On Graphs Supporting Greedy Forwarding for Directional Wireless Networks

Weisheng Si [1,4], Bernhard Scholz [2], Joachim Gudmundsson [2], Guoqiang Mao [1,2], Roksana Boreli [1,3], Albert Zomaya [2]
National ICT Australia Ltd., Australia [1]
School of IT, University of Sydney, Australia [2]
School of EE&IT, University of NSW, Australia [3]
School of Computing, Engineering, and Mathematics, University of Western Sydney, Australia [4]
Email: Firstname.Lastname@nicta.com.au, Firstname.Lastname@sydney.edu.au

***Abstract*— Greedy forwarding is an efficient and scalable geographic routing algorithm for wireless networks. To guarantee the success of greedy forwarding, many research efforts assign virtual coordinates to nodes to obtain a greedy embedding of the network. Different from these existing efforts, this paper presents an approach that enables greedy forwarding to succeed in directional wireless networks by selecting links in the network instead of assigning virtual coordinates to the nodes. Specifically, this paper studies the following problem: given a set of nodes on the Euclidean plane, how can we add a minimum number of point-to-point links, such that the greedy forwarding algorithm succeeds on the resulting network. The motivation for studying this problem is that each point-to-point link in directional wireless networks is realized by a pair of directional antennas, so minimizing the number of links will reduce the network installation cost. This paper first presents the properties of the graphs supporting greedy forwarding, and then solves the above problem optimally by Integer Linear Programming and also sub-optimally by a polynomial-time 3-approximation algorithm. Finally, this paper compares the polynomial-time algorithm with the optimal solution, showing that the polynomial-time algorithm can actually generate within 1.1 times the number of links found by the optimal solution in most cases.**

Keywords: topology control; geographic routing; greedy forwarding; nearest neighbor graphs; directional wireless networks.

I. INTRODUCTION

Geographic routing (also known as "geometric routing" or "geo-routing") algorithms exploit the position information of nodes to route packets in the wireless networks modeled by geometric graphs. If a geographic routing algorithm $\mathcal{A}$ can route a packet from any source s to any destination t in a geometric graph G, $\mathcal{A}$ is said to *succeed* on G [1] or G is said to support $\mathcal{A}$ [2]. The 1980s saw the earliest geographic routing methods [3-5], which are all based on the following greedy algorithm: when a node forwards a packet with destination t, it sends this packet to its neighbor that is closest to t. This greedy algorithm is generally called Greedy Forwarding [6] or Greedy Routing [7] in the literature. The advantages of this algorithm include: (1) low computation complexity at a node, (2) low space overhead for a packet, and (3) the capability of achieving short path length in number of hops [8].

Despite the aforementioned advantages, greedy forwarding does not succeed on a graph that contains the *void* scenario [6], in which for a certain destination t, a node u does not have a neighbor with a smaller distance to t than u. To overcome the *void* scenario, routing algorithms such as [6, 9-11] are proposed to enhance greedy forwarding with the face routing mechanism [12]. That is, when no *void* is encountered, these algorithms use greedy forwarding to advance a packet; otherwise, the face routing mechanism is used to turn around the *void*. However, the face routing mechanism not only increases the computation complexity at a node, but also increases the space overhead for a packet, which has to record additional information such as the position of the source node, the entrance edge of the current face being explored, etc.

As a result, it is desirable to design a network without the *void* scenario. Motivated by this, many research efforts [13-19] have investigated the approach of assigning virtual coordinates to the nodes to achieve a *greedy embedding* [14], which maps the original network topology $G(V, E)$, where V represents the node set and E the link set, to a metric space $(V', dist)$ with the following property: (1) V' is the same set of nodes as V but with the new virtual coordinates; (2) $dist$ is a distance function on V' such that for any pair of distinct nodes s and t in V', node s always has a neighbor w such that $dist(w, t) < dist(s, t)$. For instance, ref. [13, 16] map network topologies to a Euclidean space and ref. [15] map network topologies to a hyperbolic space to obtain a *greedy embedding*. A common feature of these efforts is that the links in E are unchanged during the embedding, since for the packets to be received successfully, the existence of a link between node u and node v should be determined by the distance between u and v in the original space and their transmission range.

