Local Geometric Routing Algorithms for Edge-augmented Planar Graphs

by

Mohammad Abdul Wahid

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
September 2013

Thesis advisor

Dr. Stephane Durocher

Author

Mohammad Abdul Wahid

Abstract

Given a geometric graph $G = (V, E)$, where V is the set of vertices and E is the set of edges and a source-target pair $\{s, t\} \subseteq V$, a local geometric routing algorithm seeks a route from s to t using only local neighborhood relationships. This thesis proposes a local geometric routing algorithm that uses only a single state bit as message overhead and guarantees delivery of messages in three different classes of edge-augmented planar graphs: convex subdivisions, quasi planar convex subdivisions (allow some augmented edges on a spanning convex subdivision) and 2-augmented triangulations (allow some augmented edges on a spanning triangulation). The proposed algorithm is origin oblivious (does not require the knowledge of the origin vertex s) and predecessor oblivious (does not require the knowledge of the predecessor vertex).

Contents

List of Figures

Acknowledgments

I express my heart-felt gratitude to Dr. Stephane Durocher for his constant supervision during my study in University of Manitoba. I am thankful to him for providing a significant amount time for my research. Due to his excellent role as my thesis supervisor, member of thesis advisory committee and thesis examining committee, let me to believe that it would not possible to complete my thesis without his care. His patience, motivation, encouragement always made me feel that I can do it and finally I did it. Thank you Dr. Steph, you will always remain in a wonderful part of my memory!

I express my gratitude to my other thesis advisory committee members Dr. Jason Morrison and Dr. James Young for their wonderful support to write a well defined and furnished thesis proposal with an interesting set of problems. I am thankful to Dr. Prosenjit Bose and Dr. James E. Young for being my thesis examining committee member. Their valuable suggestions, comments and corrections gave the final touch for making my thesis more accurate and well furnished. I am also thankful to Dr. Yang Wang (defense chair), Dr. Michael Domaratzki (associate head, graduate), and other faculty members for their support.

I would like to acknowledge the financial support from Government of Manitoba through Manitoba Graduate Scholarship (MGS), from the Faculty of Graduate Studies through University of Manitoba Graduate Fellowship and University of Manitoba Travel Award, and from the department of computer science, University of Manitoba, through Guaranteed Funding Package (GFP) for my masters program.

I will never forget the support and encouragement that I got from my fellow lab mates, Debajyoti Mondal, Saeed Meherabi, Dr. Matthew Scala and Dr. Bob Fraser.

I would like to thank all of my friends in Winnipeg for their encouragement and support during my thesis.

Finally, I would like to mention that my parents are my ultimate source of inspiration.

I would like to dedicate my thesis to my handsome father and my lovely mother.

Chapter 1

Introduction

The local routing problem is fundamental in many different disciplines, including wireless ad-hoc networks, robotics, motion planning, geographical information systems (GIS), fault-prone meshes and so on [27]. Wireless ad-hoc networks have a wide range of applications, including monitoring environmental changes in mountains, forests, oceans and so on, monitoring and controlling traffic system from remote locations, providing security in shopping malls, car parking, etc. [30]. Wireless ad-hoc networks do not have any fixed infrastructure and consist of a set of battery driven host nodes, where each node has very limited resources. The medium of communication is radio transmission and two nodes in the wireless ad-hoc network can communicate directly if each node lies within the transmission range of the other node. To communicate between two nodes that do not share a direct link requires a routing algorithm that delivers each message from its source node to its destination node. An analogous problem arises in some fields of robotics, where a robot's environment is modeled as a graph, and the robot path-finding problem is modeled by

the problem of finding a route from a source node to a target node in that graph [27].

Nowadays, finding the relative location of a wireless node is easy, because GPS receivers and related positioning devices have become inexpensive, small in size, and require little power [17]. This geometric information can be very useful for routing algorithms. A geometric routing algorithm is a routing algorithm that uses geometric information, such as coordinates, distances, angles, etc.

A wireless ad-hoc network can be modeled as a geometric graph. A geometric graph is a graph $G = (V, E)$ where the vertex set V is a set of n points in a d-dimensional Euclidean space, $\mathbb{R}^d$ (where typically $d \in \{1, 2, 3\}$). Two vertices share an edge $(u, v) \in E$, if the two corresponding nodes in the wireless ad-hoc network lie within each other's transmission ranges.

One objective of routing algorithms is to deliver a message from the source node to the destination node using as little information about the network structure as possible. A routing algorithm in a geometric graph, G, is *local* if the corresponding algorithm uses only local information to find the route. More specifically, a local geometric algorithm knows only the coordinates of the current vertex and those of its neighboring vertices. Using only this information in addition to the identity (e.g., coordinates) of the source and target nodes, a local routing algorithm must select one of its neighbors to which to forward the message. The process then repeats at the next node along the route path until the message reaches its destination.

Given a geometric graph G and two vertices s and t, the problem is to find a local geometric routing algorithms that provides a route from s to t in G.

1.1 Motivation

In traditional centralized computer networks, there are three different types of nodes: host nodes, server nodes and router nodes. Router nodes are responsible for routing a message from one node to another node. An application in the host node communicates with a server node by sending a number of messages. Along with the address of the source node and the destination node, each message is enclosed in a packet. Routers have complete knowledge of the communication network stored in a persistent routing table. Whenever, a packet comes to a router, it reads the destination address and forwards the message to appropriate channel based on the entry in the routing table.

On the other hand, a wireless ad-hoc network does not distinguish between hosts, servers, and routers. The same node in the network acts as a host, as a server, as well as a router. No node has the knowledge of the entire network structure. A node only knows about its neighboring nodes. Therefore, it is necessary to design a routing algorithm that uses only local information.

In parallel computing, resources are kept as a $n \times n$ mesh. Two resources are interconnected by a communication medium if they are vertically or horizontally adjacent to each other. A mesh can be modeled as a geometric graph $G = (V, E)$ where resources are represented by a set of nodes V and interconnections between resources are represented by a set of edges E. There is an edge $(u, v) \in E$ between two nodes $\{u, v\} \in V$ if they are adjacent in the mesh (i.e., the geometric distance between u and v is 1). If one resource wants to send a message to another resource, then a simple routing algorithm is sufficient (e.g., greedy routing [13]).

But, what happens if some nodes or communication links fail or appear after a certain period of time? Only the adjacent neighbors are aware of these sudden failures. Therefore, it is necessary to design a local routing algorithm that can route a message based on the local neighborhood relationship in the corresponding graph.

1.2 Challenges

Local routing algorithms, also known as online routing algorithms, are different in nature compared to traditional routing algorithms.

A node in a wireless ad-hoc network has very limited power and memory. Unlike traditional routing algorithms, a local routing algorithm cannot use any persistent routing database or dedicated centralized routing device. No node has the knowledge of the topology of entire network; instead, each node only knows its neighbors. A wireless ad-hoc network does not have any fixed structure, and the network topology can change unpredictably. Therefore, the topology of a wireless ad-hoc network is fully distributed. A routing algorithm is implemented in each of the wireless nodes. So, it should be simple, easy to implement, and efficient in routing and computation.

In some distributed routing algorithms, it is allowed to use persistent memory. One objective of these distributed algorithms is minimizing the size of persistent memory. Some algorithms use mark bits to distinguish the visited and unvisited nodes. In contrast, a local routing algorithm does not use any persistent memory or mark bits. It is only allowed to use some additional bits of memory inside the message, known as message overhead, to remember the state information. However, the size of the message overhead is directly proportional to the communication delay. That is why

it needs to be as small as possible. Failure to store sufficient state information as message overhead may causes the message to become stuck in a loop. In order for the algorithm to recognize a loop, it must know whether a node was already visited in any previous step. Remembering the identity of k nodes (e.g., coordinates, labels, etc.) in a graph with n nodes requires $O(k \log n)$ bits of memory as message overhead, which is very costly in some networks like wireless ad-hoc networks. Therefore, finding a local routing algorithm for wireless ad-hoc networks that guarantees delivery as well as minimizing the message overhead is a challenging task.

1.3 Problem Definition

Many different classes of geometric graphs are commonly considered in local routing problems in wireless ad-hoc networks.

Unit disc graphs are one of the common geometric models for ad-hoc wireless networks. A unit disc graph U is not necessarily planar. However, if U is connected, then using only local information, it is possible to extract a planar spanning subgraph $U' = (V, E')$ where $E' \subseteq E$ [5]. Therefore, any local routing algorithm works in unit disc graph if the same algorithm works in any planar graph. However, many other different types of geometric graphs can also be used to model a wireless ad-hoc network where it may not be possible to extract a planar subgraph locally. Therefore, it is necessary to provide a local routing algorithm that works in non-planar graphs. Moving beyond planar graphs, it is natural to consider finding a routing algorithm in edge-augmented planar graphs. An edge-augmented planar graph is a planar graph where a set of edges are added to an underlying planar graph.

The main contribution of this thesis is a better understanding of local routing in convex subdivisions and edge-augmented planar graphs. In particular, this thesis thoroughly studies three different classes of graphs: convex subdivisions, k-quasi planar convex subdivisions and 2-augmented triangulations.

Let $G = (V, E, F)$ be a planar geometric graph, where V is the set of vertices, E is the set of edges and F is the set of faces. G is a *convex subdivision* if every internal face and the boundary of the outer face are convex polygons. G is a *quasi planar convex subdivision* if each face in F may contain a set of augmented edges, each connecting a pair of vertices in the same face. These edges can cross each other but boundary edges of any convex face are non-crossing. G is a k-quasi planar convex subdivision if G is a quasi planar convex subdivision, and every augmented edges in G has at most k crossings. G is a triangulation if each face in F is a triangle (except possibly the outer face). G is an *augmented triangulation* if G is a triangulation and there are some augmented edges between some pairs of vertices and these augmented edges can cross some edges in the underlying triangulation. G is a k-augmented triangulation if G is an augmented triangulation with at most k crossings per edge.

Throughout the thesis, we assume that all points representing vertices in G are in general position, i.e., no three points are collinear. We also assume that there must be a path from $s \in V$ to $t \in V$ (i.e. the graph is connected).

This thesis addresses the following important questions:

1. Is there any constant-memory origin-oblivious predecessor-oblivious routing algorithm that guarantees delivery in any convex subdivision?

2. Is there any constant-memory origin-oblivious predecessor-oblivious routing al-

gorithm that guarantees delivery in any k-quasi planar convex subdivision for any k?

3. Is there any constant-memory origin-oblivious predecessor-oblivious routing algorithm that guarantees delivery in 2-augmented triangulations?

1.4 Outline of the Results

The main contribution of this thesis is finding an algorithm that answers all three questions asked in the previous section.

Routing in convex subdivisions We show that if G is a convex subdivision, then our algorithm guarantees delivery. We divide the path traversed by the message into a set of clockwise and counterclockwise chains. The current chain is always enclosed by the bounded region of the last chain. Therefore, our algorithm progresses if the message traverses all the chains one-by-one (see Chapter 4).

Routing in k-quasi planar graphs We show that the same algorithm proposed for convex subdivisions guarantees delivery in k-quasi planar graphs for any k. Since there is an underlying convex subdivision, we prove that no two consecutive chains cross each other along augmented edges, except in a specific case which we show does not affect the outcome. We prove that our algorithm eventually reaches the destination (see Chapter 5).

Routing in 2-augmented triangulations A slight modification to our algorithm guarantees delivery in 2-augmented triangulations. Due to the existence of an

underlying triangulation, we show that no two consecutive chains cross each other, which implies a forward progress (see Chapter 6).

A deterministic routing algorithm is *oblivious* if it does not have any message overhead and the routing decision is made based only on the message containing vertex u, all neighbors of u and the target vertex t. It is already known that no oblivious local routing algorithm exists that succeeds in all convex subdivisions [6].

Every convex subdivision is a quasi planar convex subdivision. Consequently, no oblivious local routing algorithm exists that succeeds in all quasi planar convex subdivisions. Our algorithm is origin oblivious, predecessor oblivious, requires only 1 bit of memory, and guarantees delivery of the message in both convex subdivisions and k-quasi planar convex subdivisions for any k. Therefore, our algorithm is tight in terms of memory. Greedy-compass routing is oblivious, and guarantees delivery in any triangulation [4] and 1-augmented triangulations (see Chapter 6). However, it fails in some 2-augmented triangulations. The proposed algorithm in this thesis guarantees delivery in any 2-augmented triangulation but requires 1 bit of memory.

1.5 Thesis Organization

1.5.1 Chapter 1

Chapter 1 gives the introductory idea of local routing problems and their applications. It also discusses why local routing problems are different and challenging compared to traditional routing algorithms. This chapter focuses on three interesting research questions and their proposed solutions.

1.5.2 Chapter 2

Chapter 2 discusses relevant background related to geometric local routing problems. Firstly, this chapter defines geometric local routing problems. After that, it defines many important and common terms that are used in local routing, such as guaranteed delivery, k-locality, memoryless algorithms, k-bit memory algorithms, origin-oblivious algorithms, predecessor-oblivious algorithms, deterministic algorithms, randomized algorithms and so on. This chapter also defines various classes of graphs that can be used to model a wireless ad-hoc network.

