Online Routing in Quasi-Planar and Quasi-Polyhedral Graphs

Evangelos Kranakis
School of Computer Science, Carleton University

Tim Mott
Department of Mathematics, Simon Fraser University

Ladislav Stacho
Department of Mathematics, Simon Fraser University

Abstract

We address the problem of online route discovery for a class of graphs that can be embedded either in two or in three dimensional space. In two dimensions we propose the class of quasi-planar *graphs and in three dimensions the class of* quasi-polyhedral *graphs. In both cases we provide a routing algorithm that guarantees delivery. Our algorithms need only "remember" the source and destination nodes and one (respectively, two) reference nodes. Moreover, we show that the quasi-planar routing algorithm is inherently flexible in its path-finding, and as an application demonstrate computational results for a network load problem.*

1 Introduction

Ad hoc networks are widely being adopted today in many sectors of the economy in order to enhance communication capabilities. A particular case in point are sensor networks which are employed in many sectors that benefit greatly from increased surveillance (such as transportation, agriculture, personal and institutional security, radiology, medicine, and manufacturing). Given that the nodes of such a network are expected to spontaneously create an impromptu connected system that dynamically adapts to device failure and degradation, manages movement of nodes, and may even react to changes in task and network requirements, it is not surprising that a predefined topological structure is not feasible.

Since it is usually difficult to attain the required communication efficiency within complex networks, research tends to concentrate on a "simplified" topological structure of the unstructured ad hoc network. Such a structure not only must span the entire network but also maintain a sufficient number of the old links in order to sustain connectivity. The first models adapted for this purpose were planar spanners of the ad hoc network. The planarity condition (no crossing edges) was strong enough for developing the first routing algorithms for ad hoc networks.

The most efficient way to accomplish communication exchange efficiently between a given pair of nodes of an ad hoc network is by discovering a route (i.e., a path) between them. Path finding, or *routing*, is a fundamental problem in the field of ad hoc communication networks. The inherent mobility of the nodes of an ad hoc network and the lack of a pre-designed topology imply that packets must navigate the network using only local information and constant memory. Moreover, it is vital that a route discovery strategy uses only local information and is adaptable easily to the network changes. This means that at a vertex v, a routing algorithm must base its next move on v, on its "close" neighbourhood, and a small number of extra bits (typically $O(\log n)$) of stored information. Such an algorithm is said to be *local*, or *online*.

The fundamental technique for discovering routes between two nodes in an ad hoc network was the *face routing* algorithm on a planar spanner of the wireless network [13], [6]. There has been extensive literature related to discovering routes in position-

based, wireless ad hoc networks when the underlying graph is an undirected planar geometric network, e.g., see [3, 4, 6, 9, 13, 15, 16]. In such algorithms the emphasis is not only on minimizing the length of the route but also on guaranteeing packet delivery. Recent research concentrated on extending these ideas from planar networks to more complex networks. In particular, [7] addresses the problem in directed planar networks, [8] in a class of networks that have a planar backbone, while [14] provides a general survey. We also note that related to routing is traversal which is addressed in several papers: Avis et al. [1], Bose et al. [5], Chavez et al. [8], Czyczowicz et al. [10], Gold et al. [12], Peuquet et al. [19, 20]. However, traversal is less efficient than routing for message delivery.

Since nodes in an ad hoc network are typically limited by battery power, a further area of research is load-aware routing [17, 18, 11], in which the routing algorithm attempts to avoid nodes with high traffic. This results in an overall increase in network load from using longer paths, but can be instrumental in retaining network connectivity.

1.1 Results and contribution of the paper

We represent a network as a *geometric graph*, that is, a graph G with vertices V in $\mathbb{R}^2$ or $\mathbb{R}^3$, where each vertex is aware of its coordinates. Edges in G are line segments with (distinct) endpoints in V.