In this paper, we also study the problem of obtaining a geometric graph that supports greedy forwarding, but we use the approach of changing the links among nodes instead of changing the position information of nodes. The motivation for our approach comes from the emerging wireless networks in which each node is equipped with directional antennas [20-21]. These directional wireless networks typically have the following characteristics: (1) nodes have fixed positions (not mobile) in a large area; (2) the establishment of a wireless point-to-point link requires a directional antenna at each of its two end nodes, so the number of links significantly affect the network installation cost; (3) the transmission power of the directional antennas can be adjusted and a relay device can be added if necessary, so a link between any two nodes is possible. Therefore, for such directional wireless networks, it is meaningful to construct a geometric graph supporting greedy forwarding such that the number of links is minimized to reduce the network installation cost. Another reason that we do not use the virtual coordinate approach is that it degrades the quality of paths found by greedy forwarding [15], since virtual coordinates do not reflect the real position of nodes.

For the ease of discussion, we call the graphs that support greedy forwarding 'greedy forwarding graph' later. Formally, 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 a *greedy forwarding graph (GFG)*. Here we say that node w is a neighbor of node u if there is a point-to-point link between node w and node u in G. Note that, though the concept of GFG originates from the concept of *greedy embedding*, GFG defines a type of graph, while *greedy embedding* defines a type of mapping.

Now we give our **Problem Statement** as follows: *Given an arbitrary set of nodes with known coordinates in the Euclidean plane, how can we add minimum number of point-to-point links among the nodes, such that the resulting geometric graph is a GFG*. Hereafter, we use GFG(V) to denote a GFG on a node set V, and $\text{GFG}_{\min}(V)$ to denote the GFG(V) with the minimum number of links, and $E_{\min}(V)$ to denote the link set in $\text{GFG}_{\min}(V)$.

The contributions of this paper include:

- Six interesting observations about GFGs are presented. These observations reveal the properties of GFGs and the relationships between GFGs and other well-known graphs exploited in geographic routing, such as Relative Neighborhood Graph [22], Gabriel Graph [23], and Delaunay Triangulation [24].
- An optimal solution to the minimum GFG problem by Integer Linear Programming is given. Though it can have an exponential worst-case runtime, it is simple to implement by using the AMPL/CPLEX solver [25].
- Based on the observations presented for GFGs, a polynomial-time algorithm that computes a sub-optimal minimum GFG is presented, and is proved a 3-approximation algorithm.
- We compare the performance of the polynomial-time algorithm with that of the optimal solution, showing that the polynomial-time algorithm can generally achieve less than 1.1 times the number of links found by the optimal solution. Experiments are also conducted to show that the GFGs obtained by the polynomial algorithm provide high path quality for the greedy forwarding algorithm.

II. OBSERVATIONS ON GFGS

In this section, we present six observations about GFGs' properties and relationships with other types of graphs. Hereafter, the notation "graph $G' \subseteq$ graph G " means that G' is a subgraph of G.

Observation 1. Any GFG is connected.

Proof: In a GFG, since the greedy forwarding algorithm can always find a path from any node s to any node t, this observation holds. □

Observation 1 implies that any GFG(V) has a spanning tree as its subgraph. Since the number of edges[1] in a spanning tree on V is n-1, where n is the number of nodes in V, a lower bound for $|E_{\min}(V)|$ is n-1. This lower bound is tight for some V. For example, when the nodes in V are along a straight line, the chain graph on V contains n-1 edges and is a GFG.

For a node set V, the $\text{GFG}_{\min}(V)$ can have multiple graph instances and some of them can be planar. The question is whether there always exists a $\text{GFG}_{\min}(V)$ that is planar. By conducting experiments on a large number of node sets, we are able to indentify a negative example for this question.

Observation 2. There exists a node set V such that any $\text{GFG}_{\min}(V)$ is not planar.

Proof: Figure 1 gives a set V with six nodes for which the $\text{GFG}_{\min}(V)$ has no planar instances. The graph on the left gives the only possible $\text{GFG}_{\min}(V)$, which has an edge-crossing. The two graphs in the middle and on the right both are planar GFGs, but have one more edges than the graph on the left. □

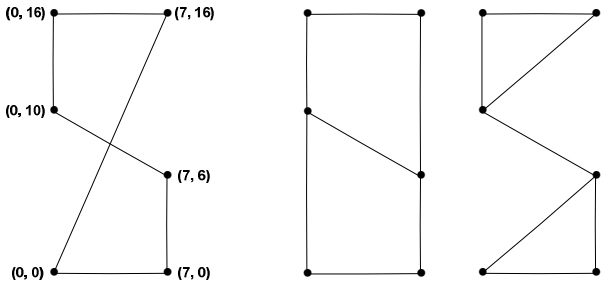

Figure 1. A 6-node set on which no minimum GFG is planar

Before giving Observation 3, we recall that a *Delaunay triangulation* on a node set V (denoted by DT(V)) is a triangulation graph on V such that no nodes are contained in the interior of the circumcircle of any of its triangular faces [24].