1.5.3 Chapter 3

Chapter 3 gives the state of art of local routing algorithms. Section 3.1 discusses different local routing algorithms proposed for different classes of undirected graphs such as triangulations, convex subdivisions, planar graphs and quasi planar graphs. Section 3.2 discusses some important results on local routing algorithms that are proposed for directed graphs.

1.5.4 Chapter 4

Chapter 4 proposes a new, simple and powerful geometric local routing algorithm. The algorithm is origin oblivious, predecessor oblivious and requires only 1 bit of memory and guarantees delivery in convex subdivisions. Section 4.1 and Section 4.2 gives the outline and pseudocode, respectively, of the algorithm. The last section proves that the proposed algorithm guarantees delivery in convex subdivisions.

1.5.5 Chapter 5

Chapter 5 presents arguments showing that the proposed algorithm in Chapter 4 also guarantees delivery in any k-quasi planar convex subdivision for any k.

1.5.6 Chapter 6

Chapter 6 shows that both greedy-compass routing and the proposed routing algorithm in Chapter 4 guarantee delivery in 1-augmented triangulations, but fail to do so in some 2-augmented triangulations. In this chapter we also show that a small change in our proposed algorithm provides guaranteed delivery in any 2-augmented triangulation.

1.5.7 Chapter 7

Chapter 7 concludes the thesis by reviewing the main contribution of this thesis. Section 7.1 compares our proposed routing algorithm with other existing routing algorithms and Section 7.2 gives possible directions for future research on local routing problems in edge-augmented planar graphs.

Chapter 2

Preliminary Discussion

The main objective of this chapter is to provide necessary background to local geometric routing problems. Section 2.1 defines local routing problems and some relevant terms used in local routing algorithms. Section 2.2 discusses different categories of local routing algorithms. Section 2.3 discusses some techniques for evaluating routing functions and Section 2.4 defines various classes of geometric graphs that are useful to model wireless ad-hoc networks as well as to understand local routing algorithms in depth.

2.1 Routing Problem

Let $G = (V, E)$ be a geometric graph and let $\{s, t\} \subseteq V$ be a pair of vertices. A local routing algorithm finds a route from s to t in G using only local information. We now define locality in a geometric graph.

Two vertices $\{u, v\} \subseteq V$ are neighbors in G if there is an edge $(u, v) \in E$. The

neighborhood $N(u)$ (also called 1-hop neighborhood) of a vertex $u \in V$ is the set of all neighbors of u in G. The k-hop neighborhood of the vertex u is the set of nodes that can be reached from u through at most k number of edges. An algorithm A is *k-local* if it has knowledge of the k-hop neighborhood of the node u that contains the message. A is *1-local* (or only *local*) if $k = 1$, i.e., A knows all its 1-hop neighbors.

Let $cw(u, v)$ and $ccw(u, v)$ denote the clockwise and counterclockwise first neighbors of u, respectively, starting from the direction of the vector $\overrightarrow{uv}$.

A routing algorithm can be thought of as a *routing function*, $f(s, t, u, v, G_k(u), M)$ where $s \in V$ is the source vertex, $t \in V$ is the target vertex, $u \in V$ is the message containing node (the current node), $v \in V$ is the predecessor of u, i.e., u received the message from v in the previous step of the algorithm, $G_k(u)$ is the subgraph containing all k-hop neighbors rooted at u, and M is the message overhead [7]. A routing function decides to which neighboring vertex the message should be forwarded. Therefore, a sequence of calls to the routing function gives the sequence of forwarding vertices that determine the route.

A routing algorithm is implemented at every node in the wireless ad-hoc network. Every neighbor of a node corresponds to a distinct pair of ports in the node: one incoming port and one outgoing port. A message is sent to one of its neighbor via the corresponding outgoing port and received from one of its neighbor via the incoming port. Some network infrastructures are able to identify the incoming port from which they receive the message. This type of infrastructure facilitates the routing algorithm by providing the identity (e.g., incoming port number, coordinates, label, etc.) of the predecessor vertex. If a routing algorithm A has knowledge of the

identity of the predecessor vertex, then A is called *predecessor aware*; otherwise A is called *predecessor oblivious*. Many local routing algorithms use the predecessor vertex (or the corresponding incident edge) as a reference to keep the track of a part of subgraph already traversed (e.g., face traversal in face routing [5]). Therefore, finding a predecessor-oblivious routing algorithm is more challenging compared to a predecessor-aware routing algorithm.

Any routing algorithm A must know the identity of the target vertex, t, which is stored in the message header. In some settings, A also knows the identity of the origin vertex s. Such an algorithm is called *origin aware*; otherwise it is called *origin oblivious*. Some routing algorithms use the st line segment as reference and traverse all the faces that intersect the st line segment (see Chapter 3). Since origin-oblivious routing algorithms do not know the coordinates of s, they cannot use any such reference line segment. This makes it more challenging to design origin-oblivious routing algorithms.

2.2 Categories of Routing Algorithm

A routing algorithm can either be *deterministic* or *randomized*. *Deterministic* routing algorithms choose a neighbor of the current node deterministically to forward the message. Therefore, a deterministic algorithm always generates the same path each time a message is passed from s to t on the same graph. On the other hand, *randomized* algorithms send a message from the current vertex u to a neighboring vertex $v \in N(u)$ based on a random distribution on $N(u)$. A routing algorithm is *k-bit randomized* if the next vertex visited from a vertex u is a function of u, $N(u)$

and k random bits. A randomized algorithm *fails* if the probability that the message reaches t from s is 0 and *succeeds* otherwise. However, in some cases the number of steps the algorithm takes to succeed approaches infinity.

Routing problems can be categorized into three different types: i). If the message is sent to a single target node t then the problem is called a *uni-casting* problem. ii). If the message is sent to a set of nodes in a particular region then the problem is called a *geo-casting* problem. iii). When the message is sent to all nodes in the network then the problem is known as a *broadcasting* problem. Note that only *uni-casting* routing will be considered in this thesis.

2.3 Performance Evaluation

A local routing algorithm, A, *succeeds* in a class of graphs if A can deliver the message from any source vertex to any target vertex in all the graph instances in that class. On the other hand, A is *defeated* by a graph if A gets stuck in a loop and cannot send the message to some target node. *A fails* in a class of graphs if A is defeated by at least one instance of the class of graphs. In some scenarios, it is required to design a deterministic routing algorithm that guarantees the delivery of the message within a finite number of steps in every instance of a particular class of graphs. The corresponding routing algorithms are called *guaranteed delivery* routing algorithms.

Since the nodes that participate in local geometric routing are not aware of the global infrastructure of the network, it becomes a challenge to establish a successful route to the destination ensuring that the message does not get stuck in a loop. To

overcome this challenge, network nodes require the message to contain additional information other than the actual data being delivered. This additional information or message overhead, denoted by $M(n)$, helps a node to forward the message correctly to its destination. A local geometric routing algorithm, A, may need to remember the coordinates of a number c of reference vertices to guarantee the delivery. Moreover, in some cases A may be required to use some bits of memory to encode state information. Storing each coordinate requires $\Omega(\log n)$ bits of memory. Therefore, if c is constant then $M(n) \in \Omega(\log n)$. If $M(n) = 0$ (i.e., A does not require any additional memory), then the routing algorithm A is called *memoryless* and if $M(n) = k$, where k is a constant (i.e., A requires k bits of additional memory), then A is called a *constant-memory* routing algorithm. In general, a *k-bit memory algorithm* is a routing algorithm that requires k bits of memory as message overhead. Note that as the message overhead increases, the length of the message which is directly proportional to the communication delay also increases. If multiple such communications occur simultaneously, the network traffic increases and the network is more likely to become congested.

Let a routing algorithm A find a path $\lambda = v_0, v_1, v_2, \ldots, v_x$ in G, where $v_0 = s$ is the source vertex and $v_x = t$ is the target vertex, for some $x \geq 1$. If the computation cost by A in a node v_i is c_i, then the running time of A is $\sum_{i=0}^{x} c_i$. If the computation cost in all nodes in the path is k where k is a constant, then the running time of the routing algorithm is a function of the path length $|\lambda|$. Minimizing the running time of a routing algorithm A is equivalent to minimizing the path length generated by A.

A local routing algorithm A is c-competitive if for every graph G and every source-

target pair (s, t), the route specified by A from s to t in G has length at most c times the length of the shortest path from s to t in G. This thesis will not consider optimizing the competitive ratio; rather it will only ensure *guaranteed delivery*.

2.4 Geometric Graphs

Let $G = (V, E)$ be a graph, where V is a set of vertices and E is the set of edges. The cardinality of the vertex set and edge set are denoted by $|V|$ and $|E|$ respectively.

G is a *directed graph* if any edge in E is an ordered pair of vertices, called a *directed* edge. A directed edge $e = (u, v)$ is considered to be directed from u to v. On the other hand, an undirected graph is one in which there is no orientation in edges. The edge (u, v) is identical to the edge (v, u). Unless otherwise specified, any graph G is assumed to be an undirected graph.

In an undirected graph G, two vertices u and v are called *connected* if there is a path from u to v in G. If for all pair of vertices in G are connected, then G is called a *connected* graph. Every graph studied in this thesis is considered to be connected (since this is necessary to ensure the existence of a path from s to t).

A graph G is *planar* if G can be drawn in a geometric plane such that no two edges cross each other in the plane; they may only have a common end vertex. A particular such drawing in a geometric plane is called a *plane graph*. A plane graph subdivides the plane into regions called *faces*. Plane graphs are also known as planar subdivisions. All the faces in a planar subdivision can be categorized into two types: inner faces and outer faces. If a face f bounds a region in the plane, then f is an inner face. The whole unbounded region, outside all the inner faces, is called the

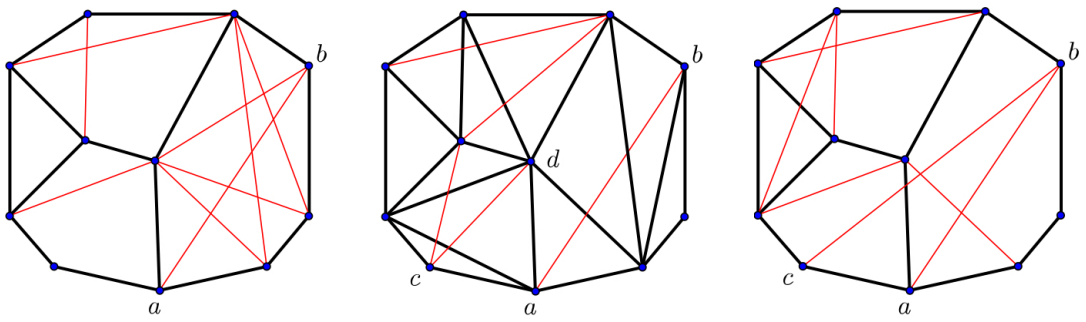
outer face.

A planar subdivision is said to be a *convex subdivision* if all of its faces and the boundary of the outer face are convex polygons. A planar subdivision is said to be a *triangulation* if all of its faces (except possibly the outer face) are triangles.

A unit disc graph $U = (V, E)$ is a geometric graph in which an edge $(u, v) \in E$ exists if and only if the geometric distance between u and v is at most one.

Given points u and v in the plane, let $disc(u, v)$ be the disc with diameter (u, v). The *Gabriel graph* $GG = (V, E)$ is a geometric graph in which the edge (u, v) is present if and only if $disc(u, v)$ contains no other vertex of V. In other words, (u, v) is an edge in GG if and only $\angle urv$ is acute for every $r \in V, r \neq u, r \neq v$ [26]. The Gabriel graph is very useful in modeling wireless ad-hoc networks.

Let P be a set of points in a geometric hyperplane. The Voronoi diagram of P is a partitioning of space into cells such that all points within a Voronoi cell are closer to the same element $p \in P$ than any other point in P. The Delaunay graph (DG) is the straight-line face dual of the Voronoi diagram [12]. Two points in P have an edge between them in the Delaunay graph, if their Voronoi cells have an edge in common. Every face in the DG is maximal (i.e., triangle), therefore, DG is also a triangulation, called a Delaunay triangulation $DT(P)$. In a Delaunay triangulation $DT(P)$, no point in P is inside the circumcircle of any triangle in $DT(P)$. $DT(P)$ maximizes the minimum angle of all the angles of the triangles in the triangulation. The Delaunay graph is a super-graph of the Gabriel graph, i.e., $GG \subseteq DG$. Note, unit disc graphs, Gabriel graphs and Delaunay triangulations are discussed with respect to related results (Chapter 3) but are not directly referenced in the results presented



(a) 4-quasi planar convex subdivision.

(b) 2-augmented triangulation.

(c) 2-augmented convex subdivision.

Figure 2.1: Examples of some edge-augmented planar graphs where red edges represent augmented edges.

in this thesis.

2.5 Edge-augmented Planar Graphs

2.5.1 Quasi Planar Convex Subdivision

Let $G = (V, E, F)$ be a convex subdivision, where F is the set of faces in G. Construct a new graph Q by adding any number of augmented edges to the faces (except the outer faces, $f_0 \in F$) of G. More formally, graph $Q = (V, E \bigcup E')$, where $(u, v) \in E'$ is an augmented edge that joins u and v in a same face of G, $f \in F \setminus \{f_0\}$. Two augmented edges can cross each other if they are in same face. No edge $e \in E$ is crossed by any augmented edge $e' \in E'$. Q is called a *quasi planar convex subdivision*, where edges in E are called *convex subdivision edges* and edges in E' are called *augmented edges*.

