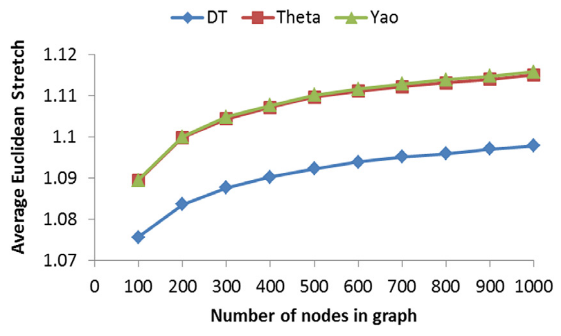


Fig. 18. Average Euclidean stretch with different n's and $k = 6$.

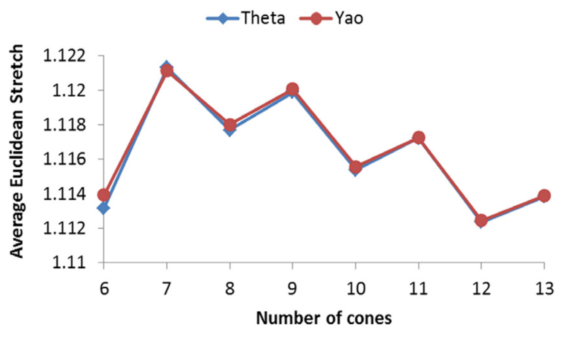


Fig. 19. Average Euclidean stretch with different k's and $n = 800$.

Specifically, the results on these three types of graphs for $n = 100, 200, \ldots, 900$ and 1000 and $k = 6$ are plotted in Fig. 18, which shows very different phenomena from those on hop stretch shown in Fig. 12. **First**, GF achieves almost identical average-case Euclidean stretches on Yao and Theta graphs, which is different from the case of hop stretch in which the GF clearly performs a little better on Yao graph. **Second**, though DTs have smaller D_{avg} than Yao and Theta graphs, GF performs better on DTs than on Yao and Theta graphs, which is opposite to the case of hop stretch in which GF performs better on Yao and Theta graphs than on DTs. This performance difference interestingly reflects that *the DT graph structure is more beneficial for GF to perform in Euclidean stretch than in hop stretch, and the impact of the DT graph structure on GF performance in Euclidean stretch overshadows the impact of the D_{avg} of DT.*

The average-case Euclidean stretches of GF on Theta and Yao graphs with $k = 6, 7, \ldots, 13$ and $n = 800$ are plotted in Fig. 19. This figure shows very similar phenomena to those on hop stretches shown in Fig. 13. First, the average-case Euclidean stretches on both Yao and Theta graphs exhibit a zigzag shape. Second, when k equals 6 or 12, GF achieves a better Euclidean stretch than other experimented k's on both Yao and Theta graphs. The comments to these phenomena will be the same as those to the hop stretches.

Similar to what we did for the hop stretches, we present a further study on the impact of the parity of k here. For the Euclidean stretch, there exist two related metrics: *absolute stretch* and *native stretch*. Both metrics are also important in practice, and we will look at whether *absolute stretch* and *native stretch* exhibit a zigzag shape when k changes between even and odd numbers.

First, the *absolute stretch* by a routing algorithm Γ from s to t in a graph G, denoted by $abs\_stretch_{\Gamma}\,(G, s, t)$, is defined as:

$$abs\_stretch_{\Gamma}(G, s, t) = \frac{\Gamma(G, s, t)\text{ in Euclidean}}{d(s, t)} \tag{4}$$

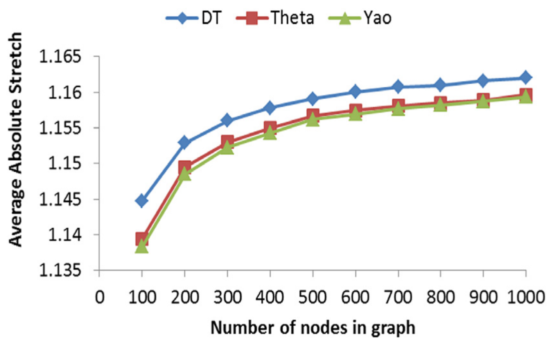


Fig. 20. Average absolute stretch with different n's and $k = 6$.

