Improved “GOAFR” Algorithm for Geometric Ad-Hoc Routing

December 21, 2002

Abstract

In this short paper, we present GOAFRPC, an improvement of the GOAFR algorithm for geometric ad-hoc routing. We show that GOAFRPC is average-case more efficient, while it keeps the asymptotical worse-case optimality of GOAFR.

1 Introduction

The geometric ad-hoc routing problem was originated from mobile ad-hoc networks. Each node in a mobile network has a *geometric location* and a *transmission range*—a disk centered at this node with fixed *communication radius*. For simplicity, we assume that all nodes have the same communication radii. Two nodes can communicate directly if and only if they are in mutual transmission range, i.e. their Euclidean distance is less than or equal to the communication radius. The *routing* problem arises if two nodes cannot communicate directly. In this case, their messages need to be relayed through a series of intermediate nodes such that any two consecutive nodes can communicate directly.

Routing is performed in a distributed manner, i.e. each intermediate node makes its own decision and there is no centralized control in the network. A node makes decisions based on its own local information and on the information carried by the message. The decisions consist of the next node for routing and the change of the information in the message. A message only has $O(1)$ spaces to carry such information.

In an ad-hoc network, a node only knows about its *neighbors*, the nodes with which it can communicate directly. A node has no knowledge about the topology of other parts of the network. Specifically, there is no traditional routing table inside a node.

For a geometric ad-hoc routing algorithm, a node knows the geometric location of its neighbors and of itself. Furthermore, the sender knows the geometric location of the destination and stores this information in the message for the routing.

There are various algorithms for this problem. The simple greedy algorithm behaves well in dense networks. But it fails for very simple configurations. The *Face Routing* (FR) algorithm[3] is the first one that guarantees success if the source and the destination are connected. However, the worst-case cost of FR is proportional to the size of the network. The first algorithm that can compete with the best route in the worst-case is the *Adaptive Face Routing* (AFR) algorithm[2]. Moreover, by a lower bound argument, AFR is shown to be asymptotically worst-case optimal. But AFR is not average-case efficient. The *Greedy Other Adaptive Face Routing* (GOAFR) algorithm[1] combines the greedy and a variance of AFR called *Other Adaptive Face Routing* (OAFR) algorithm and achieves both worst-case optimality and average-case efficiency. For a more complete survey, see [1].

Based on the simulation results of GOAFR (see [1]), there are several ways to further improve the average-case performance. This paper will present one of them. The name of the new algorithm is GOAFRPC, where “PC” denotes “partially close”.

In Section 2, we state model and preliminaries. Section 3 will look at the original GOAFR algorithm. In Section 4, we will explain the GOAFRPC algorithm in detail and analyze its worst-case cost. The simulation results will be shown in Section 5 to illustrate the average-case behavior. And at last, we conclude the results in Section 6.

2 Model and Preliminaries

Without loss of generality, we assume the communication radius is the unit distance, i.e. 1. Then, we model the geometric routing problem by the *Unit Disk Graph* (UDG) $G = (V, E)$. A node $u \in V$ denotes the node as well as the geometric location in the network. For any two nodes $u, v \in V$, there is an edge $e = (u, v) \in E$ if and only if u and v could communicate directly, i.e. if the Euclidean distance between u and v is less than or equal to 1.

The routing problem is to find a path from the source s to the destination t. The cost of a path is the sum of the costs of its edges. There are three measures for the cost of an edge e. They are the *hop metric* ($c_l(e) := 1$), the *Euclidean metric* ($c_d(e) := |e|$) and the *energy metric* ($c_e(e) := |e|^2$). Since $|e| \leq 1$, we have $c_d(e) = O(c_l(e))$ and $c_e(e) = O(c_l(e))$. Moreover, under the $\Omega(1)$ model (see below), these three metrics are equivalent up to a constant factor, i.e. $c_d(e) = \Theta(c_l(e))$ and $c_e(e) = \Theta(c_l(e))$. Since we only consider the $\Omega(1)$ model in GOAFRPC (this is also the model for AFR and GOAFR), we will focus our attention on the hop metric. Arguments for other metrics then holds automatically.