Quasi planar convex subdivisions are also known as *quasi planar graphs*. Q is a *k-quasi planar convex subdivision* if convex subdivision edges have no crossing, whereas augmented edges have at most k crossings, each of which is with some other augmented edges. As an example, Figure 2.1a shows a 4-quasi planar convex subdivision, where black edges are convex subdivision edges and red edges are augmented edges. Since the augmented edge $(a, b) \in E'$ crosses four other augmented edges and no other edges crosses more than four, the illustrated graph is a 4-quasi planar convex subdivision.

2.5.2 k-Augmented Triangulation

A *k-augmented triangulation* is a geometric graph $G = (V, E \bigcup E')$ that contains a set of augmented edges E' and a spanning triangulation $T = (V, E)$, where E is the set of edges in the underlying triangulation (also called *triangulation edges*). All edges in G have at most k crossings. Note that unlike quasi planar convex subdivisions, a triangulation edge in G can also have at most k crossings. Figure 2.1b gives an example of a 2-augmented triangulation, where augmented edges are marked in red and triangulation edges are marked in black.

2.5.3 k-Augmented Convex Subdivision

A *k-augmented convex subdivision* is a geometric graph $Q = (V, E \bigcup E')$ that contains a spanning convex subdivision $G = (V, E, F)$, where F is the set of faces in G, E is the set of convex subdivision edges and E' is the set of augmented edges. All edges in Q have at most k crossings. The difference between a k-quasi planar

convex subdivisions and a k-augmented convex divisions is that in the former class of graphs, no convex subdivision edge can have crossing, whereas in the later class of graphs, this constraint does not hold. Figure 2.1c gives an example of a 2-augmented convex subdivision, where an augmented edge bc crosses one convex subdivision edge and one other augmented edge.

It is important to mention that the partition of edges in any edge-augmented planar graph is not explicit. In fact, it is not a unique partition. Specifically, edges visible to the routing algorithm are only identified by their endpoints, no additional information is provided.

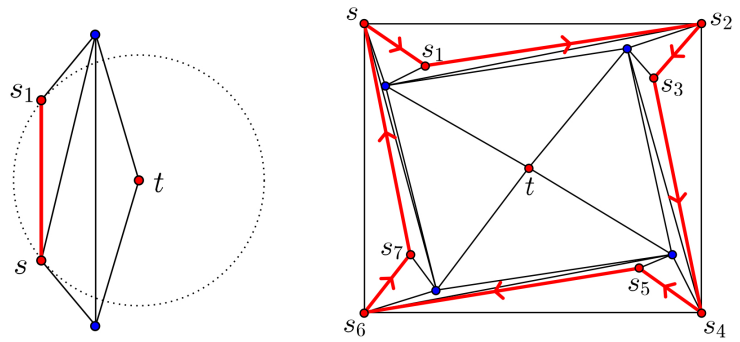
Chapter 3

Related Work

This chapter examines previous results related to local routing algorithms. The chapter is divided into two sections. Section 3.1 discusses local routing algorithms that are designed only for undirected graphs. The section is further subdivided into subsections to discuss local routing algorithms for different classes of graphs, such as triangulations, convex subdivisions, planar graphs and quasi planar convex subdivisions. Section 3.2 discusses the routing problem for directed graphs.

3.1 Undirected graph

Unit disc graphs are widely used to model wireless ad-hoc networks. A unit disc graph, U, is not necessarily planar. However, if U is connected, then using only local information, it is possible to extract a planar spanning subgraph $U' = (V, E')$, where $E' \subseteq E$ [5]. Consequently, any local routing algorithm that works in planar graphs, also works in unit disc graphs. Therefore, much study has been done on planar



(a) A triangulation that defeats the greedy routing algorithm.

(b) A triangulation that defeats the compass I routing algorithm [4].

Figure 3.1: Greedy routing and compass I routing fail in some triangulations.

graphs and many different sub-classes of planar graphs. This section discusses many important and related local routing algorithms for triangulations, convex subdivisions, planar graphs, and quasi planar convex subdivisions (quasi-planar graphs).

3.1.1 Triangulations

Greedy routing was one of the first geometric local routing algorithm proposed in the 1980s [13]. It uses location information of its neighboring nodes and chooses the best candidate node greedily to which to forward the message. One such algorithm is provided by Lin et al. [25], where coordinates or distances between the current node and the destination node are known. The algorithm sends a packet to the neighbor that is closest to the destination. Greedy routing algorithms are simple and scalable. They succeed in limited classes of graphs such as Delaunay graph. However, in general they fail in some triangulations (Figure 3.1a) [20].

A variation of greedy routing, called *midpoint routing*, was proposed by Si and Zomaya [29]. The main idea of midpoint routing is to find a neighbor of the current node which minimizes the Euclidean distance to the mid-point of the line segment joining current node and the destination node. However, similar to greedy routing, midpoint routing guarantees delivery in Delaunay triangulations, but not in all triangulations.

Kranakis et al. [20] proposed a routing algorithm that uses the direction of the target node, t, from the current node, u. The algorithm is known as *compass I* routing. The term *compass neighbors* refers to two special neighbors: $cw(u,t)$ and $ccw(u,t)$. The compass I routing algorithm sends the message to the compass neighbor, v, that forms the smallest angle with the line segment (u,t). This algorithm is also simple, memoryless but messages may get stuck in a loop, and therefore, message delivery is not guaranteed (See Figure 3.1b).

Recently, in 2013, Chun et al. [11] gave a unified view of existing greedy routing algorithms. They present a general form for a greedy routing algorithm using a generalized objective function $g(v,u,t)$, where u is the current node, $v \in N(u)$ and t is the target node. The algorithm repeatedly forwards the message from the current node u to a neighbor v that minimizes the value $g(v,u,t)$. They defined a class of objective functions called congruent-invariant objective functions that satisfy $g(v,u,t) = g(v',u',t')$ whenever two triangles $\triangle vut$ and $\triangle v'u't'$ are congruent (Recall that two triangles are congruent if and only if one triangle can be obtained from the other triangle by a combination of translations, rotations and reflections). They showed that the objective functions used in greedy routing, midpoint routing and

compass routing belong to the class of congruent-invariant objective functions. They also proposed three other algorithms based on some other congruent-invariant objective functions. All routing algorithms with congruent-invariant objective functions guarantee delivery in Delaunay triangulations.

Bose et al. [4] studied various online routing algorithms in triangulation graphs. Both greedy routing and compass I routing fail in some triangulations. However, the combination of greedy routing and compass routing, also known as greedy-compass routing, guarantees delivery in any triangulation. The outline of this algorithm is as follows: Let u be the current vertex, t be the target vertex, $v_1 = ccw(u,t)$ and $v_2 = cw(u,t)$. Greedy-compass routing forwards the message to whichever of the vertices v_1 and v_2 is closest to the target t. Greedy-compass is memoryless, origin oblivious and predecessor oblivious.

3.1.2 Convex Subdivisions

Although there is an oblivious routing algorithm (greedy-compass routing) that guarantees delivery in any triangulation, one might expect that there exist such an algorithm for convex subdivision as well. However, Bose et al. [6] proved that there is no oblivious routing algorithm possible for convex subdivisions. They constructed a class of convex subdivisions such that any oblivious routing algorithm fails in at least one of the graphs.

Kranakis et al. [20] proposed another geometric local routing algorithm, called *compass II*, that uses a line segment from source to destination, and then traverses the sequence of faces that intersect the line segment and updates the source node.

Due to its face traversal mechanism, the algorithm is also called *face routing*. Since, the algorithm needs to use this particular line segment, it is required to store the source vertex dynamically in the message. Furthermore, the source endpoint of the segment is updated dynamically throughout the algorithm, requiring its coordinates to be passed with the message header. Therefore, the algorithm is not oblivious. Bose et al. [5] give a memoryless origin-oblivious predecessor-aware face routing algorithm that provides guaranteed delivery in any static convex subdivision.

Morin [27] proposed a 1-bit memory randomized routing algorithm, called *random-compass*, for convex subdivisions. The algorithm achieves that the probability of successfully sending the message to the target is very high. Random-compass forwards the message to one of the compass neighbors of the current vertex u, based on tossing a random coin. If the outcome is "heads", then algorithm sends the message to $ccw(u, t)$, otherwise it sends the message to $cw(u, t)$.

Subsequently, Chen et al. [10] showed that, for any memoryless randomized algorithm A, there exists a convex subdivision $G = (V, E)$, where $n = |V|$, having two vertices s and t such that the expected number of steps taken by A is $\Omega(n^2)$. A random walk on G routes a message with the same expected path-length [28]. In addition to these, they show that there exists a triangulation such that the expected number of steps taken by random-compass routing is $2^{\Omega(n)}$.

There is no known deterministic constant-memory, origin-oblivious predecessor-oblivious routing algorithm for convex subdivisions. One well-known origin-aware predecessor-aware routing algorithm for convex subdivision is the right-hand rule algorithm. The term "right-hand rule" originated from the technique of traversing

a maze where there is a single entrance and single exit, both are on the maze's perimeter. If someone starts traversing the maze by touching the wall using the right hand without lifting the hand, then after finite number of steps he or she will reach the exit. The same concept can be applied as a local routing algorithm for any convex subdivision [27]. We can consider a line from the source vertex, s to the target vertex, t. Avoid all the edges that intersect the line (s, t) and forward the message using right-hand rule. After a number of steps the message will eventually reach to t because virtually the algorithm is deleting all the intersecting edges. Note that same algorithm can also be designed using the *left-hand rule*. Since, the algorithm needs to store s and the predecessor vertex inside the message, it is origin-aware and predecessor-aware.

3.1.3 Planar Graphs

Face routing also guarantees delivery in any planar graph [5]. However, in some planar graph (e.g., a snake like path from s to t), face routing traverses $\Omega(n^2)$ (here n is the number vertices in G) edges of G.

de Berg et al. [2] showed that any edge in the convex subdivision has a unique rank with respect to a fixed remote point and that rank can be computed locally. The lowest-ranked edge in a face is called the *entry edge* of that face. If any face traversal is started from its entry edge, then it is possible to represent the subdivision as a tree, where each face is represented by a node. Two nodes are adjacent in the tree if and only if the corresponding faces share an entry edge. Therefore, there is a predecessor-aware memoryless routing algorithm for any convex subdivision. Bose et

al. [3] use the same traversal techniques of de Berg et al. [2] to define an constant-memory predecessor-aware local routing algorithm for any planar graph. It is also shown that no deterministic memoryless algorithm guarantees delivery in 2-connected graphs [4].

In another work, Braverman [8] shows that there is an $O(\log n)$-bit memory routing algorithm that guarantees delivery in any undirected graph (regardless of geometry). He proved that any undirected graph can be represented by a universal exploration sequence and it is sufficient to use $O(\log n)$ bits of memory to store the index into the traversal sequence. However, no good lower bound is known on the required memory for routing algorithms that guarantees delivery in many common classes of geometric graphs, e.g., planar graphs.

3.1.4 Quasi Planar Convex Subdivisions

A wireless ad-hoc network can also be modeled by many types of graphs other than unit disc graph. Therefore, researchers are interested in finding routing algorithms for broader classes of graphs, for example, quasi planar convex subdivisions. Urrutia et al. [31] give a traversal technique in a quasi planar convex subdivision (or quasi-planar graph) that does not need any mark bits. Using a mark bit means that the algorithm can mark a vertex (e.g., to denote whether a vertex has already been visited) using some extra memory. They achieve a running time of $O(|E| \log |E|)$ for the traversal algorithm.

Kranakis et al. [21] provide a modified version of the right-hand rule for quasi planar convex subdivisions and quasi-polyhedral graphs. The modified right-hand

rule alternatively uses the right-hand and the left-hand rules respectively.

They proved that their algorithm guarantees delivery in k-quasi planar convex subdivisions for any k. Since, the algorithm remembers the st line segment and a reference vertex, it requires $\Omega(\log n)$ bits of memory.

3.1.5 Other Relevant Work

Although this thesis focuses on guaranteed delivery, many other local routing algorithms have also been proposed to improve the competitive ratio. In some perspective, local routing algorithms can be classified into three different groups: restricted directional flooding, greedy routing and face routing algorithms. Flooding algorithms are mainly used with some types of controlled packet duplication techniques to ensure that every destination receives at least one copy of the message. Due to extensive consumption of bandwidth, battery and computational resources, flooding algorithms are typically avoided. *Restricted directional flooding* algorithms compute an expected zone for the destination node and flooding is done towards the expected zone [1, 19].

Some algorithms combine greedy routing and face routing techniques. In the *Greedy Parameter Stateless Routing (GPSR)* algorithm, if the current node is the closest node to the destination node compared to its neighbors, then it applies the face routing algorithm of Karp et al. [18], otherwise, it applies the greedy routing algorithm. It provides guaranteed delivery on static connected planar graphs, but fails to guarantee delivery for graphs with non-convex faces.