From this definition, we see that *absolute stretch* is another metric for evaluating the performance of a routing algorithm besides the *stretch* described in Section 4. Further, the *average absolute stretch* by Γ on a graph G, denoted by $abs\_stretch_{\Gamma,avg}(G)$, is defined as the average $abs\_stretch_{\Gamma}(G,\ s,\ t)$ of all (s, t) pairs in G.

Second, the *native stretch* from s to t in a graph G, denoted by $nat\_stretch\,(G, s, t)$, is defined as:

$$nat\_stretch(G, s, t) = \frac{SP(G, s, t)\text{ in Euclidean}}{d(s, t)} \tag{5}$$

From this definition, we see that *native stretch* is regardless of routing algorithms but a reflection of a graph's structure. Further, the *average native stretch* of a graph G, denoted by $nat\_stretch_{avg}\,(G)$, is defined as the average $nat\_stretch\,(G,\ s,\ t)$ of all (s, t) pairs in G; the *maximum native stretch* of a graph G, denoted by $nat\_stretch_{max}(G)$, is defined as the maximum $nat\_stretch\,(G, s, t)$ of all (s, t) pairs in G. As a worthy note, for a type of graphs, if the $nat\_stretch_{max}(G)$ of every graph instance G has a constant upper bound c, this type of graphs is called a c-spanner, and c is called the *spanning ratio* (or *stretch factor*) of this type of graphs [18]. For example, DT is known to be a 1.998-spanner [37], Y_6 is known to be a 5.8-spanner [1], and Θ_6 is known to be a 2-spanner [3]. And determining the tight (i.e., the minimum) spanning ratio for a type of graphs is a long-existing research area [18].

It is easy to observe that the following relationship holds among *Euclidean stretch*, *absolute stretch* and *native stretch*:

$$Euclidean\ stretch_{\Gamma}(G, s, t) = \frac{abs\_stretch_{\Gamma}(G, s, t)}{nat\_stretch(G, s, t)} \tag{6}$$

Thus, *absolute stretch* and *native stretch* not only reflect the routing performance and the graph structure, but also help understand the behavior of the *Euclidean stretch*. Therefore, we also measured the $abs\_stretch_{GF,avg}(G)$ and the $nat\_stretch_{avg}(G)$ for Yao graph, Theta graph and DT in our experiments. The measurement results on $abs\_stretch_{GF,avg}(G)$ are shown in Fig. 20 ($k = 6$ and varying n) and Fig. 21 ($n = 800$ and varying k), and the results on $nat\_stretch_{avg}(G)$ are shown in Fig. 22 ($k = 6$ and varying n) and Fig. 23 ($n = 800$ and varying k).

In Fig. 20, both Yao and Theta graphs see a smaller *absolute stretch* than DT; and in Fig. 22, both Yao and Theta graphs see a smaller *native stretch* than DT. These two facts are consistent with the intuition that larger D_{avg} results in smaller *absolute stretch* and *native stretch*. However, recall that Fig. 18 shows both Yao and Theta graphs actually see a greater Euclidean *stretch* than DT. This reflects that *although the impact of DT's graph structure outweighs the impact of D_{avg} on Euclidean stretch, this outweighing does not happen on absolute stretch and native stretch.*

Also, Fig. 20 shows that Yao graph sees a slightly smaller *absolute stretch* than Theta graph; and Fig. 22 shows that Yao graph has a slightly smaller *native stretch* than Theta graph. Recall that, in

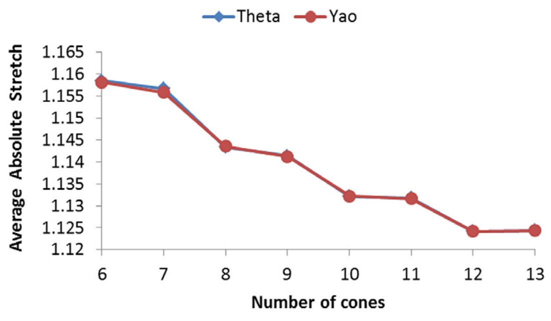


Fig. 21. Average absolute stretch with different k's and $n = 800$.