Our goal is to continue a step towards routing in more general networks than strictly planar graphs, cf. [2, 8]. We address the problem of online route discovery in a class of graphs that is richer than planar. In two dimensions the class of these graphs is a subclass of *quasi-planar* graphs defined in [8]. We will continue using the same name for the subclass. Intuitively speaking, such graphs are geometrically embedded into $\mathbb{R}^2$ and have underlying planar backbones with convex faces. However, within each face, arbitrary edges are allowed. In three dimensions we define a new class of graphs, *quasi-polyhedral* graphs, which extends the notion of quasi-planar graphs into $\mathbb{R}^3$. The backbones of these graphs are collections of convex polyhedra, and arbitrary edges are allowed within each polyhedron. It is important to note that for the purposes of our algorithms only the existence of a backbone is essential. The algorithms do not explicitly know which edges belong to the backbone; its existence is used only in proofs of correctness of the algorithms.

We will extend the well-known right-hand rule routing algorithm [13], [6] for planar graphs to quasi-planar. Furthermore we extend our techniques to a routing algorithm for quasi-polyhedral graphs. Our algorithm for quasi-planar graphs needs only remember the source and destination vertices and one reference vertex used to store information about the underlying face currently being traversed. Our algorithm for quasi-polyhedral graphs requires enough memory to store the source and destination vertices, and two reference vertices.

In addition to using very little memory, our quasi-planar routing algorithm is also robust: at each node, it constructs a set of candidates for its next local destination, and can use any rule or heuristic to choose from this set. This provides more flexibility than, for example, the standard Greedy algorithm, which has only one option from any node. In Section 3 we apply our algorithm as a heuristic for load-aware routing on unit disk graphs, and compare computational results with the Greedy algorithm.

Due to the space limitation, we omit proofs as well as the discussion about the quasi-polyhedral graphs. They will appear in the journal version of this paper.

2 Quasi-planar routing in $\mathbb{R}^2$

Let $G = (V, E, F)$ be a planar graph with vertex set V, edge set E, and face set F. A *convex embedding* of G is a straight-line embedding into the plane such that the boundary of every face is a convex polygon; we will associate G with its convex embedding. For the remainder of the paper we assume that such a graph G has no three collinear vertices.

Let $G = (V, E, F)$ be a convex embedding, and construct a new graph Q by adding chords to the faces of G except for the outer face f_O. That is $Q = (V, E \cup E')$, where each edge $e \in E'$ joins two vertices of some face $f \in F \setminus \{f_O\}$. We call such a graph Q a *quasi-planar graph*: there may be many crossing edges, but a facial structure remains. Figure 1 illustrates an example of a quasi-planar graph.

We refer to G as an *underlying planar graph* of Q,

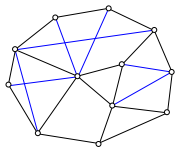

Figure 1. A quasi-planar graph and one of its underlying planar graphs.

and say that the faces $f_i \in F$ of G are *underlying faces* of Q. Note that an underlying planar graph is not necessarily unique for a given quasi-planar graph. For the purposes of our routing algorithm it is enough to know that such a graph G exists; the particular choice of G is irrelevant and will not affect the behaviour of the algorithm. In fact, the existence of the graph G is used only in proofs of correctness of the algorithm.

Define $\mathbf{cw}(u, v)$ to be the first clockwise neighbour of u starting from the direction uv. Note that uv is not required to be an edge. Similarly, $\mathbf{ccw}(u, v)$ is the first counterclockwise neighbour of u starting from the direction uv. These two functions can be computed locally, as long as $uv \in E$ or the location of v is known. Let $u, v, w_1, w_2, \ldots, w_p \in V$. Then $w_1, w_2, \ldots, w_p$ form a *clockwise sequence* around u from v if they are the first p consecutive clockwise neighbours of u starting from the direction determined by v. Note that v is not necessarily adjacent to u. A *counterclockwise sequence* is defined analogously.