Another such hybrid algorithm is GOAFR+ due to Kuhn et al. [22]. GOAFR+ first defines an area and then uses face routing on that area and updates the previously

defined area. GOAFR+ achieves a route with length at most a constant factor times that of the shortest path. However, GOAFR+ only guarantees delivery in convex subdivisions.

A quasi unit disc graph contains all edges shorter than a given fixed parameter d between 0 and 1 and no edges longer than 1. Kuhn et al. [23] showed that any algorithm that does not use routing tables requires sending of $\Omega((\frac{c}{d})^2)$ messages to guarantee delivery in quasi unit disc graphs, where c is the length of the shortest path between the source vertex and the target vertex. They gave a routing algorithm that combines both greedy routing and flooding technique, and sends asymptotically optimal number of messages to guarantee delivery in quasi unit disc graphs.

A quasi unit disc graph is not necessarily planar, but it can be viewed as a virtual planar graph by adding a virtual node at each point where two or more edges cross as well as splitting the edges at these virtual nodes [23]. Kuhn et al. [23] and Lillis et al. [24] maintain routing tables in real nodes to keep the track of virtual nodes. However, Guan [16] showed that it is possible to compute the real path from the resulting virtual path by 1-hop information. So, the algorithm can send the message to the next real node with the help of virtual nodes.

Some significant work has also been done on defining bounds on the locality of distributed routing algorithms for undirected graphs [7]. It is possible to construct various classes of graphs that require at least k-local information for providing guaranteed delivery. Bose et al. [7] give bounds on the value of k for different classes of graphs.

3.2 Directed Graph

Unlike undirected graphs, less work has been done for a local geometric routing algorithms in directed graphs. Fraser et al. [15] show the minimum memory requirement for any routing algorithm in the strongly-connected directed planar graph is $\Omega(n)$ bits. Another study provides the routing algorithm for strongly-connected directed Eulerian and directed outer-planar graphs using constant memory [9]. Chávez et al. [9] showed that the routing algorithm for the Eulerian graphs takes $O(n^2)$ steps and the routing algorithm for outer-planar graphs takes at most $2n - 1$ steps. Recently, Fraser [14] showed that the given $O(n)$ memory lower bound for directed graphs is tight. Her strategy for obtaining an $O(n)$-memory algorithm is similar to the face routing algorithm, but for the reversely directed edge it finds an outer-path to proceed. The iterative process of finding such an outer-path needs at most $O(n)$ bits of memory.

Chapter 4

Routing in Convex Subdivisions

This chapter proposes an origin-oblivious predecessor-oblivious algorithm that guarantees delivery in any convex subdivision using only a single state bit that is passed with the message. As mentioned in Chapter 3, Bose et al. [6] showed that no origin-oblivious predecessor-oblivious memoryless local routing algorithm guarantees delivery in convex subdivisions; as we now show, a single bit of memory suffices. Section 4.1 gives the outline of our proposed algorithm, and Section 4.2 gives the pseudocode. In Section 4.3 we prove that our algorithm guarantees delivery in any convex subdivision.

4.1 Outline of the Algorithm

Let $G(V, E)$ be a convex subdivision, $s \in V$ be the source vertex and $t \in V$ be the target vertex in G. Let H_L and H_R be the left and right half-planes, respectively, defined by the vertical line L_V through t. Similarly, let H_T and H_B be the top and

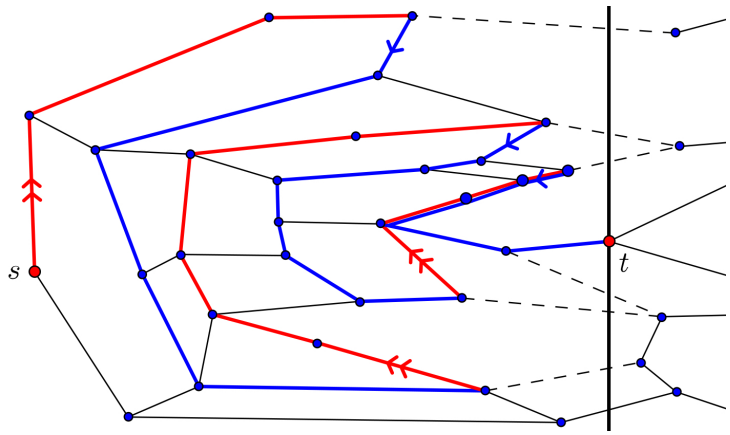


Figure 4.1: Starting from the source vertex s, Algorithm 1 traverses a set of chains, where any two consecutive chains are oppositely directed. After traversing all the chains, the message reaches to target t.

bottom half-planes, respectively, defined by the horizontal line L_H through t.

Without loss of generality, we assume that $s \in H_L$. Here is the outline of the algorithm: initially the direction bit d is set to 0 and the current vertex u is set to s. Our algorithm has two different phases:

Phase 1 (while $d = 0$) The algorithm repeatedly forwards the message to $u = ccw(u, t)$ while $u \in H_L$. Then our algorithm is switched into Phase 2 by changing the direction bit d to 1.

Phase 2 (while $d = 1$) The algorithm repeatedly forwards the message to $u' = cw(u', t)$ while $u' \in H_L$. Then our algorithm is switched into Phase 1 by changing the direction bit d to 0.

Figure 4.1 shows the path followed by our algorithm to send a message from the source vertex s to the target vertex t. In Phase 1, the message is forwarded through a

sequence of red colored edges and in Phase 2, it is forwarded through a sequence of blue colored edges. Note that some edges are used in both Phase 1 and Phase 2. After a finite number of steps the message reaches the target vertex t. Thus, the path, followed by the message can be represented as a sequence of counterclockwise and clockwise ordered chains, where two consecutive chains are always oppositely directed. We prove that the sequence of chains implies forward progress and eventually terminates by reaching the target t.

4.2 Pseudocode

If $u \in V$ is the current vertex and d is a binary variable that contains either 0 or 1, then in each step, our algorithm takes routing decision by a rule $F(u, d)$, which is defined as follows:

$$F(u,d) = \begin{cases} ccw(u,t) & \text{if } d = 0 \\ cw(u,t) & \text{if } d = 1 \end{cases}$$

Algorithm 1 gives the pseudocode of our proposed algorithm. It initially sets the direction bit d to 0. The while loop runs until t is one of the neighbors of u. If $F(u, 0) \in H_L$, the algorithm forwards the message to $F(u, 0) = ccw(u, t)$. When the message reaches a vertex u such that $F(u, 0) \in H_R$, the algorithm changes the value of d from 0 to 1 (in Line 5). The algorithm then continues forwarding the message to $u = F(u, 1) = cw(u, t)$ while $F(u, 1) \in H_L$. Again, when the message is reaches into a vertex u such that $F(u, 1) \in H_R$, the algorithm changes the value of d from 1 to 0.

Algorithm 1 Routing Algorithm for Convex Subdivision

1: Let s be the source, t be the destination node.
2: $d \leftarrow 0$
3: **while** t is not in $N(u)$ **do** ▷ t is not a neighbor of u
4: **if** $F(u, d) \in H_R$ **then**
5: $d \leftarrow 1 - d$ ▷ toggle the value of d
6: **else**
7: Send the message to $F(u, d)$.
8: $u = F(u, d)$
9: **end if**
10: **end while**
11: Forward the message to t.

4.3 Proof of Guaranteed Delivery

Let us analyze the path followed by the message. Let u_0 be the vertex that contains the message when the algorithm switches from Phase 1 to Phase 2, i.e., $u_0 \in (H_L \bigcap V)$ and $ccw(u_0, t) \in (H_R \bigcap V)$. After traversing a sequence of vertices $u_0, u_1, u_2, u_3, \ldots, u_{r-1}$ in $H_L \bigcap V$ by repeatedly applying the forwarding rule $u_{j+1} = cw(u_j, t)$ for $0 \leq j \leq r - 1$, the algorithm reaches a vertex $u_r \in (H_L \bigcap V)$ where $cw(u_r, t) \in (H_R \bigcap V)$. In this situation, the algorithm switches into Phase 1. Starting from $v_0 = u_r$, it then traverses another sequence of vertices $v_0, v_1, v_2, v_3, \ldots, v_{s-1}$ in $H_L \bigcap V$ by repeatedly applying the forwarding rule $v_{j+1} = cw(v_j, t)$ for $0 \leq j \leq s - 1$ and the algorithm reaches a vertex v_s in $H_L \bigcap H_T \bigcap V$ such that $ccw(v_s, t) \in (H_R \bigcap V)$. Our algorithm then switches into Phase 2 again and proceeds accordingly.

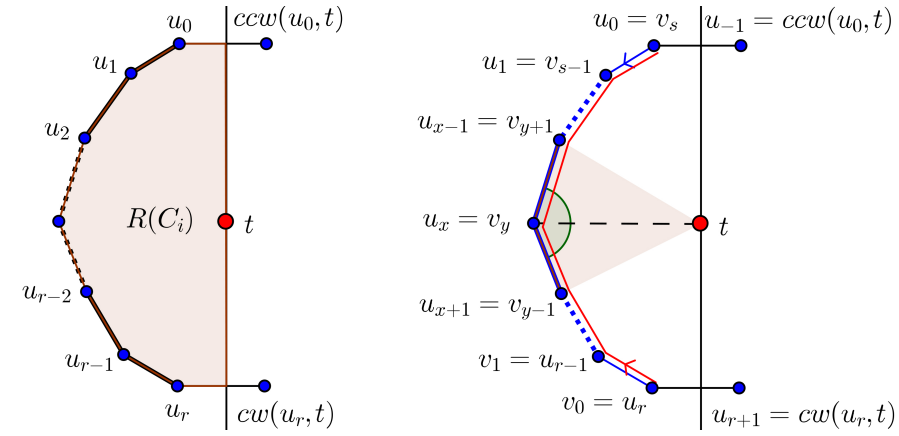
Thus, the message repeatedly traverse counterclockwise and clockwise chains (see Figure 4.1). In this section we prove that our algorithm guarantees that the message will reach the destination t in any convex subdivision. To keep the proof simpler, we first introduce some definitions.

Definition 1 *Let $C = \{C_0, C_1, C_2 \ldots, C_m\}$ in H_L be a set of ordered chains generated by Algorithm 1. For any $0 \leq i \leq m-1$, C_i and C_{i+1} are two consecutive chains, where C_i was traversed prior to traversing C_{i+1}. The source vertex s is an element in C_0. Based on the traversal order we define a notation, $\succ$ that defines the order among chains. Therefore, $C_0 \succ C_1 \succ C_2, \ldots, \succ C_m$.*

Definition 2 *A chain $C_i = u_0, u_1, u_2, \ldots, u_r$ for any $0 \leq i \leq m$, is an ordered set of vertices that starts from u_0 and ends at u_r such that there is a path from u_0 to u_r. Specifically, $u_j \in (H_L \bigcap V \bigcap C_i)$ for any $0 \leq j \leq r$ and edge $(u_j, u_{j+1}) \in E(G)$ for any $0 \leq j \leq r-1$. Let $V(C_i)$ and $E(C_i)$ denotes the set of vertices and the set of edges, respectively, in chain C_i and let $|C_i|$ denotes the number of vertices in chain C_i.*

Definition 3 *The direction of a chain $C_i = u_0, u_1, u_2, \ldots, u_r$, for any $0 \leq i \leq m$, is determined by the rule that is applied for generating the chain C_i. Chain C_i is in counterclockwise direction if $u_{j+1} = ccw(u_j, t)$ and it is in clockwise direction if $u_{j+1} = cw(u_j, t)$ for any $0 \leq j \leq r-1$.*

Property 1 *Let $C_i = u_0, u_1, u_2, \ldots, u_r$ be a chain that starts from u_0 and ends at u_r and let $C_{i+1} = v_0, v_1, v_2, \ldots v_s$ be the next chain starts at v_0 and ends at v_s, then $v_0 = u_r$ and C_i and C_{i+1} are oppositely directed (see Figure 4.1).*



(a) Region $R(C_i)$ of any chain $C_i = u_0, u_1, u_2, \ldots, u_r$.

(b) Two consecutive chains $C_i \neq C_{i+1}$

Figure 4.2: Defining the closed region $R(C_i)$ and illustration in support of Lemma 1.