2.1 The $\Omega(1)$ Model

The $\Omega(1)$ model makes the following assumption. The Euclidean distance between any two nodes is bounded from below by a constant number, i.e. $|(u, v)| \geq d_0$, where $0 < d_0 \leq 1$ is a constant. The $\Omega(1)$ model features the following properties:

1. All three metrics mentioned above are equivalent up to a constant factor. See Lemma 3.1 in [2] for the proof.

2. The number of vertices inside a convex region is bounded from above by the area of the region, i.e. $|V \cap R| = O(area(R))$, where R is a convex region of any shape, e.g. a cycle, an ellipse or a polygon. See Lemma 4.2 in [2] for the proof.

3. The Unit Disk Graph is a sparse graph, i.e. $|E| = O(|V|)$ since the degree of a vertex is bounded by a constant.

2.2 Gabriel Graphs

In general, a Unit Disk Graph is not necessarily a planar graph, even under the $\Omega(1)$ model. However, planarity is essential in the series of face-based routing algorithms since planar graphs feature faces. To achieve planarity, we employ the *Gabriel Graph*. For a Unit Disk Graph $G = (V, E)$, the Gabriel Graph built on G is $G_G = (V, E_G)$, where $e \in E_G$ if and only if (a) $e \in E$, and (b) the cycle with e as diameter only contains the two endpoints of e. The Unit-Disk Gabriel Graph features the following properties.

1. The Gabriel Graph is a planar graph since it is a subgraph of the Delaunay Triangulation.

2. A Unit-disk Gabriel Graph can be computed locally. To find the adjacent edges, a node only needs to know the geometric locations of the nodes with distances less than or equal to 1.

3. The Gabriel Graph is connected if and only if the original Unit Disk Graph is connected.

4. Under the $\Omega(1)$ model, the shortest path in a Gabriel Graph is only by a constant factor longer than in the original Unit Disk Graph. See Lemma 4.4 in [2] for the proof.

All routing algorithms in this paper, i.e. Greedy, OFR, OAFR, GOAFR and GOAFRPC run on the Unit-Disk Gabriel Graph, under the $\Omega(1)$ model.

3 The GOAFR Algorithm

We will briefly review the key points of GOAFR in this section. For a complete description, see [1].

3.1 Greedy Routing

The greedy algorithm always picks the neighbor closest to t to be next node for routing. It can be easily stuck at some local minimum, i.e. no neighbor is closer to t than the current node.

3.2 Other Face Routing — OFR

Since GOAFR combines the Greedy algorithm and OAFR — a variance of AFR, we will directly introduce the corresponding variance of FR, called *Other Face Routing* (OFR) here.

OFR utilizes the face structure of planar graphs. The message is routed from s to t by traversing a series of face boundaries. We denote these faces by $f_1, f_2, \ldots, f_\ell$. Let $s_i (1 \le i \le \ell)$ be the starting node of the traversing in f_i ($s_1 = s$). At node s_i, OFR chooses f_i to be the face that contains the line segment $\overline{s_i t}$ in the immediate environment of s_i. Then the message is routed along the entire boundary of f_i. The aim is to find the best node on the boundary, i.e. the closest node to the destination t. When finished, the message returns to s_i, and we denote t_i the best node on the boundary of f_i. There are two possibilities:

1. t_i is s_i.
2. t_i is not s_i, i.e. $|\overline{t_i t}| < |\overline{s_i t}|$.

It can be proven that for a Unit-Disk Gabriel Graph, case 1 implies that s_i and t are disconnected, i.e. OFR guarantees to find a better node (case 2) if t is reachable. So, if the result is case 1, OFR simply reports "Destination Unreachable". If it is case 2, OFR chooses s_{i+1} to be t_i, the starting node for the next face. OFR routes the message along the boundary from s_i to s_{i+1} and starts to traverse the next face at s_{i+1}. From above, we can see that OFR always reaches t if t is indeed reachable. The cost of OFR is $O(|V|)$ since it visits each edge constant number of times.