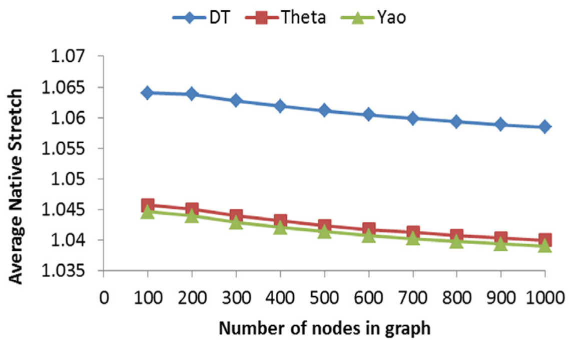


Fig. 22. Average native stretch with different n's and $k = 6$.

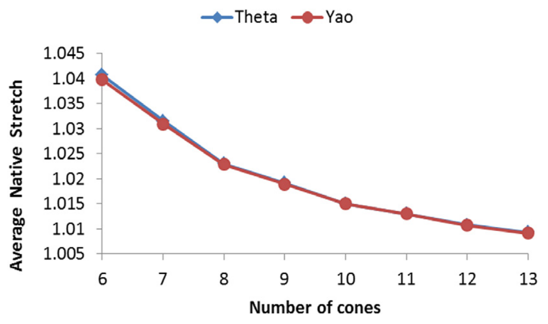


Fig. 23. Average native stretch with different k's and $n = 800$.

Fig. 18, we cannot tell on which graph GF performs better in terms of Euclidean *stretch*, but here we have Fig. 20 and Fig. 22 as rescue to demonstrate that Yao graph is better than Theta graph in terms of *absolute stretch* and *native stretch*.

Furthermore, Fig. 22 contributes a hint on determining the tight spanning ratio of Y_6. As aforementioned, the current smallest spanning ratio known for Y_6 is 5.8 [1], which is much larger than the spanning ratio of Θ_6 (equal to 2, known to be tight [3]). Since Fig. 22 shows that Y_6 actually has a smaller $nat\_stretch_{avg}(G)$ than Θ_6, it suggests that the spanning ratio of Y_6 may also be improved to 2.

Finally, from Fig. 21, we observe that the absolute stretches of both Yao and Theta graphs exhibit a slight zigzag shape but do *decrease strictly* with the growth of k, which shows that the parity of k has an impact on the absolute stretches, but the impact is not as strong as that on the stretch. And from Fig. 23, we observe that the native stretches of both Yao and Theta graphs decrease strictly with the growth of k without exhibiting a zigzag shape. This shows that the parity of k does not have an apparent impact on the native stretch. Overall, the results from these two figures comply with the intuition that larger k gives rise to smaller absolute stretch and native stretch.

5.5. Summary of experimental results

Since this section presented many experimental results, a summary is provided here to list the main ones. Within the summary, the guidelines for exploiting Yao graph, Theta graph and DT as network topologies are also given when possible.