Definition 4 *A vertex v is **out-of-bounds** if $v \in H_R \bigcap V$. Otherwise v is **in-bounds**. Algorithm 1 only forwards a message to a vertex that is **in-bounds**. Consequently, every vertex in a chain must be **in-bounds**.*

Definition 5 *A chain $u_0, u_1, u_2, \ldots u_r$ is **complete** if $ccw(u_0, t)$ and $cw(u_r, t)$ are out-of-bounds (if the chain has clockwise orientation) or $cw(u_0, t)$ and $ccw(u_r, t)$ are out-of-bounds (if the chain has counterclockwise orientation). Every chain, except possibly for the first and last, is complete.*

Definition 6 *For any clockwise (resp., counterclockwise) complete chain C_i with corresponding vertex sequence $u_0, u_1, u_2, \ldots u_r$, let $R(C_i)$ denotes the closed region of the plane bounded by the path $P = ccw(u_0, t), u_0, u_1, u_2, \ldots, u_r, cw(u_r, t)$ (resp., the path $P = cw(u_0, t), u_0, u_1, u_2, \ldots, u_r, ccw(u_r, t)$) and the vertical line L_v through t (see Figure 4.2a).*

Definition 7 *If $C_i = u_0, u_1, u_2, \ldots, u_r$ and $C_j = v_0, v_1, v_2, \ldots, v_s$ are two consecutive chains, then $C_i = C_j$ if and only if they share all vertices, i.e., $v_0 = u_r, v_1 = u_{r-1}, v_2 = u_{r-2}, \ldots, v_s = u_0$. As we show in Lemma 1, two chains can never be equal.*

Definition 8 *If $C_i = u_0, u_1, u_2, \ldots, u_r$ and $C_{i+1} = v_0, v_1, v_2, \ldots, v_s$ are two consecutive chains then C_{i+1} **breaches** C_i if and only if they have at least one common vertex $u_x = v_y$ such that $v_{y+1} \in H_L \setminus R(C_i)$.*

Now, for all $1 \leq i \leq m-1$, it is required to prove:

1. $C_i \neq C_{i+1}$,
2. $R(C_{i+1})$ is a proper subset of $R(C_i)$, i.e., $R(C_{i+1}) \subset R(C_i)$,
3. $t \in C_m$.

Lemma 1 *For all $0 \leq i \leq m-1$, $C_i \neq C_{i+1}$*

Proof. Let $C_i = u_0, u_1, u_2, \ldots, u_r$ and $C_{i+1} = v_0, v_1, v_2, \ldots, v_s$ be two consecutive chains. Assume without loss of generality that C_i is in clockwise direction and therefore, C_{i+1} is in counterclockwise direction. By Property 1, we already know $v_0 = u_r$. To make the proof simpler, we consider two out-of-bounds vertices u_{-1} and u_{r+1} with respect to chain C_i, where $u_{-1} = ccw(u_0, t)$ and $u_{r+1} = cw(u_r, t)$.

To drive a contradiction, we assume that two consecutive chains are equal, i.e. $C_{i+1} = C_i$. Consider three consecutive vertices $\{u_{x-1} = v_{y+1}, u_x = v_y, u_{x+1} = v_{y-1}\}$ (see Figure 4.2b). Since C_i is a clockwise chain, $u_{x+1} = cw(u_x, t)$, and the cone

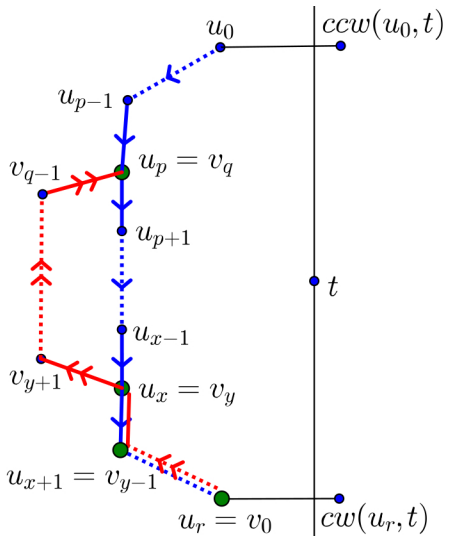


Figure 4.3: Chain $C_{i+1} = v_0, v_1, v_2, \ldots, v_s$ does not breach chain $C_i = u_0, u_1, u_2, \ldots, u_r$.

$\angle t u_x u_{x+1}$ does not contain any neighbor of u_x. On the other hand, C_{i+1} is a counterclockwise chain, $v_{y+1} = ccw(v_y, t)$ and the cone $\angle t v_y v_{y+1}$ does not contain any neighbor of v_y. Therefore, cone $\angle u_{x-1} u_x u_{x+1}$ does not contain any neighbor of $u_x = v_y$. Since, $C_i = C_{i+1}$ then for every $1 \leq x \leq r$, the cone $\angle u_{x-1} u_x u_{x+1}$ does not contain any neighbor of u_x. This argument can be applied to every vertex u_j in $C_i = C_{i+1}$ Therefore, $R(C_i)$ does not contain any vertex that is a neighbor to any vertex u_j. That contradicts that t is in a convex face. ■

Lemma 2 *For all* $1 \leq i \leq m-1$, C_{i+1} *does not breach* C_i.

Proof. Let $C_i = u_0, u_1, u_2, \ldots, u_r$ and $C_{i+1} = v_0, v_1, v_2, \ldots, v_s$ be two consecutive chains. Suppose without loss of generality that C_i is in clockwise direction and, therefore, C_{i+1} is in counterclockwise direction. By Property 1, we already know that

$v_0 = u_r$.

To derive a contradiction we assume that C_{i+1} breaches C_i at a vertex $u_x = v_y$ for some $0 < x \leq r$ and $0 < y \leq s$.

By the definition of a clockwise chain, we know that $cw(u_{x-1}, t) = u_x$. Since, C_{i+1} breaches C_i at the vertex $u_x = v_y$, $v_{y+1} \in (H_L \setminus R(C_i))$. In other words, v_{y+1} is the in the opposite half-plane (defined by the line $v_y u_{x-1}$) that contains t (see Figure 4.3). Therefore, the cone $\angle t v_y v_{y+1}$ must contain the vertex u_{x-1}. However, if the cone $\angle t v_y u_{x-1}$ contains a neighbor g' of v_y, then $v_{y+1} = ccw(v_y, t) = g'$. Otherwise $v_{y+1} = ccw(v_y, t) = u_{x-1}$. This is clearly a contradiction that v_{y+1} is the in the opposite half-plane (defined by the line $v_y u_{x-1}$) that contains t. So, C_{i+1} never breaches C_i. ■

Lemma 3 *For all $1 \leq i \leq m-1$, $R(C_{i+1})$ is a proper subset of $R(C_i)$, i.e., $R(C_{i+1}) \subset R(C_i)$.*

Proof. Let $C_i = u_0, u_1, u_2, \ldots, u_r$ and $C_{i+1} = v_0, v_1, v_2, \ldots, u_s$ be two consecutive chains. From the definition, we know that C_{i+1} appears after C_i. From Lemma 2, chain C_{i+1} cannot breach chain C_i and from Lemma 1 $C_{i+1} \neq C_i$. Therefore, $C_{i+1} \subset R(C_i)$ and $R(C_{i+1}) \subset R(C_i)$. ■

Lemma 4 *If t is the target vertex and C_m is the last chain in C, then $t \in C_m$.*

Proof. To derive a contradiction, we assume that t is not connected to chain C_m. However, since C_m is the last chain, region $R(C_m)$ does not contain any vertex from V. Because, if there exists some vertices in V inside $R(C_m)$, then at least one of them is connected to at least one vertex in C_m and that contradicts that C_m is the

last chain. Since, region $R(C_m)$ is empty, t does not have any neighbor in the left half-plane H_L, which contradicts that G is a convex subdivision. ■

Theorem 1 *Our algorithm construct a path from s to t through the a set of chains C in any convex subdivision.*

Proof. Starting from the source vertex $s \in C_0$, our algorithm traverses all the chains $C_0, C_1, C_2, \ldots C_m$ one-by-one. Whenever the algorithm updates the direction bit, d, it moves into the next chain. By Lemma 3 we know that $R(C_m) \subset R(C_{m-1}) \subset R(C_{m-2}) \subset \cdots \subset R(C_0)$. Therefore, after a finite number of steps the algorithm reaches the chain C_m. By Lemma 4, t is connected to C_m. So, Algorithm 1 delivers the message to t. ■

In the first section we assume that s is in the left half-plane H_L. If s is in the right half-plane H_R, then our algorithm generates a set of chains in the right half-plane H_R and never includes any vertex from the left half-plane H_L. Also note that the algorithm successfully delivers the message even if s and t (any one or both) are on the convex hull of the convex subdivision.

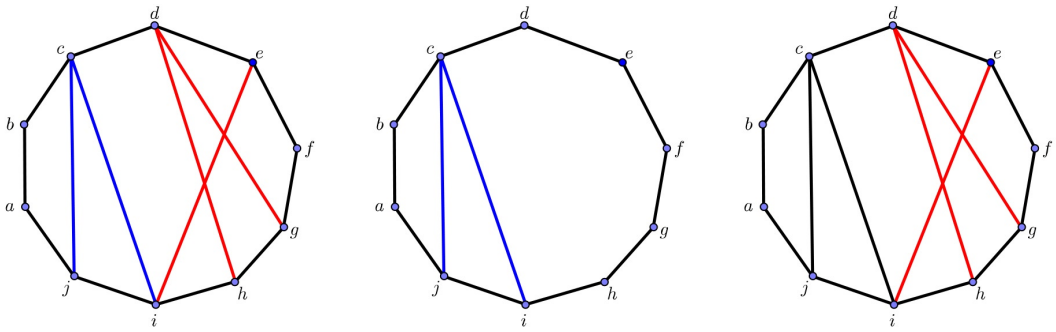
Chapter 5

Routing in Quasi Planar Convex Subdivisions

In this chapter we show that in addition to guaranteeing delivery in any convex subdivision, Algorithm 1 guarantees delivery in any k-quasi planar convex subdivision. Recall that an augmented edge in a face of a k-quasi planar convex subdivision can have at most k crossings, but no edge in the underlying convex subdivision is crossed by any augmented edge. Algorithm 1 does not depend on the value of k. That is why throughout the chapter we refer to "quasi planar convex subdivision" instead of "k-quasi planar convex subdivision".

Let $Q = (V, E, F)$ be a quasi planar convex subdivision, where V is the set of vertices, E is the set of edges, and F is the set of convex faces. The edges in E can be categorized into three different groups (see Figure 5.1a):

1. convex subdivision edges: Edges that are taken from the underlying convex subdivision (black edges in Figure 5.1a).



(a) Black edges are convex subdivision edges, blue edges are non-crossing augmented edges and red edges are crossing augmented edges.

(b) Blue edges can also be considered as convex subdivision edges.

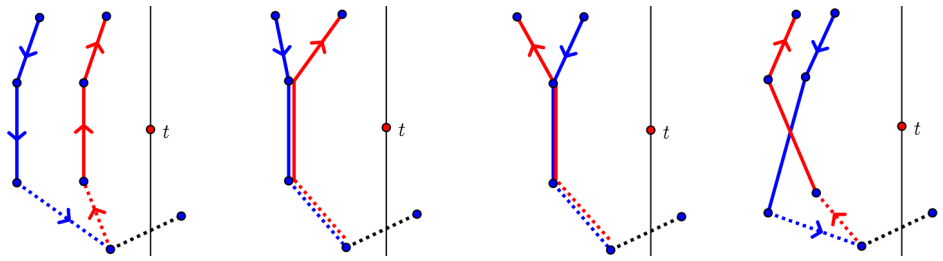
(c) Partitioning of edges into two sets, where black edges are convex subdivision edges and red edges are augmented edges.

Figure 5.1: Different type of edges in a quasi planar convex subdivision.

2. Non-crossing augmented edges: Edges that are augmented but do not cross any other edges (blue edges in Figure 5.1a).

3. Crossing augmented edges: Edges that are augmented and cross with other augmented edges (red edges in Figure 5.1a).

A non-crossing augmented edge subdivides the convex face into two sub-faces, each of which is also convex. For example, in Figure 5.1b, two non-crossing augmented edges, (c, j) and (c, i), divide the face $(a, b, c, d, e, f, g, h, i, j)$ into three sub-faces: (a, b, c, j), (j, c, i) and (c, d, e, f, g, h, i). So, if we replace the face $(a, b, c, d, e, f, g, h, i, j)$ by three convex faces, (a, b, c, j), (j, c, i) and (c, d, e, f, g, h, i), then the graph still remains a convex subdivision. Let $G = (V, E, F')$ be a quasi planar convex subdivision, where V and E are the set of vertices and edges from Q, and F' is the set of resultant



(a) Shared vertex. (b) Shared vertices. (c) **Breaches**. (d) **Crosses**.

Figure 5.2: Relationship between two consecutive chains C_i (blue) and C_{i+1} (red).

faces after maximally partitioning all the faces in F with non-crossing augmented edges. Note that this partitioning is neither unique nor is it known to the routing algorithm. It only simplifies the proof of guaranteed delivery. We prove that Algorithm 1 guarantees delivery in G, and hence in Q. Therefore, rest of this chapter assume that there are only two types of edges in G: convex subdivision edges (black colored edges) and augmented edges (red colored edges) (see Figure 5.1c).

In Chapter 4, we said two chains $C_i = u_0, u_1, u_2, \ldots, u_r$ and $C_{i+1} = v_0, v_1, v_2, \ldots v_s$ **breach** each other if and only if they share a common vertex $u_p = v_q$ such that vertex v_{q-1} in chain C_{i+1} lies inside the closed region $R(C_i)$ and vertex v_{q+1} lies outside the closed region $R(C_i)$ (see Figure 5.2c). In this chapter we give a similar definition for chain crossings.