Observation 3. Given an arbitrary node set V, the DT(V) is a GFG(V).

Proof: It is proved in [2] that in a DT(V), every node has a neighbor node closer to any destination node, so a DT(V) is a GFG(V). □

Since a DT(V) has $3n$-k-3 edges, where k is the number of convex hull edges on V [24], an upper bound for $|E_{\min}(V)|$ is $3n$-k-3. This upper bound is tight for some V. For example, when the nodes in V are placed to form adjacent equilateral triangles (see Figure 2), the DT(V) is the same as the $\text{GFG}_{\min}(V)$. Note that all edges in Figure 2 have to be present to form a GFG, since a GFG requires that for any node pair (u, v), u has a neighbor *strictly* closer to v, which is essential for avoiding routing loops.

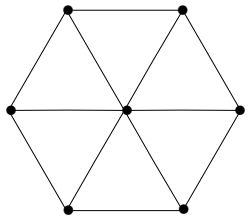

Figure 2. A DT that is also the minimum GFG

Given a node set V, it is known that the Euclidean Minimum Spanning Tree on V (denoted by EMST(V)), the Relative Neighborhood Graph on V (denoted by RNG(V)), the Gabriel Graph on V (denoted by GG(V)), and DT(V) have the following relationship [24]:

$$\text{EMST}(V) \subseteq \text{RNG}(V) \subseteq \text{GG}(V) \subseteq \text{DT}(V)$$

With Observation 3, it is natural to ask whether the EMST(V), RNG(V), and GG(V) are also GFG(V)'s. Unfortunately, we have Observation 4 below, which reveals that only the DT(V)

[1] In this paper, 'link' and 'edge' are used interchangeably.

among these four types of graphs are guaranteed to be a GFG(V).

Observation 4. There exists a node set V such that the EMST(V), RNG(V), and GG(V) are not a GFG(V).

Proof: Figure 3 gives such a node set V with five nodes a, b, c, d, and e, for which the GG(V) is not a GFG(V). Since EMST(V) $\subseteq$ RNG(V) $\subseteq$ GG(V), the EMST(V) and RNG(V) are not a GFG(V) either. □

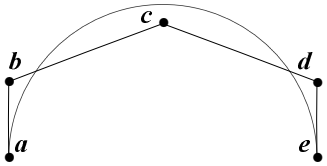

Figure 3. A Gabriel Graph that is not GFG

Before giving Observation 5, we recall that a *nearest neighbor graph* on a node set V (denoted by NNG(V)) is obtained by connecting each node in V with its nearest neighbor [24]. Note that this paper assumes that (1) if a node p has more than one nearest neighbors with equal distance, all of them are connected to p and (2) the NNG(V) is considered in the undirected sense.

Observation 5. Given an arbitrary node set V, the NNG(V) is a subgraph of any GFG(V).

Proof: We only need to prove that every edge in NNG(V) is in any GFG(V). Suppose a GFG(V) does not contain an edge in NNG(V) which connects node p to its nearest neighbor q. Consider p as the destination. Clearly q does not have a neighbor w with d(w, p) < d(q, p), which contradicts to the definition of a GFG(V). □

It is known that NNG(V) $\subseteq$ GG(V) holds for any node set V [24]. We also have mentioned EMST(V) $\subseteq$ RNG(V) $\subseteq$ GG(V). Thus, it is natural to ask whether EMST(V), RNG(V), and GG(V) are also subgraphs of any GFG(V). Unfortunately, Figure 4 gives a GFG on a node set V with four nodes for which the EMST(V) is not a subgraph of the given GFG(V). Thus, only NNG(V) among these four types of graphs is guaranteed to be a subgraph of any GFG(V).