- In general, *the GF algorithm performs well on both Yao and Theta graphs*, with average-case stretch less than 1.14 in both hop and Euclidean metrics on all experimented parameters in this paper.
- For both hop and Euclidean stretches, for both Yao and Theta graphs, GF generally has smaller stretches when k is even than when k is odd (e.g., $k = 8$ sees a smaller stretch than $k = 9$, etc.), so *when selecting cone numbers, an even number should be favored over an odd number*.
- For both hop and Euclidean stretches, for both Yao and Theta graphs, when k is a multiple of six, GF achieves the smallest stretches among nearby k's (e.g., $k = 6$ sees smaller stretches than $k = 7, 8, 9, 10$ and 11, but fails to beat $k = 12$). Also considering that $k = 6$ is the smallest cone number enabling Yao and Theta graphs void-free and involving less edges than $k = 7$–11, $k = 6$ is probably the best cone number to adopt in constructing network topologies.
- For hop stretch, GF performs slightly better on Yao graphs than on Theta graphs, and performs the worst on DTs. So *in terms of hop stretch, Yao graph is the best choice as network topologies among these three types of graphs*.
- For Euclidean stretch, surprisingly, though Yao and Theta graphs with $k = 6$ have larger D_{avg} than DT, both of them incur larger stretches than DT. So *DT is the best choice as network topologies in terms of Euclidean stretch*. If DT is excluded, the decision between Yao and Theta graphs cannot be made by comparing their average Euclidean stretches because GF performs almost the same on these two graphs. But *if the absolute stretch and the native stretch are considered, GF performs slightly better on Yao graph in both metrics*.

6. Related work

Since the study presented in this paper spans the following three areas: (1) the void-free property of geometric graphs, (2) the competitiveness of routing algorithms and (3) the experimental study of average-case stretches, the related works in these three areas are of interest to this paper. While we have overviewed the related work for the area (1) in the Introduction section, this section will focus on the related works for the areas (2) and (3).

The area (2) is a hard research area, because not many results on competitive routing algorithms exist so far. Below we try to review existing major results, which are essentially sporadic in that each result is only applicable to a particular type of graph. Since two metrics are generally considered for the competitiveness in the literature: hop metric and Euclidean metric, we organize our survey based on these two metrics below.

For the hop metric, [9] proved that no *online* competitive routing algorithms exist for DTs (here *online* means that a forwarding node only knows its 1-hop neighborhood information). Accordingly, [9] suggested that competitive algorithms under the hop metric are harder to obtain than their Euclidean counterparts. Despite the difficulty, some competitive algorithms did appear for other types of graphs. For example, [19] presented a competitive routing algorithm for the graphs in which nodes are located in a narrow strip; and [15] proposed a polynomial-time algorithm to embed Combinatorial Unit Disk Graphs into $O(\log^2 n)$-dimensional space such that GF is competitive. As stated in [15] itself, the proposed algorithm can also be viewed as an adapted GF algorithm that is competitive on Combinatorial Unit Disk Graphs. Different from [15] and [16,19] did not achieve a constant upper bound on the worst-case stretch of the face routing algorithm [20] on finite undirected graphs, but proved an upper bound expressed by the number of nodes and the number of edges in the graph.

For the Euclidean metric, [26] described the θ-routing algorithm for Theta graphs, which achieves a competitive ratio of $1/(1 - 2sin(\theta/2))$ for $k \geq 7$, where θ is the cone angle. This competitive ratio of the θ-routing algorithm was later improved slightly in [4], which divides k into four categories: $4n + 2$, $4n + 3$, $4n + 4$ and $4n + 5$ (here n is an integer ≥ 1) and gives specific competitive ratios for different categories of k. For Yao graphs, we do not see existing works on competitive algorithms. For DTs, [8] first proved that GF is not competitive on DTs and then gave an algorithm called Parallel Voronoi Routing that is competitive on DTs. For other types of graphs, [6] gave a routing algorithm that is competitive on a type of spanner graphs with maximum degree 12; and [5] gave a competitive routing algorithm on the so-called half-θ_6-graphs.

Our paper studies the competitiveness of GF on Theta and Yao graphs, which has no known results from previous work. We are successful in showing that GF is not competitive in hop metric on these two types of graphs. In Euclidean metric, we cannot determine the competitiveness of GF on Yao and Theta graphs in this paper, and will study it as our next-step work. We note that studying the competitiveness of routing algorithms is a challenging area, and there are several open problems in this area as noted in [4,7].