Definition 9 *If $C_i = u_0, u_1, u_2, \ldots, u_r$ and $C_{i+1} = v_0, v_1, v_2, \ldots, v_s$ are two consecutive chains, then C_{i+1} **crosses** C_i if and only if for some $0 < i < r$ and some $0 < j < s$, edge $(v_j, v_{j+1}) \in E(C_{i+1})$ crosses edge $(u_i, u_{i+1}) \in E(C_i)$ as well as v_j and v_{j+1} are in the interior and exterior, respectively, of region $R(C_i)$ (see Figure 5.2d).*

Definition 10 *Let $LH(u, v)$ and $RH(u, v)$ be the left half-plane and right half-plane,*

respectively, defined by the line through u and v, oriented according to the vector $\overrightarrow{uv}$.

In order to prove that our algorithm guarantees delivery in any quasi planar convex subdivision, we closely look at the face structure and its relationship with the set of chains constructed by Algorithm 1.

All convex subdivision edges in G are crossing-free. Each augmented edge joins two vertices in a face of the underlying convex subdivisions in G and, furthermore, each augmented edge has one or more crossings, each of which is with another augmented edge on the same face.

A convex subdivision edge (u, v) borders on two faces in the underlying convex subdivision. The edge (u, v) determines two half-planes, one of which contains t. The face adjacent to (u, v) contained in the half-plane that contains t is the *forward* face of (u, v). An augmented edge (u, v) crosses the face of the underlying convex subdivision. This face is its *forward* face (it only has one face).

For any chain $C_i = u_0, u_1, u_2, \ldots, u_r$, let $F = f_0, f_1, f_2, f_3, \ldots, f_z$ be the sequence of forward faces of the edges in $E(C_i)$. We categorize all the forward faces in F into two groups: terminal faces and intermediate faces. A face $f_k \in F$ is a *terminal* face if it contains either the first vertex u_0 or the last vertex u_r and its interior is intersected by the vertical line L_v. The rest of the faces in F are called *intermediate* faces. Any chain C_i has exactly two terminal faces: f_0 and f_z. Let $R'(C_i)$ denote the closed region of the plane bounded by F and the vertical line L_v through t.

Figure 5.3 shows a chain $C_i = u_0, u_1, u_2, \ldots u_{10}$ that has eight forward faces associated with it. Faces f_0 and f_7 are terminal faces and all the other forward faces are intermediate faces. Region $R'(C_i)$ is illustrated by the region bounded by the solid

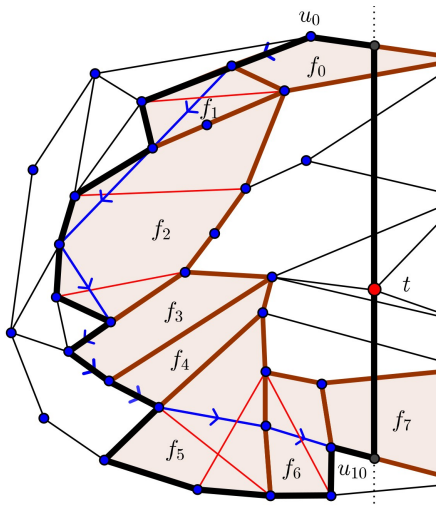


Figure 5.3: A chain C_i is shown by directed blue lines. All the shaded faces are the forward faces of C_i and the closed region $R'(C_i)$ is bounded by the solid bold black line.

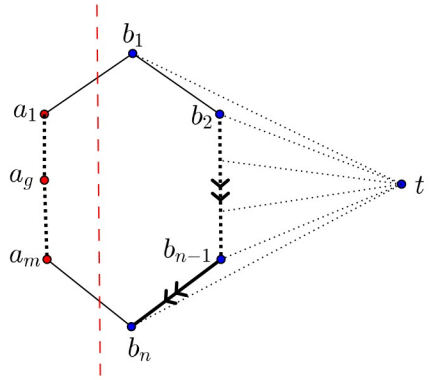


Figure 5.4: Example of internal vertices $(a_1, a_2, a_3, \ldots a_m)$ and external vertices $(b_1, b_2, b_3, \ldots b_n)$ of an intermediate face f_k.

bold black line.

Any vertex u in face f_k is an *internal* vertex of f_k if the line segment (u, t) intersects the closed region of f_k (i.e. u is any vertex between a_1 and a_m in Figure 5.4). On the other hand, u is an *external* vertex of f_k if the line segment (u, t) does not intersect

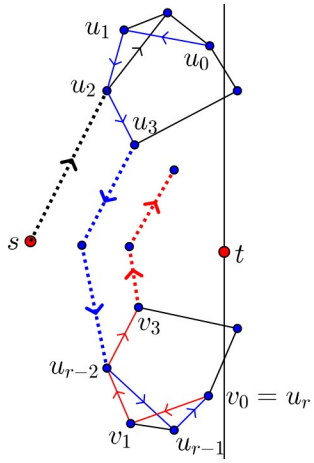
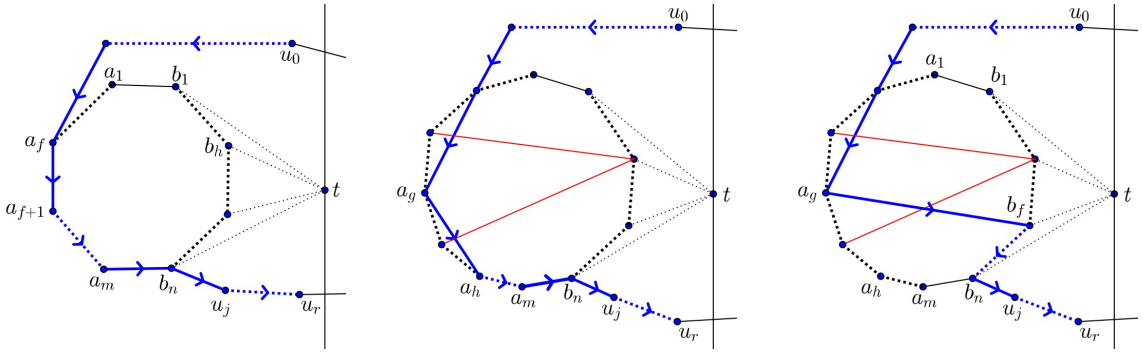


Figure 5.5: Schematic view of chains in a quasi planar convex subdivision. Starting from the source vertex s, the algorithm traverses a set of chains. Two consecutive chains C_i (blue) and C_{i+1} (red) are oppositely directed and might cross only in their common terminal face.

the closed region of f_k (i.e. u is any vertex between b_1 and b_m in Figure 5.4). Since f_k is convex, internal and external vertex sets can be separated by a line (red dashed line in Figure 5.4). Let us partition all the vertices of f_k into two counterclockwise ordered sets around t: $A = \{a_1, a_2, a_3, \ldots, a_m\}$ and $B = \{b_1, b_2, b_3, \ldots, b_n\}$, where A is the internal vertex set and B is the external vertex set of f_k (see Figure 5.4).

Any chain C_i traverses (or includes) some internal and some external vertices from a face f_k. **Traversing** a face $f_k \in F$ by chain C_i means visiting one or more internal vertices of f_k. **Entering** into a face $f_k \in F$ by a chain C_i means visiting an internal vertex a_g for any $1 \leq g \leq m$ from f_k in C_i for the first time. **Leaving** a face f_k by chain C_i means visiting an external vertex from f_k in C_i for the last time.

Starting from a terminal face f_0 a chain C_i, generated by Algorithm 1 traverses some intermediate faces from f_1 to f_{z-1} and then ends in a terminal face f_z (see



(a) C_i visits some internal vertices and an external vertex b_n.

(b) C_i includes some augmented edges and then visits b_n.

(c) C_i includes some augmented edges and then visits b_f.

Figure 5.6: Chain C_i (represented by blue edges) always leaves f_k through an external vertex.

Figure 5.5). The next chain C_{i+1} starts from a terminal face g_0 and ends after traversing some forward faces $g_1, g_2, \ldots, g_y$. Chains C_i and C_{i+1} might share the terminal face $f_z = g_0$. However, later in this chapter, we prove that no other forward face from f_0 to f_{z-1} is common to both C_i and C_{i+1}.

Property 2 *In any step, if a clockwise chain C_i includes the last external vertex b_n of a face $f_k \in F$, then in the next step, C_i does not include any vertex from f_k.*

Proof. Since face f_k is convex, all the vertices in f_k are in the right half-plane $RH(t, b_n)$ (see Figure 5.6a-5.6c). Since there is an underlying convex subdivision in any quasi planar convex subdivision, there must be at least one neighbor of b_n in the left half-plane $LH(t, b_n)$. So, $u_j = cw(b_n, t)$ cannot be a vertex in f_k. ■

Property 3 *Let $F = f_0, f_1, f_2, \ldots f_z$ be the sequence of forward faces of a chain C_i. For any face $f_k \in F$, where $0 \leq k \leq z-1$, C_i leaves f_k through an external vertex.*

Proof. Assume without loss of generality that C_i is a clockwise chain and it enters into face f_k through an internal vertex a_f. From the definition of an internal vertex, we know that the line segment (a_f, t) intersects the closed region of f_k.

We show all the possible ways by which chain C_i may leave face f_k. There are two possible cases to consider:

Case 1: If C_i does not include any augmented edge in f_k and $cw(a_f, t)$ is another internal vertex of f_k, then $cw(a_f, t) = a_{f+1}$ (see Figure 5.6a). C_i includes (traverses) all the internal vertices $a_f, a_{f+1}, a_{f+2}, \ldots a_m$. However, $cw(a_m, t) = b_n$ is the last external vertex (by Property 2).

Case 2: Now we consider the case in which chain C_i includes one or more augmented edges from the same face f_k (see Figure 5.6b–5.6c). Consider the last augmented edge e included by the chain C_i from f_k. There are two different sub-cases to consider:

Sub-case 1: Assume that $e = (a_g, a_h)$, i.e., C_i is traversed from an internal vertex a_g to another internal vertex a_h (see Figure 5.6b). Since a_h is an internal vertex and (a_g, a_h) is the last augmented edge traversed by C_i from f_k, C_i includes all the internal vertices from a_h to a_m and then includes b_n (analogous to Case 1).

Sub-case 2: Assume that $e = (a_g, b_f)$, i.e., C_i is traversed from an internal vertex a_g to an external vertex b_f (see Figure 5.6c). If the cone $\angle t b_f b_{f+1}$

contains a neighbor of b_f, then b_f is the external vertex through which C_i leaves. However, if the cone $\angle t b_f b_{f+1}$ does not contain any neighbor of b_f, then b_{f+1} will be included in the chain. Similarly, if the cone $\angle t b_{f+1} b_{f+2}$ does not contain any of neighbor of b_{f+1}, then b_{f+2} will also be included in the chain. This way, C_i may include at most all the external vertices from b_f to b_n.

Therefore, by Property 2, C_i always includes at least one external vertex of f_k and then leaves f_k. ■

Note that Property 3 does not hold for the last terminal face f_z because, the last external vertex b_n of face f_z is in region H_R. We know from Chapter 4 that no chain includes vertices from region H_R. Since the next chain C_{i+1} starts from the last vertex of chain C_i, i.e., $v_0 = u_r$, C_{i+1} and C_i might share the same terminal face $g_0 = f_z$ and C_{i+1} might cross C_i in that terminal face (see Figure 5.5).

Property 4 *Let $e = (u, v)$ be an augmented edge in a face f_k. No chain traverses e from u to v if u is an external vertex of f_k.*

Proof. Assume without loss of generality that C_i is a clockwise chain, u is an external vertex and v is any vertex (either external or internal) in face f_k. To derive a contradiction we assume that chain C_i includes e (see Figure 5.7). We need to consider following two different cases.

Case 1: Suppose u is the last external vertex in face f_k, i.e., $u = b_n$. By Property 2, no vertex from face f_k is traversed by any clockwise chain C_i after traversing

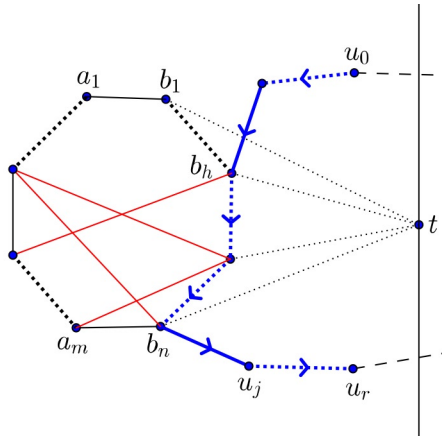


Figure 5.7: Chain C_i is represented by blue edges and augmented edges in a face f_k is represented by red edges. No augmented edge will be included by chain C_i if C_i visits the endpoint of some augmented edge as the external vertex of f_k.

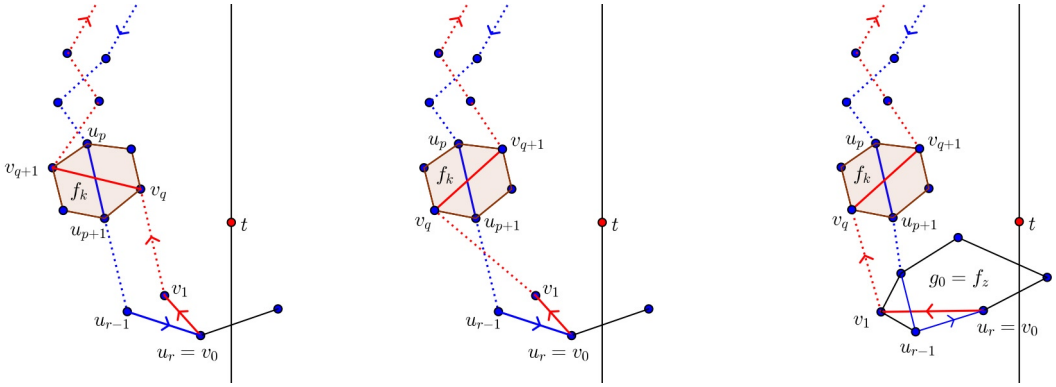
the last external vertex b_n. This contradicts that chain C_i traverses from u to v.