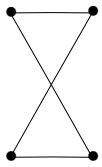

Figure 4. An exemplar GFG of which the EMST is not a subgraph

Before giving Observation 6, we recall the following definitions from [24]. In a geometric graph, for an edge pq with two end nodes p and q, the *bisector of p and q* is defined as the perpendicular bisector of the edge pq. For the two open half planes delimited by this bisector, the one containing p is denoted $h(p, q)$, and the other one containing q is denoted $h(q, p)$. Note that 'open' here means that the bisector is excluded from the half plane. With these definitions, we introduce a new concept called the *closer region* of a node, which is used in Observation 6.

Definition 1. Given a geometric graph G, the *closer region* of a node p in G is defined as the intersection of all $h(p, u)$'s, where u is a neighbor of p in G.

This concept is illustrated in Figure 5, where the given geometric graph contains four nodes of a chain shape, and the bisectors of its edges are depicted by dotted lines. In this graph, the closer region of node b is the shaded sector extending to the infinity, and the closer region of node d is the entire $h(d, c)$.

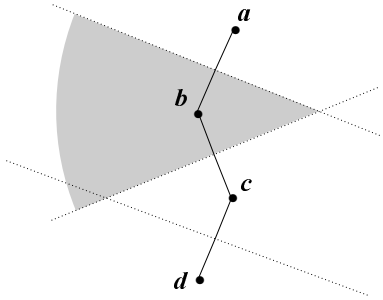

Figure 5. Concept of closer region

Note that the closer region concept differs from the Voronoi region concept [24] in the following aspects:

1) The closer regions can overlap with each other, while the Voronoi regions do not overlap.
2) The closer regions are defined on a geometric graph, while the Voronoi regions are defined on a set of nodes.
3) Given a geometric graph $G(V, E)$ and the Voronoi diagram on V, the closer region of a node p regarding G contains the Voronoi region of p regarding V. This is because the Voronoi region of p is the intersection of all $h(p, q)$'s, where q is any other node in V.

Now we are ready to give Observation 6 as follows.

Observation 6. A necessary and sufficient condition for a geometric graph G to be a GFG is that for every node v in G, the closer region of v and its boundary do not contain any other node in G.

Proof: We prove the necessary condition by contradiction. Suppose in a GFG, the closer region of a node p contains another node q. Since a neighbor of p cannot lie in p's closer region, q is not a neighbor of p. Now consider q as the destination. Since q is in the closer region of p and is not a neighbor of p, p does not have a neighbor w with a smaller distance to q. This contradicts to the definition of a GFG.

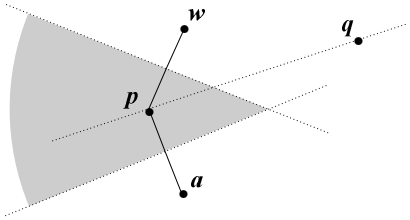

Figure 6. How to find a closer neighbor to destination

We prove the sufficient condition by construction. For any node pair p and q in a geometric graph G, we find a neighbor w of p such that d(w, q) < d(p, q) by the following method. As shown in Figure 6, draw the supporting line pq. Since q is not in the closer region of p, pq must intersect with a bisector that bounds p's closer region. Consider p's neighbor w that shares this bisector with p. Since w and q are in the same side of this bisector while p is in the other side, we have d(w, q) < d(p, q). Therefore, G is a GFG. □

III. THE OPTIMAL SOLUTION: AN INTEGER LINEAR PROGRAM

To obtain optimal solution to the minimum GFG problem, we formulate the problem as an Integer Linear Program (ILP). Though an ILP is in general NP-hard, its solution can be found by using available ILP solvers. In our case, we use the AMPL/CPLEX solver [25].

Let V denote the given node set, and $U = V \times V - \{(v,v): v \in V\}$. Our goal is to find a minimum set of links $E \subseteq V \times V$, such that for each source/destination pair $(u,v) \in U$, u has an adjacent link $(u,w) \in E$ with $d(w,v) < d(u,v)$. For each link $(u,v) \in V \times V$ we introduce a binary variable $x_{(u,v)}$ in the integer linear program that determines whether link (u,v) exists in E, i.e., the binary variable $x_{(u,v)}$ is 1 if $(u,v) \in E$ and 0 otherwise. Since the GFGs are undirected graphs, for any link (u,v), variable $x_{(v,u)}$ is equal to variable $x_{(u,v)}$. Thus, we obtain the following ILP:

$$\begin{aligned} \min. \quad & \frac{1}{2}\sum_{(u,v)\in U} x_{(u,v)} \\ s.t. \quad & \sum_{w\in V} \delta(u,w,v)\cdot x_{(u,w)} \geq 1 \qquad \forall (u,v)\in U \\ & x_{(u,v)} = x_{(v,u)} \qquad \forall (u,v)\in V\times V \\ & x_{(u,v)} \in \{0,1\} \qquad \forall (u,v)\in V\times V \end{aligned}$$