The area (3), the experimental study of average-case stretches, also only sees sporadic studies. Aforementioned works [15,19] conducted experiments on the average-case stretches of their proposed routing algorithms on their exploited graphs respectively in hop metric. The average-case Euclidean stretch by GF on DTs is first experimented in [13]. After this, [8] presented comparisons on Euclidean stretches by GF and five other routing algorithms on DTs. A recent work [30] presented comparisons on both hop and Euclidean stretches by GF and six other routing algorithms on DTs. *The experimental work in this paper is the first to study both hop and Euclidean stretches by GF on Theta and Yao graphs.*

7. Conclusions

Since GF is an important routing algorithm for wireless networks, this paper studied how well Yao and Theta graphs support GF, and presented both theoretical and experimental results. Theoretically, this paper proved that (1) when $k < 6$, Yao graph or Theta graph is not void-free on certain node sets in the plane; (2) when $k \geq 6$, both Yao and Theta graphs are void-free on any node set in the plane; and (3) GF is not competitive on Yao graph or Theta graph in hop metric when $k \geq 6$. Experimentally, this paper showed several meaningful results that can guide the selection of graphs as network topologies. A summary of these results and guidelines is provided in Section 1.5.

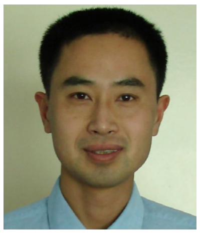
While we have shown that GF is not competitive on Yao and Theta graphs in hop metric, we conjecture that GF is competitive on Yao and Theta graphs in Euclidean metric when $k \geq 6$. However, we are unable to complete the proof at this stage and will investigate it in our future work.