3.3 Other Adaptive Face Routing — OAFR

The main reason for the high cost of OFR is that the message may traverse extremely huge face boundaries because we have no knowledge about the correct direction that should be taken. The *Other Adaptive Face Routing* (OAFR) algorithm tackles this problem by using a *bounding ellipse*.

We first describe the procedure of *Other Bounded Face Routing* (OBFR). We introduce a bounding ellipse. The two foci of the bounding ellipse are s and t. The length of its major axis is some given constant number c. In general, OBFR is OFR that runs on the subgraph within the bounding ellipse. The details are as follows. At each starting node s_i, OBFR determines f_i in the same way as OFR, i.e. f_i is chosen to be the face that contains $\overline{s_i t}$ in the immediate environment of s_i. When OBFR routes the message along the boundary, it also tries to find the best possible node. But it may not be able to explore the entire face boundary because of the bounding ellipse. If OBFR does not hit the bounding ellipse, its behavior is exactly the same as for OFR.

Let us consider the case that OBFR hits the bounding ellipse. When this happens for the first time, OBFR goes back to s_i and continues to explore the boundary in the other direction until it hits the bounding ellipse again. Note that it is impossible for OBFR to hit the bounding ellipse only once. As soon as OBFR hits the bounding ellipse for the second time, the best possible node by using OBFR has been found. The part of the boundary visited by OBFR is a chain. We also denote t_i the best node on this chain. There are again two possibilities.

1. t_i is s_i.
2. t_i is not s_i.

But this time, case 1 only implies that t is unreachable for the subgraph within the bounding ellipse. Is it because the bounding ellipse is too small, or because t is unreachable at all? We do not know. That is where OAFR comes in. We summarize the behavior of OBFR as follows.

1. The bounding ellipse is not hit and no better node is found: report "Destination Unreachable".
2. The bounding ellipse is hit twice and no better node is found: report "Destination Unreachable inside the Current Bounding Ellipse".
3. Otherwise: go to the best node t_i and start to traverse the next face.

Now assume we know the cost c^* of an optimal route. We let the major axis of the bounding ellipse be $c = c^*$, and run OBFR. Since the cost of any route that contains nodes outside the bounding ellipse is greater than c^*, the optimal route(s) must be completely inside the bounding ellipse. So, OBFR guarantees to reach the destination along a route completely inside the bounding ellipse. From Property 2 of the $\Omega(1)$ model, the total number of nodes inside the bounding ellipse is $O((c^*)^2)$. Hence, the cost of the route found by OBFR is $O((c^*)^2)$.

The problem is that we do not know c^*. OAFR first guesses the optimal cost, sets $c = 2|\overline{st}|$ and starts OBFR with this c value. If OBFR reports "Destination Unreachable inside the Current

Bounding Ellipse", OAFR doubles the value of c and continues OBFR from the face where it is detained. Since the value of c increases exponentially, the total cost of OAFR is still bounded by $O((c^*)^2)$.

3.4 GOAFR — Greedy+OAFR

GOAFR combines the Greedy Routing and OAFR, such that it is both average-case efficient and worst-case optimal. In general GOAFR does Greedy Routing as long as possible and only uses OAFR to tackle the local minima. The details of GOAFR are as follows. GOAFR also has a bounding ellipse. Initially, the length of the major axis is $c = 2|\overline{st}|$. The algorithm starts by Greedy Routing inside the bounding ellipse. There are two cases to interrupt a Greedy Phase.

1. The bounding ellipse is too small, i.e. the current node does have neighbors closer to t, but all such neighbors lie outside the bounding ellipse.

2. The current node is indeed a local minimum, i.e. it has no neighbor closer to t in the entire graph.

In the former case, we double the length of c, and continue the Greedy Routing inside the larger ellipse. In the latter case, we have to use OAFR. An OAFR Phase of GOAFR only traverses one face boundary to get around the local minimum. After that, GOAFR returns to the Greedy Routing immediately. The details of an OAFR Phase is the same as the original OAFR algorithm. It tries to find the best possible node inside the bounding ellipse and doubles the major axis when necessary. Note that the bounding ellipse never shrinks after GOAFR returns to the Greedy Routing.