Case 2: Suppose u is not the last external vertex. Let b_h be any external between b_1 to b_{n-1} and $u = b_h$. b_h has one neighbor b_{h+1}, which is also an external vertex in f_k. If the cone $\angle t b_h b_{h+1}$ does not contain any neighbor of b_h, then $cw(b_h, t) = b_{h+1}$. This also contradicts that C_i traverse from u to v.

So, C_i does not traverse any augmented edge $e = (u, v)$ from u to v if u is an external vertex of f_k. ■

To prove that Algorithm 1 guarantees delivery in any quasi convex subdivision, we show that for all $1 \leq i \leq m-1$:

1. $R'(C_{i+1})$ is a proper subset of $R'(C_i)$, i.e., $R'(C_{i+1}) \subset R'(C_i)$,
2. $t \in C_m$.



(a) Edge (v_q, v_{q+1}) will not be included (traversed) by the red chain since v_q is an external vertex of f_k.

(b) Edge (v_q, v_{q+1}) may be included by the red chain but this contradicts that C_{i+1} crosses C_i for the first time.

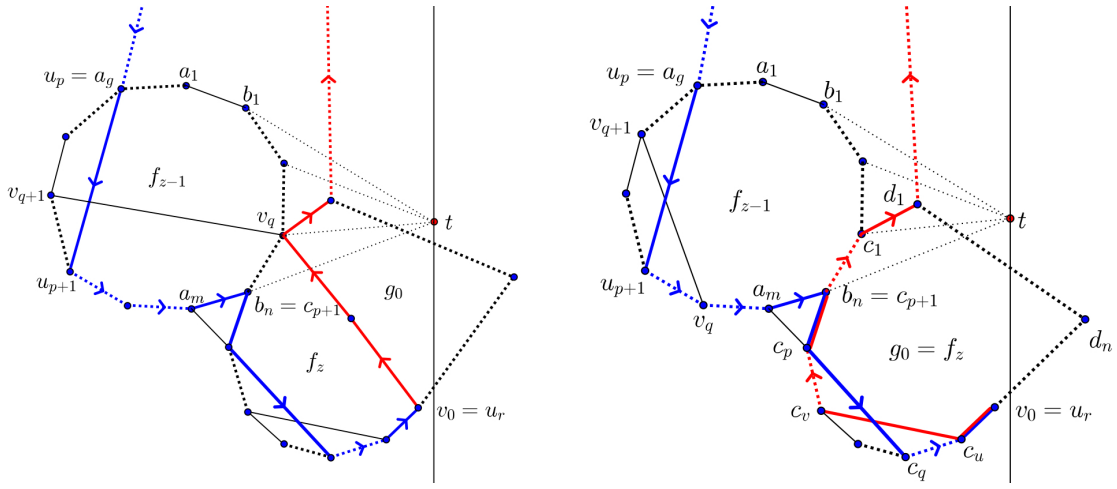
(c) Chain C_{i+1} might cross chain C_i in $g_0 = f_z$. To derive a contradiction in Property 5 we assume that C_{i+1} enters into face f_k through a path from v_1 to v_q.

Figure 5.8: C_{i+1} (red) does not cross C_i (blue) in any face between f_0 and f_{z-1}

Property 5 *If $F = f_0, f_1, f_2, \ldots, f_{z-1}, f_z$ is a set of forward faces of chain C_i, then C_{i+1} does not cross C_i in any face between f_0 and f_{z-1}.*

Proof. Assume without loss of generality that $C_i = u_0, u_1, u_2, \ldots, u_r$ is a clockwise chain and $C_{i+1} = v_0, v_1, v_2, \ldots, v_s$ is a counterclockwise chain. Let $G = \{g_0, g_1, g_2, \ldots g_y\}$ be the set of forward faces of chain C_{i+1}.

To derive a contradiction we assume that chain C_{i+1} crosses chain C_i in some face between f_0 to f_{z-1}. Consider the first occurrence where chain C_{i+1} crosses chain C_i at a crossing between the edges $e_1 = (u_p, u_{p+1})$ and $e_2 = (v_q, v_{q+1})$ in face $f_k \in F$. In some previous step, C_i entered into the face $f_k \in F$ by an internal vertex and left f_k by an external vertex (by Property 3). Since $e_2 = (v_q, v_{q+1})$ is an augmented edge in



(a) C_i and C_{i+1} does not share any forward face.

(b) Chain C_i and C_{i+1} shares a terminal face.

Figure 5.9: Chain C_i is represented by blue edges and C_{i+1} is represented by red edges. $R'(C_{i+1})$ is the proper subset of region $R'(C_i)$, i.e., $R'(C_{i+1}) \subset R'(C_i)$.

f_k, there are two cases to consider:

Case 1: If v_q is an external vertex, then by Property 4 we know that e_2 will not be included by C_{i+1}. This contradicts our assumption (see Figure 5.8a).

Case 2: If v_q is an internal vertex, then C_{i+1} needs to cross C_i in the sub-chain from v_0 and v_q which is contradicting that this is the first occurrence of crossings (see Figure 5.8b). If we repeatedly apply this argument to the subsequent face from f_{k+1} to f_{z-1}, then chain C_{i+1} can cross chain C_i in f_k only one possible way. That is, C_{i+1} must cross C_i in face $g_0 = f_z$ and then enters into face f_k after traversing a set of faces that are outside of region $R'(C_i)$ (see Figure 5.8c).

Case 3: Let C_{i+1} cross C_i in face $g_0 = f_z$. Consider the last two faces f_{z-1} and f_z of

chain C_i in Figure 5.9b. C_i enters into face f_{z-1} through an internal vertex a_g and leaves f_k through an external vertex $b_n = c_{p+1}$, which is also the internal vertex of face $f_z = g_0$. Now consider the chain C_{i+1} that starts from $v_0 = u_r$. By Property 3, we know that chain C_{i+1} leaves g_0 by an external vertex d_k ($k = 1$ in Figure 5.9b). Any external vertex of face $g_0 = f_z$ must be inside the region $R(C_i)$, and again by Case 1, this contradicts that C_{i+1} enters into f_k.

Therefore, C_{i+1} does not cross C_i in any face between f_0 and f_{z-1} . ■

Lemma 5 *For any two consecutive chains C_i and C_{i+1}, $R'(C_{i+1})$ is a proper subset of $R'(C_i)$, i.e., $R'(C_{i+1}) \subset R'(C_i)$.*

Proof. Assume without loss of generality that $C_i = u_0, u_1, u_2, \ldots, u_r$ is a clockwise chain and $C_{i+1} = v_0, v_1, v_2, \ldots, v_s$ is a counterclockwise chain. Let $F = f_0, f_1, f_2, \ldots, f_{z-1}, f_z$ and $G = g_0, g_1, g_2, \ldots g_y$ be the set of forward faces, respectively, of chain C_i and chain C_{i+1}. Recall from Chapter 4 that $R(C_i)$ denotes the closed region bounded by the path $P = ccw(u_0, t), u_0, u_1, u_2, \ldots, u_r, cw(u_r, t)$ and the vertical line L_v through t. From the definition $R(C_i)$ and $R'(C_i)$ we can say $R(C_i) \subseteq R'(C_i)$. Let us recall the Property 5.

Firstly, if $g_0 \neq f_z$, then $R'(C_{i+1}) \subset R(C_i) \subseteq R'(C_i)$ (see Figure 5.9a). Because, C_{i+1} never goes out of the region $R(C_i)$ by crossing any forward face between f_0 and f_z.

Secondly, if $g_0 = f_z$, then C_{i+1} might cross C_i inside $g_0 = f_z$ but $g_0 \subset R'(C_i)$ (see Figure 5.9b). C_{i+1} leave g_0 by an external vertex $d_1 \in R(C_i)$. The

sub-chain $v_q = d_1, v_{q+1}, v_{q+2}, \ldots, v_s$ never goes outside the region $R(C_i)$ by crossing any forward face between f_0 and f_{z-1}. So, $C_{i+1} \subset R'(C_i)$, and therefore, $R'(C_{i+1}) \subset R'(C_i)$.

■

Lemma 6 *If t is the target vertex and C_m is the last chain in C in a quasi planar convex subdivision, then $t \in C_m$.*

Proof. This Lemma follows by an argument analogous to that used to prove Lemma 4 in Chapter 4. ■

Theorem 2 *Algorithm 1 guarantees delivery in any quasi planar convex subdivision.*

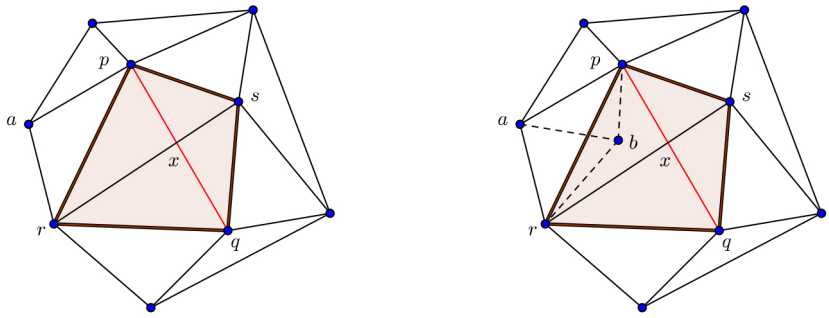
Proof. Starting from the source vertex s, Algorithm 1 forwards the message through a set of chains $C_0, C_1, C_2, C_3, \ldots C_m$. By Lemma 5 we can say that $R'(C_m) \subset R'(C_{m-1}) \subset R'(C_{m-2}) \cdots \subset R'(C_1)$. Therefore, after a finite number of steps algorithm reaches the chain C_m. By Lemma 6, t is connected to C_m. So, Algorithm 1 delivers the message to t. ■

Chapter 6

Routing in 2-Augmented Triangulations

This chapter examines local routing algorithms for 2-augmented triangulations. Section 6.1 shows that every 1-augmented triangulation is a 1-quasi planar convex subdivision, and therefore, greedy-compass routing works in any 1-augmented triangulation (or in any 1-quasi convex subdivision). Recall that greedy-compass routing is memoryless, origin oblivious, and predecessor oblivious. Section 6.2 shows that both greedy-compass routing and Algorithm 1 fail in some 2-augmented triangulations. The same section proposes a variation of Algorithm 1 and proves that the new algorithm guarantees delivery in any 2-augmented triangulation.

Let $G = (V, E \bigcup E')$ be a k-augmented triangulation (see Chapter 2.5.2) where the set of edges is partitioned into two subsets: E and E', where E is the set of triangulation edges and E' is the set of augmented edges. Note that this partition is neither unique nor is it known to the routing algorithm. It is used only to show



(a) Env(p,q) in a 1-augmented triangulation is {p,r,q,s}.

(b) Illustration in support of Lemma 7.

Figure 6.1: Envelope in a 1-augmented triangulation.

guaranteed delivery.

We first define the term **envelope** for any augmented edge in a k-augmented triangulation. Note that the term envelope is only defined for augmented edges, not for triangulation edges. The envelope $Env(p, q)$ of an augmented edge $(p, q) \in E$ is the innermost clockwise cycle in $E \setminus \{(p, q)\}$ (smallest cycle by area) that contains (p, q) defined in terms of the sequence of edges on the boundary of the faces of the triangulation that intersect (p, q) (see Figure 6.1a). The length of the envelope is the number of vertices in the envelope, denoted by $|Env(p, q)|$.