References

[1] L.F. Barba, et al., New and improved spanning ratios for Yao graphs, J. Comput. Geom. (2015).
[2] M.d. Berg, O. Cheong, M.v. Kreveld, M. Overmars, Computational Geometry: Algorithms and Applications, third ed., Springer-Verlag, New York, 2008, pp. xii, 386 p.
[3] N. Bonichon, C. Gavoille, N. Hanusse, D. Ilcinkas, Connections between theta-graphs, Delaunay triangulations, and orthogonal surfaces, in: The 36th International Workshop on Graph-Theoretic Concepts in Computer Science, 2010, pp. 266–278.
[4] P. Bose, J.-L.D. Carufel, P. Morin, A.v. Renssen, S. Verdonschot, Towards tight bounds on theta-graphs, Theoret. Comput. Sci. (2015).
[5] P. Bose, R. Fagerberg, A.v. Renssen, S. Verdonschot, Competitive routing in the half-θ6-graph, in: ACM-SIAM Symposium on Discrete Algorithms, 2012, pp. 1319–1328.
[6] P. Bose, R. Fagerberg, A.v. Renssen, S. Verdonschot, Competitive routing on a bounded-degree plane spanner, in: Canadian Conference on Computational Geometry, 2012.
[7] P. Bose, R. Fagerberg, A.v. Renssen, S. Verdonschot, Competitive local routing with constraints, J. Comput. Geom. 8 (1) (2017) 125–152.
[8] P. Bose, P. Morin, Online routing in triangulations, SIAM J. Comput. 33 (4) (2004) 937–951.
[9] P. Bose, et al., Online routing in convex subdivisions, Int. J. Comput. Geom. 12 (4) (2002) 283–295.
[10] CGAL, Computational geometry algorithms library, 2016. http://www.cgal.org/.
[11] D. Chen, P.K. Varshney, A survey of void handling techniques for geographic routing in wireless networks, IEEE Commun. Surv. Tutor. 9 (1) (2007) 50–67.
[12] K. Clarkson, Approximation algorithms for shortest path motion planning, in: ACM Symposium on Theory of Computing, 1987, pp. 56–65.
[13] P. Cucka, N.S. Netanyahu, A. Rosenfeld, Learning in navigation: Goal finding in graphs, Int. J. Pattern Recognit. Artif. Intell. 10 (5) (1996) 429–446.
[14] D. Eppstein, M.T. Goodrich, Succinct greedy geometric routing using hyperbolic geometry, IEEE Trans. Comput. 60 (11) (2011) 1571–1580.
[15] R. Flury, S.V. Pemmaraju, R. Wattenhofer, Greedy routing with bounded stretch, in: IEEE INFOCOM, 2009, pp. 1737–1745.
[16] H. Frey, I. Stojmenovic, On delivery guarantees and worst-case forwarding bounds of elementary face routing components in ad hoc and sensor networks, IEEE Trans. Comput. (2010) 1224–1238.
[17] K.R. Gabriel, R.R. Sokal, A new statistical approach to geographic variation analysis, Syst. Zool. 18 (3) (1969) 259–278.
[18] J. Gao, L.J. Guibas, J. Hershberger, L. Zhang, A. Zhu, Geometric spanners for routing in mobile networks, IEEE J. Sel. Areas Commun. 23 (1) (2005) 174–185.
[19] J. Gao, L. Zhang, Load-balanced short-path routing in wireless networks, IEEE Trans. Parallel Distrib. Syst. 17 (4) (2006) 377–388.
[20] F. Kuhn, R. Wattenhofer, Y. Zhang, A. Zollinger, Geometric ad-hoc routing: Of theory and practice, in: ACM PODC, 2003, pp. 63–72.
[21] F. Li, Z. Chen, 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.
[22] X.-Y. Li, G. Calinescu, P.-J. Wan, Y. Wang, Localized Delaunay triangulation with applications in wireless ad hoc networks, IEEE Trans. Parallel Distrib. Syst. 14 (10) (2003) 1035–1047.
[23] X.-Y. Li, W.-Z. Song, W. Wang, A unified energy-efficient topology for unicast and broadcast, in: ACM Int. Conference on Mobile Computing and Networking, MobiCom, 2005.
[24] P. Morin, S. Verdonschot, On the average number of edges in theta graphs, in: SIAM Meeting on Analytic Algorithmics & Combinatorics, ANALCO, 2014, pp. 1–20.
[25] S. Poduri, S. Pattem, B. Krishnamachari, G.S. Sukhatme, Using local geometry for tunable topology control in sensor networks, IEEE Trans. Mob. Comput. 8 (2009) 218–230.
[26] J. Ruppert, R. Seidel, Approximating the d-dimensional complete Euclidean graph, in: Canadian Conference on Computational Geometry, 1991.
[27] W. Si, B. Scholz, J. Gudmundsson, G. Mao, R. Boreli, A.Y. Zomaya, On graphs supporting greedy forwarding for directional wireless networks, in: IEEE International Conference on Communications, ICC, 2012.
[28] W. Si, Q. Tse, Cone-Based Spanners User Manual, 2016. http://doc.cgal.org/latest/Cone_spanners_2/index.html#Chapter_ConeBasedSpanners.
[29] W. Si, Q. Tse, G. Mao, A.Y. Zomaya, How well do Yao graph and theta graph support greedy forwarding?, in: IEEE GLOBECOM, 2014.
[30] W. Si, A.Y. Zomaya, New memoryless online routing algorithms for Delaunay triangulations, IEEE Trans. Parallel Distrib. Syst. (2012) 1520–1527.
[31] W. Si, A.Y. Zomaya, S. Selvakennedy, Ageometric deployment and routing scheme for directional wireless mesh networks, IEEE Trans. Comput. 63 (6) (2014).
[32] W.-Z. Song, Y. Wang, X.-Y. Li, 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.
[33] G.T. Toussaint, The relative neighborhood graph of a finite planar set, Pattern Recognit. 12 (1980) 261–268.
[34] Y. Wang, X.-Y. Li, Efficient Delaunay-based localized routing for wireless sensor networks, Int. J. Commun. Syst. 20 (7) (2007) 767–789.
[35] Wikipedia, Hexagon, 2015. http://en.wikipedia.org/wiki/Hexagon.
[36] Wikipedia, Unit disk graphs, 2016. https://en.wikipedia.org/wiki/Unit_disk_graph.
[37] G. Xia, The stretch factor of the Delaunay triangulation is less than 1.998, SIAM J. Comput. 42 (4) (2013) 1620–1659.