The cost of all OAFR Phases is dominated by the original OAFR algorithm, i.e. $O((c^*)^2)$. So, we only need to analyze the Greedy Phases. Draw a cycle centered at t with radius $|\overline{st}|$. All nodes visited during Greedy Phases are completely inside this cycle. From Property 2 of the $\Omega(1)$ model, the number of nodes in this cycle is $O((|\overline{st}|)^2) = O((c^*)^2)$. So, the cost of all Greedy Phases is $O((c^*)^2)$ since Greedy Routing visits each node at most once. The total cost of GOAFR then follows.

4 The GOAFR$_{\text{PC}}$ Algorithm

We are now ready to improve the GOAFR algorithm. GOAFR is average-case efficient because it makes use of the greedy strategy. The algorithm only uses OAFR when it has to. Otherwise, it always use Greedy Routing. However, GOAFR explores the entire face boundary in an OAFR Phase. This is the starting point for our improvement. The idea is *not to look at the entire face boundary in an OAFR Phase.*

At the end of [1], another algorithm has already been proposed for the same purpose. The algorithm was called GOAFR$_{\text{FC}}$ where "FC" denotes "First Close". GOAFR$_{\text{FC}}$ falls back to the Greedy Routing immediately when meeting the first closer node to t in an OAFR Phase. The simulation results have shown evident improvement in average-case over the original GOAFR algorithm. But GOAFR$_{\text{FC}}$ loses the asymptotical optimality in the worst-case.

To understand why this happens, we need to look at the worst-case arguments for GOAFR again. It is easy to see that the cost of all Greedy Phases remains unchanged in GOAFR$_{\text{FC}}$. Let $c_1, c_2, \ldots, c_h$ denote the lengths of the major axes of the bounding ellipses from the beginning to the end, i.e.

$$c_i = 2^i|\overline{st}|, \quad 1 \le i \le h \tag{1}$$

and

$$\begin{aligned} c_h &\ge c^* &(2)\\ c_{h-1} &< c^* &(3)\end{aligned}$$

We divide the whole routing process into different periods according to the size of the bounding ellipse. Period i is the time interval when the routing is bounded by the ellipse of size c_i. The following lemma shows a sufficient condition for the worst-case optimality.

Lemma 1 *The cost of the algorithm is $O((c^*)^2)$, if in each period, the cost of all OAFR Phases on any face f is $O(|f|)$, where $|f|$ is the number of edges of f.*

Proof: Since the cost of all Greedy Phases is $O((c^*)^2)$, we only need to show the cost of all OAFR Phases is also $O((c^*)^2)$.

Consider period i. During this period, the major axis of the bounding ellipse is c_i. From Property

2 of the $\Omega(1)$ model, the number of nodes inside the bounding ellipse is $O(c_i^2)$. Since G is a planar graph, the number of edges inside the bounding ellipse is also $O(c_i^2)$. So, the total cost in this period is

$$\sum_f (\text{cost on } f) = \sum_f |f| = O(c_i^2)$$

Sum over all periods, we obtain the cost of all OAFR Phases

$$\sum_{i=1}^{h} c_i^2 = O(c_h^2)$$

From (1)–(3), we have $c_h^2 \le (2c^*)^2 = O((c^*)^2)$. The lemma follows. □

The OAFR Phases of GOAFR satisfies Lemma 1 since the original OAFR algorithm visits a face at most once when the bounding ellipse is fixed.

How about GOAFRFC? Fig. 8 in [1] showes a counterexample; GOAFRFC can visit a face many times in one period.

4.1 The GOAFRPC Algorithm

Like GOAFRFC, GOAFRPC is the same as GOAFR except for an additional criterion for OAFR Phases to return faster. The details are as follows. First, we introduce a constant r, the *exit-ratio.* Let s_i denote the local minimum where a new OAFR Phase starts. We maintain two variables p and q where

- p is the number of nodes closer to t than s_i in the current OAFR Phase.
- q is the he number of nodes farther from t than s_i in the current OAFR Phase.

Initially, when the message is at s_i, $p = q = 0$. When a node v occurs, if v has not been visited since the current OAFR Phase starts from s_i, we do the following.