Note that there is a coefficient 1/2 in the objective function, since $x_{(u,v)}$ equals $x_{(v,u)}$. The first constraint in the ILP enforces that the solution is a GFG. The parameter $\delta(u,w,v)$ is defined as,

$$\delta(u,w,v) = \begin{cases} 1, & if\ d(w,v) < d(u,v) \\ 0, & otherwise \end{cases}$$

That is, the function is one if for a pair (u,v) the neighbor w of u is closer to v than u; otherwise zero. If there is a link (u,w) in E and w is closer to v, the term $\delta(u,w,v) \bullet x_{(u,w)}$ becomes one. Since at least one of the neighbors needs to be closer, the sum of the terms over all neighbors has to be $\geqslant 1$. The second and third constraints in the ILP enforce that the resulting graph is undirected and the variables are binary variables.

Figure 7 illustrates an exemplar $GFG_{min}(V)$ obtained by solving the above ILP on a 100-node set V, with its nodes randomly distributed in a disk area.

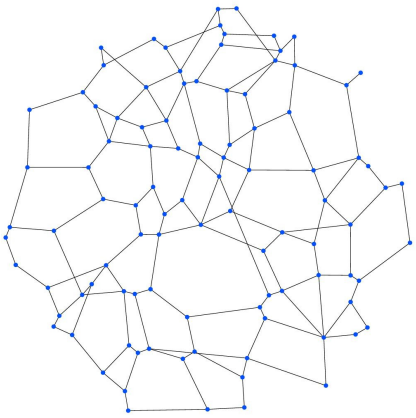

Figure 7. The minimum GFG on a 100-node set

IV. A POLYNOMIAL-TIME APPROXIMATION ALGORITHM

We give a polynomial-time algorithm to get a suboptimal solution for the minimum GFG problem. As established in the previous observations, given a set of nodes V, the DT(V) is a GFG(V) and the NNG(V) is a subgraph of any GFG(V). Thus, the basic idea of our algorithm is to calculate the DT(V) and the NNG(V) first, and then attempt to remove the edges that are in DT(V) but not in NNG(V), i.e., DT(V) - NNG(V). In testing whether an edge in DT(V) - NNG(V) can be removed, Observation 6 is used. This algorithm is detailed below.

Input: a set of nodes V with known (x, y)-coordinates

Output: a GFG that approximates the minimum GFG

1. Calculate DT(V).
2. Based on DT(V), calculate NNG(V) and determine the list of edges that are in DT(V) - NNG(V).
3. Go through the list of edges in DT(V) - NNG(V). For each edge uv (u and v are the two end nodes of this edge), test whether it can be removed such that the remaining graph is still a GFG(V). The test is based on Observation 6: for both u and v, check whether they are still the only nodes lying in their closer regions. If uv can be removed, remove it; otherwise not.
4. The remaining graph is the output GFG(V).

We have the following result about the computational complexity of this algorithm.

Result 1. The above algorithm has a complexity of $O(n^2)$, where n is the number of given nodes.

Proof: For the above algorithm, step 1 takes $O(n\log(n))$ by using an existing efficient algorithm given in [24]. Step 2 takes $O(n)$ according to the algorithm described in [26]. Step 3 takes $O(n^2)$, since there are $O(n)$ edges in DT(V) - NNG(V), and for each edge uv, the test involves $O(n)$ examinations of whether a node is in the closer regions of u and v, and each examination takes constant time in average since we prune from a DT and each node in a DT has six neighbors in average. Therefore, the overall complexity of the above algorithm is $O(n^2)$. •

We have the following result about the performance of this algorithm.

Result 2. The above algorithm is a 3-approximation algorithm in terms of the number of edges.

Proof: Recall that given a node set V, the DT(V) has $3n-k-3$ edges [24], where k is the number of convex hull edges on V and n is the number of nodes in V. Since the algorithm starts pruning edges from a DT(V), the obtained GFG at most has $3n-k-3$ edges. Moreover, as stated in Observation 2, the lower bound for $|E_{min}(V)|$ is $n-1$, so the above algorithm at most generates 3 times the number of edges found by the optimal solution. •

To help understanding this algorithm, Figures 8-10 illustrate the DT(V), the NNG(V), and the suboptimal GFG(V) obtained by our algorithm on the same node set used in Figure 7.