We denote by uv the line segment through vertices u and v; it will be clear from context whether uv refers to an edge or a line segment. The line segment st separates the vertex set into two subsets V_A and V_B that we can think of as containing vertices "above" and "below" st, respectively. Specifically, $V_A = \{v \in V : 0 < \angle tsv < \pi\}$ and $V_B = \{v \in V : \pi < \angle tsv < 2\pi\}$, and $V = \{s, t\} \cup V_A \cup V_B$.[1] Since G is represented by a convex embedding and using the assumption that $st \notin E$, it follows that both V_A and V_B are non-empty. If a vertex v knows the geometric locations of s and t, it is a fast local computation to determine whether $v \in V_A$ or $v \in V_B$. Finally, for any vertex v of G, $N(v)$ denotes the set of neighbours of G.

[1]The definitions of V_A and V_B depend on the choice of s, t; however, their reference will be omitted as it can be easily understood from the context.

2.1 The QUASI-PLANAR algorithm

We now describe an $O(1)$-memory routing algorithm that guarantees delivery on quasi-planar graphs. The QUASI-PLANAR algorithm traverses vertices within the underlying faces intersecting st, alternately using the left- and right-hand rules (*i.e.*, using the functions **ccw** and **cw**) when $v \in V_A$ and $v \in V_B$, respectively; see Algorithm 1.

If $s = t$ or $st \in E$, then routing from s to t is obviously trivial. We may therefore assume that s and t are distinct and non-adjacent; for brevity in the following algorithm we refrain from explicitly checking for these trivial cases.

As is typical of all algorithms using the face routing technique, the QUASI-PLANAR algorithm only requires enough memory to remember s, t, and one other reference vertex x; this latter vertex is used to store information about the current underlying face. Whenever the current vertex v is in V_A, x will be in V_B, and *vice versa*.

Finally, QUASI-PLANAR requires a rule $\mathcal{R}$ that will determine the next vertex from the neighbours of the current vertex v. First suppose $v \in V_A$, and hence $x \in V_B$. Let $b_1, b_2, \ldots, b_p, a$ be a counterclockwise sequence around v from x, where $p \geq 0$, $b_i \in V_B$, and $a \in V_A$. Although the set $\{b_1, b_2, \ldots, b_p\}$ may be empty (that is, $p = 0$ is possible), we can guarantee the existence of a. We require that the function $\mathcal{R}(v, x)$ evaluate to an element from the (non-empty) set $\{b_1, b_2, \ldots, b_p, a\}$; see Figure 2.

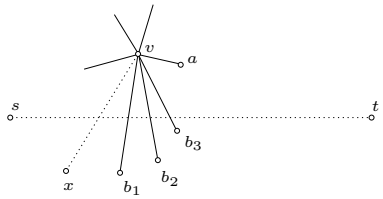

Figure 2. The current vertex is v; candidates for the next vertex are $\{b_1, \ldots, b_p, a\}$.

For sake of simplicity, we abuse notation and also refer to $\mathcal{R}(v, x)$ when $v \in V_B$ and $x \in V_A$, with the understanding that $\mathcal{R}$ is symmetric about st. That is,

$\mathcal{R}(v,x) \in \{a_1, a_2, \ldots, a_q, b\}$ where $a_1, a_2, \ldots, a_q, b$ is a clockwise sequence around v from x, $q \geq 0$, $a_i \in V_A$, and $b \in V_B$.

As we will prove shortly, the particular choice of $\mathcal{R}$ does not affect the correctness of the algorithm on quasi-planar graphs. Incidentally, observe that the algorithm can emulate standard face-routing by choosing $\mathcal{R}(v,x) = a$ for all (v,x). A more effective rule for most applications is naturally $\mathcal{R}(v,x) = argmin\{d(u,t) : u \in \{b_1, \ldots, b_p, a\}\}$, where $d(u,w)$ measures the Euclidean distance between vertices u and w. More complex rules are possible if the algorithm has access to other (non-geometric) information at each node, such as network load.