- Increase p by 1, if v is closer to t than s_i
- Increase q by 1, if v is farther from t than s_i

If $p \ge rq$ at some node — r is the exit-ratio — the algorithm returns to the Greedy Routing. GOAFRPC also memorizes the best node, i.e. the closest node to t during an OAFR Phase. When the algorithm is about to return to the Greedy Routing, it first goes back to this best node and then returns.

We call the new algorithm GOAFRPC. "PC" or "Partially Closer" means that it returns to Greedy when a constant portion of the nodes are closer. It is easy to see that GOAFRPC is fully distributed and only needs $O(1)$ spaces in the message.

4.2 Worst-case Analysis

As in the GOAFRFC algorithm, GOAFRPC is possible to visit a face many times in one period. However, the following lemma shows that the cost on any face f is still $O(|f|)$ in each period.

Lemma 2 *In one period, the cost of all OAFR Phases on any face f is $(1 + 1/r)|f|$, where r is the exit-ratio.*

Proof: Consider the cost in one period. Assume there are k OAFR Phases is on f.

Let $s_1, s_2, \ldots, s_k$ denote the starting nodes of each time.

For the ith time, $(1 \le i \le k)$, we define

$$\begin{aligned}
T_i &:= \{v : v \text{ is visited in the } i\text{th time}\}\\
P_i &:= \{v \in T_i : |vt| < |s_i t|\}\\
Q_i &:= \{v \in T_i : |vt| > |s_i t|\}\\
R_i &:= \{v \in T_i : v \text{ is visited before}\}\\
N_i &:= \{v \in T_i : v \text{ is not visited before}\}
\end{aligned}$$

Clearly,

$$|T_i| = |P_i| + |Q_i| + 1 = |R_i| + |N_i|. \qquad (4)$$

Since GOAFRPC always chooses the closest node to t when switching to a Greedy Phase and the Greedy Phase can only get closer, we have

$$|\overline{s_i t}| < |\overline{vt}| \qquad \forall v \in \bigcap_{j<i} T_j$$

So, all re-visited nodes are farther from t than s_i, and all nodes closer to t have not been visited before, i.e.

$$\begin{aligned}
R_i &\subseteq Q_i\\
P_i &\subseteq N_i
\end{aligned}$$

Hence,

$$|R_i| \leq |Q_i| \tag{5}$$
$$|P_i| \leq |N_i| \tag{6}$$

From the algorithm, we have

$$|P_i| \geq r|Q_i| \tag{7}$$

Put (5)–(7) together, we obtain

$$|R_i| \leq |Q_i| \leq \frac{|P_i|}{r} \leq \frac{|N_i|}{r} \tag{8}$$

The intuitive meaning of (8) is that the re-visited nodes cannot be too many more than the new nodes in each phase. Sum over i, we obtain the total cost on this face

$$\begin{aligned} \sum_{i=1}^{k} |T_i| &= \sum_{i=1}^{k} (|R_i| + |N_i|) \\ &\leq \sum_{i=1}^{k} (\frac{|N_i|}{r} + |N_i|) \\ &= (1 + \frac{1}{r}) \sum_{i=1}^{k} |N_i| \\ &\leq (1 + \frac{1}{r})|f| \end{aligned}$$

The last equation follows from the fact $\sum_{i=1}^{k} |N_i| \leq |f|$. □

Combine Lemma 1 and Lemma 2, we obtain the following theorem.

Theorem 1 *Under the $\Omega(1)$ model, the total cost of GOAFR_{PC} is $O((c^*)^2)$ in the worst-case, where c^* is cost of an optimal route. This is asymptotically optimal.*

5 Simulations

6 Summary

References

[1] *Worst-Case Optimal and Average-Case Efficient Geometric Ad-Hoc Routing.* Unpublished.

[2] F. Kuhn, R. Wattenhofer and A. Zollinger. *Asymptotically Optimal Geometric Mobile Ad-Hoc Routing.* In Proc. of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM), Atlanta, Georgia, September 2002.

[3] E. Kranakis H. Singh, and J. Urrutia. *Compass Routing on Geometric Networks.* In Proc. of the 11th Canadian Conference on Computational Geometry, Vancouver, Canada, August 1999.