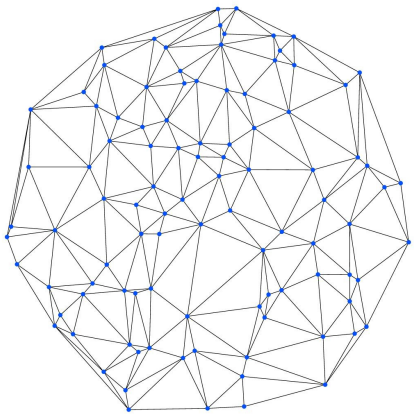

Figure 8. The DT on a 100-node set

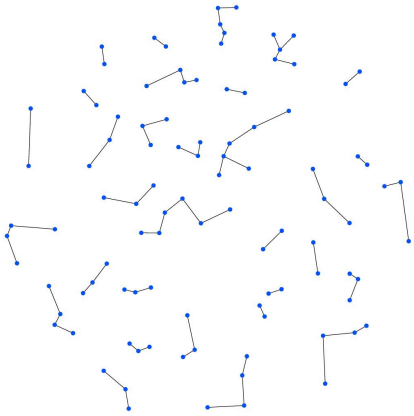

Figure 9. The NNG on a 100-node set

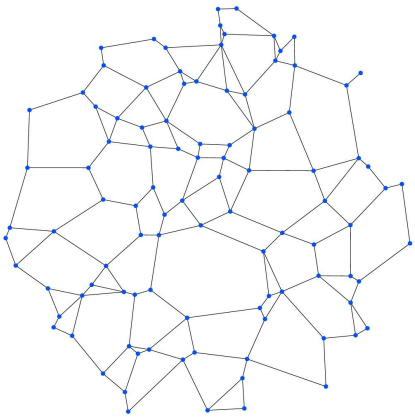

Figure 10. The GFG obtained by our algorithm on a 100-node set

Comparing the GFG obtained by our algorithm (Figure 10) with the minimum GFG obtained by ILP (Figure 7), it can be seen that our algorithm only generates slightly more number of edges than the optimal solution. This is further verified by our evaluations below.

V. EVALUATION

In this section, we compare our polynomial-time algorithm with the optimal solution in terms of the number edges generated to construct the GFGs. We also evaluate the path quality of the GFGs generated by our polynomial-time algorithm for the greedy forwarding algorithm. The performance metric for path quality we use is known as 'stretch', which is to be detailed later. We developed computer programs for these evaluation purposes. Our DT construction implementation is based on the open-source software Triangle [27], and we use the AMPL/CPLEX solver [25] to solve the ILP formulated in Section III.

We generated nodes according to a uniform distribution in a disk area with radius equal to 500m. Then we solve our ILP and run our polynomial-time algorithm on the generated node sets to obtain the optimal and suboptimal GFGs respectively. We conduct experiments on the networks with 50, 100, 150, 200, 300, and 400 nodes. For each fixed node number, we conduct 200 experiments with each node placement randomly generated. Since the number of our experiments is large, the ranges of confidence intervals are within 0.3% of the mean values and we plot the standard deviations instead in the figures to come. In our experiments, we do not vary node density under a fixed node number, since changing node density on the plane is a dilation transformation (stretching or shrinking with respect to a center point) [28], which will not affect our experiment results on the number of edges or stretches.

A. *Number of edges in the GFGs*

To compare the number of edges generated by our polynomial-time algorithm with that of the optimal solution, we calculate the following performance factor:

$$performance\ \ factor = \frac{\#\ of\ \ edges\ in\ the\ GFGs\ by\ our\ algorithm}{\#\ of\ \ edges\ in\ the\ minimum\ GFGs}$$

Figure 11 plots the average value and the standard deviation of this performance factor at different node numbers with 200 experiments. This figure shows that (1) for all the node numbers experimented, this factor is less than 1.1, reflecting that the polynomial time algorithm only uses slightly more number of edges than the optimal solution; (2) the standard deviation is within 1.5% of the average value, reflecting that the performance of the polynomial-time algorithm is stable.