[38] A.C. Yao, On constructing minimum spanning trees in k-dimensional spaces and related problems, SIAM J. Comput. (1982).

[39] Z. Yu, J. Teng, X. Bai, D. Xuan, W. Jia, Connected coverage in wireless networks with directional antennas, in: IEEE INFOCOM, 2011, pp. 2264–2272.



Weisheng Si is currently a lecturer in the School of Computing, Engineering and Mathematics, Western Sydney University. Prior to this, he was a postdoctoral researcher at National ICT Australia (NICTA). He received his Ph.D., M.S., and B.S. degrees in computer science from University of Sydney (Australia), University of Virginia (USA), and Peking University (China) respectively. His research interests include routing and topology control in wireless networks, graph theory, software-defined networks, and data center networks.

Quincy Tse received both his Ph.D. and Bachelor degrees in computer science from University of Sydney (Australia). He ever worked as a research assistant in Western Sydney University. He is now working in industry.

Guoqiang Mao joined the University of Technology Sydney in February 2014 as Professor of Wireless Networking and Director of Center for Real-time Information Networks. Before that, he was with the School of Electrical and Information Engineering, the University of Sydney. He has published about 200 papers in international conferences and journals, which have been cited more than 5000 times. He is an editor of the IEEE Transactions on Wireless Communications (since 2014), IEEE Transactions on Vehicular Technology (since 2010) and received "Top Editor" award for outstanding contributions to the IEEE Transactions on Vehicular Technology in 2011, 2014 and 2015. He is a co-chair of IEEE Intelligent Transport Systems Society Technical Committee on Communication Networks. He has served as a chair, co-chair and TPC member in a large number of international conferences. He is a Fellow of IEEE and IET. His research interest includes intelligent transport systems, applied graph theory and its applications in telecommunications, Internet of Things, wireless sensor networks, wireless localization techniques and network performance analysis.

Albert Y. Zomaya is currently the Chair Professor of High Performance Computing & Networking in the School of Information Technologies, University of Sydney. He is also the Director of the Centre for Distributed and High Performance Computing which was established in late 2009. Professor Zomaya was an Australian Research Council Professorial Fellow during 2010–2014. He published more than 550 scientific papers and articles and is author, co-author or editor of more than 20 books.

He is the Founding Editor in Chief of the IEEE Transactions on Sustainable Computing and previously he served as Editor in Chief for the IEEE Transactions on Computers (2011–2014). Currently, Professor Zomaya serves as an associate editor for 22 leading journals, such as, the ACM Computing Surveys, IEEE Transactions on Computational Social Systems, IEEE Transactions on Cloud Computing, and Journal of Parallel and Distributed Computing. He delivered more than 180 keynote addresses, invited seminars, and media briefings and has been actively involved, in a variety of capacities, in the organization of more than 700 national and international conferences.

Professor Zomaya is the recipient of the IEEE Technical Committee on Parallel Processing Outstanding Service Award (2011), the IEEE Technical Committee on Scalable Computing Medal for Excellence in Scalable Computing (2011), and the IEEE Computer Society Technical Achievement Award (2014), and the ACM MSWIM Reginald A. Fessenden Award (2017). He is a Chartered Engineer, a Fellow of AAAS, IEEE, IET (UK), and an IEEE Computer Society's Golden Core member. Professor Zomaya's research interests lie in parallel and distributed computing, networking, and complex systems.