Algorithm 1 Quasi-Planar Routing

```
1: procedure QUASI-PLANAR(Q, s, t, R)
2:     v ← ccw(s, t)
3:     x ← cw(s, t)
4:     while vt ∉ E do
5:         if v ∈ V_A then
6:             Find the counterclockwise sequence b_1, b_2, ..., b_p, a around v from
                   x, where p ≥ 0, a ∈ V_A and b_i ∈ V_B, 1 ≤ i ≤ p.
7:             if R(v, x) = a then
8:                 x ← b_p
9:                 v ← a
10:            else        ▷ in this case R(v, x) = b_k for some k,   1 ≤ k ≤ p
11:                x ← v
12:                v ← b_k
13:            end if
14:        else                                                    ▷ v ∈ V_B
15:            Find the clockwise sequence a_1, a_2, ..., a_q, b around v from x,
                   where q ≥ 0, b ∈ V_B and a_i ∈ V_A, 1 ≤ i ≤ q.
16:            if R(v, x) = b then
17:                x ← a_q
18:                v ← b
19:            else        ▷ in this case R(v, x) = a_k for some k,   1 ≤ k ≤ q
20:                x ← v
21:                v ← a_k
22:            end if
23:        end if
24:    end while
25:    v ← t
26: end procedure
```

Theorem 1 *Given a quasi-planar graph Q and distinct, non-adjacent vertices $s, t \in V(Q)$, the* QUASI-PLANAR *algorithm successfully routes from s to t.*

3 Computational Results

We performed several experiments with load-aware routing on unit disk graphs in $\mathbb{R}^2$, using QUASI-PLANAR as a heuristic and compared them against GREEDY, which minimizes the Euclidean distance to t at every step.

The transmission of packets at nodes is one of the biggest energy drains [11], so we measure the number of packets $load(v)$ sent through each node v. Assume every node has the power to transmit M packets before dying, for some constant M. To preserve network integrity, we therefore attempt to minimize the maximum load $MaxLoad := max_v\{load(v) : v \in V\}$ on the network.

To this end, as a simple heuristic we use the rule $\mathcal{R}(v,x) = argmin\{load(u) : u \in \{b_1, \ldots, b_p, a\}\}$ for QUASI-PLANAR; that is, it chooses the vertex with minimum load from the set of candidates.

The test graphs each consist of 200 randomly-placed vertices in a unit square, with an edge joining two vertices if their Euclidean distance falls within a threshold τ. The parameter τ is chosen in each case to produce a graph with a desired average degree. For each test we take averages over 100 such graphs. Packets are sent between randomly-chosen pairs, and are assumed to have the same size.

Both algorithms are terminated after 25 steps; the traffic from unsuccessful messages still contributes to the total.

We present three sets of results. The first (see Table 1) shows the average success rates of the algorithms after sending 1000 packets, for several choices of average degree d. For relatively sparse graphs ($d < 25$), QUASI-PLANAR does not perform as effectively as GREEDY, since there will be large non-convex "pockets" missing in the interior of the graphs. The success rate for both algorithms is close to 100% for larger values of d.

For the second set (Table 2), we take M to be a very large number (we can assume $M = \infty$), and measure $MaxLoad$ as the number of packets increases to 1000. We fix $d = 20$ for this test.

The last test (Table 3) is the opposite of the second: we measure the number of packets sent before the first node dies, for values of M up to 1000. Again $d = 20$.

Success rate on unit disk graphs with average degree d

d	GREEDY	QUASI-PLANAR
> 25	1.000	0.990
25	1.000	0.980
20	0.999	0.970
15	0.996	0.947
10	0.932	0.805

Table 1

Maximum load after sending p packets

p	GREEDY	QUASI-PLANAR
200	17.2	17.5
400	33.0	30.2
600	48.1	42.9
800	62.6	56.9
1000	78.0	69.7

Table 2

Packets sent, in thousands, before first node death

M	GREEDY	QUASI-PLANAR
200	2.76	3.02
400	5.54	6.13
600	8.32	9.27
800	11.09	12.39
1000	13.85	15.50

Table 3

Finally note that, interestingly, QUASI-PLANAR performed very well on the random unit-disk graphs. Also note that the length of the path computed by QUASI-PLANAR is not related to the length of a shortest path joining the input vertices, but it is related to the length of faces that lie "between" the two vertices.