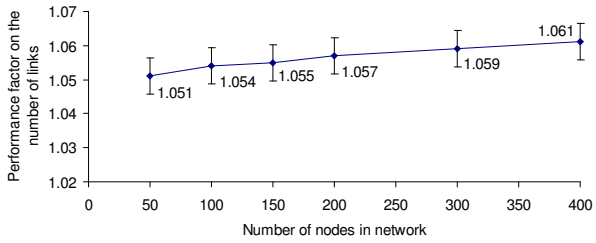

Figure 11. Performance factor on the number of edges at different node numbers

B. *Stretches by greedy forwarding on generated GFGs*

Now we evaluate how well the GFGs generated by our polynomial-time algorithm support greedy forwarding by measuring two kinds of stretches. We define **absolute stretch** by greedy forwarding for a source/destination pair (s, t) in a GFG as the ratio between *the Euclidean length of the path found by greedy forwarding from s to t* and *the Euclidean distance from s to t*. *Absolute stretch* reflects the quality of the path found by greedy forwarding compared with the direct distance between s and t. We define **relative stretch** as the ratio between *the Euclidean length of the path found by greedy forwarding from s to t* and *the Euclidean length of the shortest path from s to t*. *Relative stretch* reflects how close the path found by greedy forwarding is to the real shortest path in the generated GFGs. For each GFG genearated by our algorithm, we calculate the

average *absolute stretch* and *relative stretch* for all source/destination pairs in this GFG by greedy forwarding. We further calculate the average *absolute stretch* and *relative stretch* over 200 GFGs for each node number.

Figure 12 plots the average value and the standard deviation of *absolute stretch* at different node numbers. This figure shows that (1) greedy forwarding generally can achieve absolute stretches less than 1.25 in GFGs generated by our polynomial time algorithm; (2) the standard deviation of absolute stretch is within 2% of the average value, reflecting absolute stretch by greedy forwarding is stable in our experimental setting; (3) absolute stretch increases slowly with the growth of node number, since greedy forwarding only considers 1-hop information and hence performs worse when the network size is larger.

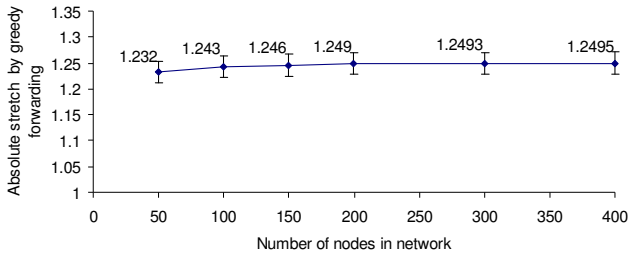

Figure 12. Absolute stretch by greedy forwarding at different number of nodes

Figure 13 plots the average value and the standard deviation of relative stretch at different node numbers. This figure shows that (1) greedy forwarding generally can achieve relative stretches less than 1.1 in GFGs generated by our polynomial time algorithm; (2) the standard deviation of relative stretch by greedy forwarding is within 2% of the average value; (3) relative stretch also increases slowly with the growth of node number.

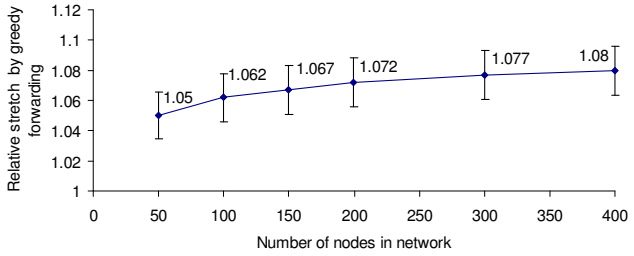

Figure 13. Relative stretch by greedy forwarding at different number of nodes

VI. CONCLUSION

This paper considers the problem of generating minimum GFGs for the wireless networks with directional antennas. We first presented six observations regarding GFGs, which reveal the properties of GFGs and the relationships between GFGs and other well-known graphs exploited in geographic routing. We then give an optimal solution by ILP as well as a polynomial time 3-approximation algorithm for the minimum GFG problem. The experimental results show that (1) our polynomial-time algorithm can actually produce within 1.1 times the number of links found by the optimal solution in general and (2) greedy forwarding achieves small stretches on our generated GFGs. Note that the observations and algorithms presented in this paper have a fundamental nature, so they can also be applied to other scenarios modelled by geometric graphs, such as path finding in robotics.

ACKNOWLEDGMENT

This research work has been supported by funding from National ICT Australia (NICTA).

REFERENCE

[1] P. Bose, *et al.*, "Bounding the Locality of Distributed Routing Algorithms", in *ACM PODC,* pp. 250-259, 2009.
[2] P. Bose and P. Morin, "Online Routing in Triangulations", *SIAM Journal of Computing,* vol. 33, pp. 937-951, 2004.
[3] H. Takagi and L. Kleinrock, "Optimal transmission ranges for randomly distributed packet radio terminals", *IEEE Transactions on Communications,* vol. 32, pp. 246–257, 1984.
[4] T. Hou and V. Li, "Transmission range control in multihop packet radio networks", *IEEE Transactions on Communications,* vol. 34, pp. 38–44, 1986.
[5] G. Finn, "Routing and addressing problems in large metropolitan-scale internetworks," USC/ISI, Tech. Rep. ISI/RR-87-180,1987.
[6] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks ", in *ACM MobiCom*, pp. 243-254, 2000.
[7] P. BOSE and P. MORIN, "Routing with Guaranteed Delivery in Ad Hoc Wireless Networks", *Wireless Networks,* vol. 7, pp. 609–616, 2001.
[8] P. Cucka, *et al.*, "Learning in navigation: Goal finding in graphs", *International Journal of Pattern Recognition and Artificial Intelligence,* vol. 10, pp. 429-446, 1996.
[9] F. Kuhn, *et al.*, "Worst Case Optimal and Average Case Efficient Geometric AdHoc Routing", in *ACM MobiHoc*, pp. 267 - 278, 2003.
[10] B. Leong, *et al.*, "Path Vector Face Routing: Geographic Routing with Local Face Information", in *ICNP*, pp. 147--158, 2005.
[11] F. Kuhn, *et al.*, "Geometric ad-hoc routing: of theory and practice", in *ACM PODC*, pp. 63--72, 2003.
[12] E. Kranakis, *et al.*, "Compass routing on geometric networks", in *Canadian Conference on Computational Geometry*, pp. 51–54, 1999.
[13] A. Rao, *et al.*, "Geographic routing without location information", in *ACM MobiCom*, pp. pages 96–108, 2003.
[14] C. H. Papadimitriou and D. Ratajczak, "On a conjecture related to geometric routing", *Theoretical Computer Science,* vol. 344, pp. 3-14, 2005.
[15] R. Kleinberg, "Geographic Routing Using Hyperbolic Space", in *IEEE Infocom*, 2007.
[16] B. Leong, *et al.*, "Greedy Virtual Coordinates for Geographic Routing", in *IEEE ICNP*, pp. pp 71-80, 2007.
[17] C. Westphal and G. Pei, "Scalable Routing Via Greedy Embedding", in *IEEE INFOCOM*, pp. 2826-2830, 2009.
[18] M. Goodrich and D. Strash, "Succinct Greedy Geometric Routing in the Euclidean Plane", in *The 20th International Symposium on Algorithms and Computation*, pp. 781-791, 2009.
[19] R. Sarkar, *et al.*, "Greedy routing with guaranteed delivery using Ricci flows", in *International Conference on Information Processing in Sensor Networks (IPSN)*, 2009.
[20] Q. Liu, *et al.*, "Topology control for multi-channel multi-radio wireless mesh networks using directional antennas", *Wireless Networks,* vol. 17, pp. 41-51, 2011.
[21] W. Si, *et al.*, "A Geometric Deployment and Routing Scheme for Directional Wireless Mesh Networks", *IEEE Transactions on Computers,* in print, 2011.
[22] G. T. Toussaint, "The relative neighborhood graph of a finite planar set", *Pattern Recognition,* pp. 261-268, 1980.
[23] K. R. Gabriel and R. R. Sokal, "A new statistical approach to geographic variation analysis", *Systematic Zoology,* pp. 259-278, 1969.
[24] M. d. Berg, *et al.*, *Computational geometry: algorithms and applications*, 3rd ed. New York, Springer, 2008.
[25] R. Fourer, *et al.*, *AMPL: A Modeling Language for Mathematical Programming*, Brooks/Cole Publishing Company, 2002.
[26] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, 1987.
[27] J. R. Shewchuk, *Triangle version 1.6,* Available: http://www.cs.cmu.edu/~quake/triangle.html, 2005.
[28] WolframMathworld, *Dilation transformation,* Available: http://mathworld.wolfram.com/Dilation.html, 2011.