Acknowledgements

Many thanks to the Morelia group for numerous inspiring conversations.

References

[1] D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. In *Proc. of 7th Annu. ACM Sympos. Comput. Geom.*, pages 98–104, 1991.

[2] L. Barriere, P. Fraigniaud, and L. Narayanan. Robus position-based routing in wireless ad hoc networks with unstable transmission ranges. In *5th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM'01)*, Rome, Italy, July 2001.

[3] P. Boone, E. Chávez, E. Gleitzky, E. Kranakis, J. Opatrný, G. Salazar, and J. Urrutia. Morelia test: Improving the effi ciency of the gabriel test and face routing in ad-hoc networks. In *SIROCCO*, volume 3104. Springer, LNCS, 2004.

[4] P. Bose, A. Brodnik, S. Carlsson, E. Demaine, R. Fleischer, A. Lopez, P. Morin, and I. Munro. Online routing in convex subdivisions. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 47–59. Springer, LNCS, 2000.

[5] P. Bose and P. Morin. An improved algorithm for subdivision traversal without extra storage. *Internat. J. Comput. Geom. Appl.*, 12(4):297–308, 2002. (also in proceedings of Annual International Symposium on Algorithms and Computation (Taipei, 2000)).

[6] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7:609–616, 2001.

[7] E. Chávez, S. Dobrev, E. Kranakis, J. Opatrný, L. Stacho, and J. Urrutia. Route discovery with constant memory in oriented planar geometric networks. In *Algosensors*, volume 3121, pages 147–156. Springer, LNCS, 2004.

[8] E. Chavez, S. Dobrev, E. Kranakis, J. Opatrny, L. Stacho, and J. Urrutia. Traversal of a quasi-planar subdivision without using mark bits. accepted for 4th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN'04), Santa Fe, New Mexico, 2004.

[9] E. Chávez, S. Dobrev, E. Kranakis, J. Opatrný, L. Stacho, and J. Urrutia. Local construction of planar spanners in unit disk graphs with irregular transmission ranges, 2005, to appear.

[10] J. Czyczowicz, E. Kranakis, N. Santoro, and J. Urrutia. Traversal of geometric planar networks using a mobile agent with constant memory. in preparation.

[11] J. Gao and L. Zhang. Load balanced short path routing in wireless networks.

[12] C. Gold, U. Maydell, and J. Ramsden. Automated contour mapping using triangular element data structures and an interpolant over each irregular triangular domain. *Computer Graphic*, 11(2):170–175, 1977.

[13] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proc. of 11th Canadian Conference on Computational Geometry*, pages 51–54, August 1999.

[14] E. Kranakis and L. Stacho. Routing and traversal via location awareness in ad-hoc networks. In A. Boukerche, editor, *Algorithms and Protocols for Wireless and Mobile Networks*. CRC Press, 2005, to appear.

[15] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric ad-hoc routing: Of theory and practice. In *Proc. of the 22nd ACM Symposium on the Principles of Distributed Computing (PODC)*, July 2003.

[16] F. Kuhn, R. Wattenhofer, and A. Zollinger. Worst-case optimal and average-case effi cient geometric ad-hoc routing. In *Proc. of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, June 2003.

[17] W. H. Lee, J. M. Chang, and Y. Hasan. Dynamic memory measuring tool for C++ programs. In *Proceedings of The Third IEEE Symposium on Application-Specific Systems and Software Engineering Technology (ASSET 2000)*, Richardson, TX, 2000.

[18] J. Li, C. Blake, D. S. J. D. Couto, H. I. Lee, and R. Morris. Capacity of ad hoc wireless networks. In *Mobile Computing and Networking*, pages 61–69, 2001.

[19] D. Peuquet and D. Marble. Arc/info: an example of a contemporary geographic information system. In *Introductory Readings in Geographic Information Systems*, pages 90–99. Taylor & Francis, 1990.

[20] D. Peuquet and D. Marble. Technical description of the dime system. In *Introductory Readings in Geographic Information Systems*, pages 100–111. Taylor & Francis, 1990.