6.1 1-Augmented Triangulations

Lemma 7 *Any edge with* 1 *crossing in a* 1*-augmented triangulation* G *has an envelope of length* 4*.*

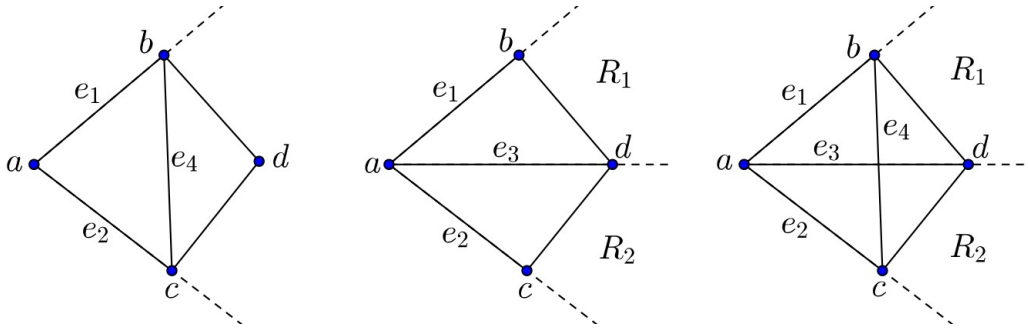
Proof. Assume without loss of generality that an augmented edge (p, q) crosses a triangulation edge (r, s) in a point x. We have to prove that the envelope, $Env(p, q) = \{p, r, q, s\}$. In other words, we have to prove that $\{(p, r), (r, q), (q, s), (s, p)\} \subset E$ (see Figure 6.1b). To derive a contradiction, assume that the edge (p, r) is crossed by some edge (a, b). One endpoint b of the edge (a, b) is inside $\triangle pxr$; otherwise, (a, b) would intersect either (p, q) or (r, s), which leads to a contradiction that G is 1-augmented. On the other hand, b cannot be incident to an edge other than (a, b) that crosses (p, r) because (p, r) can have at most one crossing edge (a, b). Therefore, b can only be connected to p and r but that is not sufficient to complete the triangulation. This is a contradiction that G has an underlying triangulation. By an analogous argument it follows that $(r, q), (q, s)$ and (s, p) all are triangulation edges and therefore, there is an envelope of length 4 around the augmented edge (p, q). ■

Proposition 1 *Any 1-augmented triangulation is a 1-quasi planar convex subdivision.*

Proof. Let G be a 1-augmented triangulation. If we delete all the crossing edges from G, then we find a convex subdivision G' where all the faces in G' are either triangles or convex quadrilaterals. At most two edges can be augmented in each convex quadrilateral and that will give at most 1-crossing per edge. Therefore, G' is the underlying convex subdivision and G is a 1-quasi planar convex subdivision. ■

Lemma 8 *The greedy-compass routing succeeds in any 1-augmented triangulation*[1].

[1]Lemma 8 and its proof were discussed with the members of the Computational Geometry Laboratory at the University of Manitoba, including Stephane Durocher, Saeed Mehrabi, Debajyoti Mondal and Matthew Skala



(a) No augmented edge between edge e_1 and edge e_2

(b) No augmented edge (b, c)

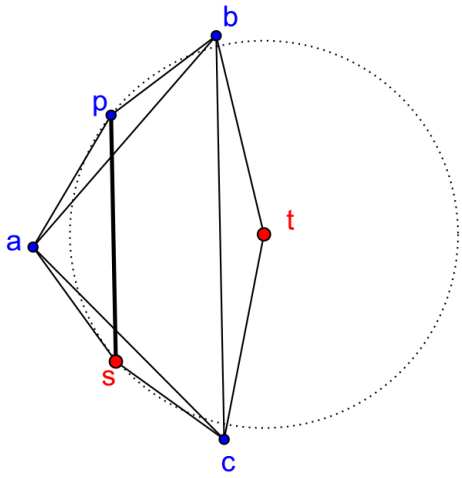
(c) One edge is augmented

Figure 6.2: Greedy-compass routing succeeds in 1-augmented triangulation.

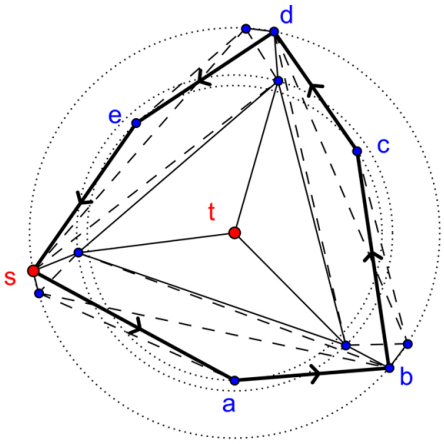
Proof. Let us consider a 1-augmented triangulation $G = (V, E \bigcup E')$ and let $T = (V, E)$ be an underlying triangulation in G, i.e., $T \subseteq G$. It is already known that greedy-compass routing guarantees delivery in T [6]. Let P be the route from s to t in G generated by greedy-compass routing on T.

Let the target vertex t be lies in one of the region, R_1 and R_2, as defined in Figure 6.2c. At any vertex $a \in P$, there are three possible edges, (a, b), (a, c), (a, d), two of which are compass neighbors of a. After forwarding the message to one of them, the edge (b, c) will never be a candidate edge. Therefore, the route taken by the message corresponds to that taken in the triangulation given by removing the edge e_4. So, greedy-compass routing succeeds in 1-augmented triangulations.

■



(a) A 2-augmented triangulation defeats greedy-compass routing



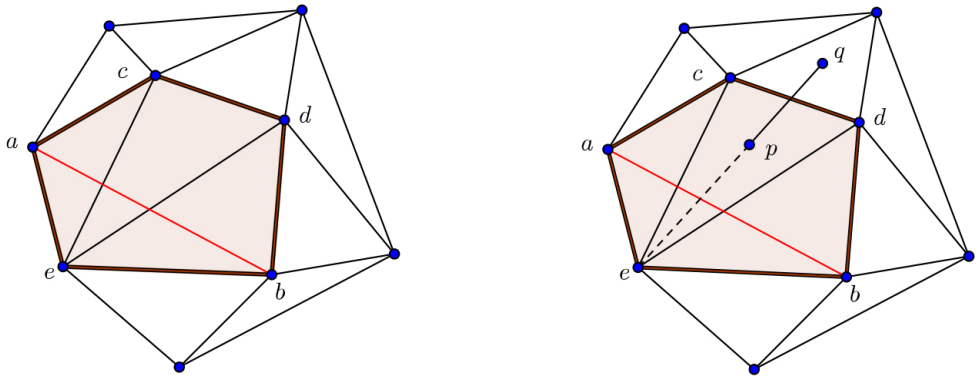
(b) A 2-augmented triangulation defeats predecessor-aware greedy-compass routing

Figure 6.3: s is the source vertex and t is target vertex. Greedy-compass routing is defeated by some 2-augmented triangulations

6.2 2-Augmented Triangulations

6.2.1 Greedy-compass

Although greedy-compass routing succeeds in 1-augmented triangulations, it fails in 2-augmented triangulations. Figure 6.3a shows a scenario where greedy-compass routing becomes stuck in a loop. It forwards the message from the source vertex s to vertex p and then from p to s again. This loop could be avoided if greedy-compass routing had the knowledge of the predecessor vertex. However, predecessor-aware greedy-compass routing still fails in some 2-augmented triangulations. For example, in Figure 6.3b, predecessor-aware greedy-compass routing becomes stuck in a cycle



(a) $Env(a, b) = \{a, c, d, b, e\}$. (b) Illustration in support of Property 6.

Figure 6.4: Envelope of an edge with 2 crossings (a, b) in a 2-augmented triangulation.

of length 6.

Before proposing our new algorithm, we give one of the most important properties of 2-augmented triangulations.

Property 6 *Any edge* (a, b) *with* 2 *crossings in a* 2*-augmented triangulation has an envelope of length* 5*.*

Proof. If (a, b) is an edge with 2 crossings in a 2-augmented triangulation, then it intersects three underlying triangles (see Figure 6.4a). Based on the geometric properties, these three triangulations can be drawn in the plane only two different ways: i). two vertices lie in the left half-plane $LH(a, b)$ and one vertex lies in the right half-plane $RH(a, b)$, or ii). one vertex lies in $LH(a, b)$ and two vertices lie in $RH(a, b)$.

Figure 6.4a illustrates an example, where augmented edge (a, b) intersects three triangles $\triangle aec$, $\triangle ced$, and $\triangle deb$. We claim that $\{c, d, b, e, a\}$ is the envelope of edge

(a, b). It is sufficient to prove that all the edges in $\{(c, d), (d, b), (b, e), (e, a), (a, c)\}$ are triangulation edges.

To derive a contradiction we assume that edge (c, d) is an augmented edge that crosses a triangulation edge $e = (p, q)$ (see Figure 6.4b). If p lies inside one of the three triangles intersected by (a, b) (shaded region in Figure 6.4b), then it is not possible to complete the triangulation by maintaining the constraint of 2-augmented triangulations. If $p = e$, then (p, q) crosses (a, b) and (a, b) has 3 crossings, which is clearly a contradiction. Again, if $p = a$ (respectively, b), then (p, q) crosses another triangulation edge (c, e) (respectively, (d, e)), which contradicts that (p, q) is a triangulation edge. Therefore, (c, d) must be a triangulation edge. Analogous arguments shows that $\{(a, c), (c, d), (d, b), (b, e), (e, a)\}$ are triangulation edges.

■

Note that any edge on the envelope can be crossed by an augmented edge. For example, in Figure 6.5, the envelope of the augmented edge (s, a) is $\{s, c, a, t, d\}$, where one edge (d, t) on the envelope is crossed by another edge (s, b). However, (s, b) is an augmented edge and (d, t) is a triangulation edge (since (d, t) is an edge in $Env(s, a)$).

6.2.2 Updated Algorithm

The algorithm we proposed in Chapter 4 (Algorithm 1) is defeated by a 2-augmented triangulation. In Figure 6.5, both the $cw(s, t)$ and $ccw(s, t)$ is in H_R. Although s is the only vertex in the last chain generated by Algorithm 1, it is not connected to t.

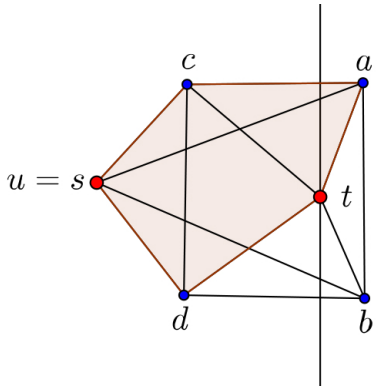
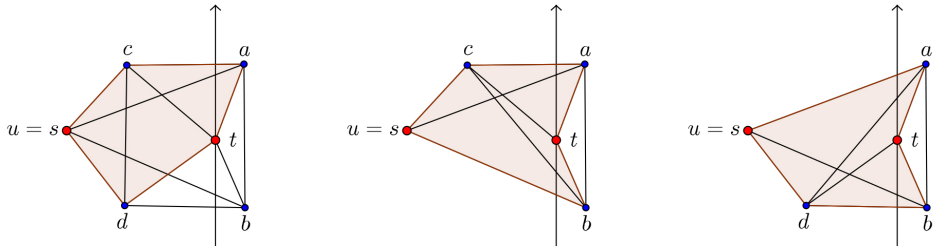


Figure 6.5: A 2-augmented triangulation defeats Algorithm 1

However, we claim that a small variation of Algorithm 1 works for 2-augmented triangulations. Algorithm 1 fails in a 2-augmented triangulation if both the candidate $cw(u,t)$ and $ccw(u,t)$ of the current vertex u lie in H_R as well as both $(u, cw(u,t))$ and $(u, ccw(u,t))$ are augmented edges. However, we claim that both $cw(u,t)$ and $ccw(u,t)$ is connected to t. Therefore, including two extra vertices from H_R in each chain solves this problem.

Lemma 9 *Let C_m be the last chain generated by Algorithm 1 in a 2-augmented triangulation and u be an element in C_m. If u is not connected to t and both $a = cw(u,t)$ and $b = ccw(u,t)$ are in the right half-plane H_R, then edge (t,a) and edge (t,b) are triangulation edges.*

Proof. Since there is an underlying triangulation in a 2-augmented triangulation, there is at least one vertex in the left plane H_L that is connected to t. Since u is not connected to t, either (u,a) or (u,b) or both edges have 2 crossings. For instance, Figure 6.6a-6.6c shows these three possibilities. To derive a contradiction we assume that t is not connected to a. There are two cases to consider:



(a) Both (u, a) and (u, b) are augmented edges.

(b) (u, a) is an augmented edge.

(c) (u, b) is an augmented edge.

Figure 6.6: Illustration in support of Lemma 9.

Case 1: (u, a) is an augmented edge with 2 crossings. The vertices in $Env(u, a)$ can have one of the two different formations, 1). two vertices, c and p, lie in $LH(u, a)$ and one vertex, i.e., t lies in $RH(u, a)$, or 2). one vertex lies c lies in $LH(u, a)$ and two vertices p and t lie in $RH(u, a)$. The envelope $Env(u, a)$ must contain four vertices $\{u, c, a, t\}$. It also contains one additional vertex p. As an example, see Figure 6.6a and Figure 6.6b. There are two possible cases to consider for vertex p:

Sub-case 1: Assume that p lies in the right half-plane $RH(u, a)$ ($p = d$ in Figure 6.6a and $p = b$ in Figure 6.6b). We know that t is not connected to u. By Property 6, t must be connected to both $p = b$ and a, which is a contradiction.

Sub-case 2: Now assume that p lies in the left half-plane $LH(u, a)$. Then t is the only vertex in the envelope $Env(u, a)$ that lies in the right half-plane $RH(u, a)$. Therefore, t must be connected to u which is clearly a contradiction.

Case 2: (u, b) is an augmented edge with 2 crossings. An analogous argument as shown in Case 1 proves that (t, b) is a triangulation edge (see Figure 6.6a or Figure 6.6c).

Since either edge (u, a) or edge (u, b) or both are augmented edges with 2 crossings, (t, a) and (t, b) are triangulation edges.

■

Before describing the algorithm in details, we define two additional terms.

Definition 11 *Let u be any vertex that has some neighbor in the opposite half-plane defined by the vertical line L_v through t. $ccw'(u, t)$ is the first counterclockwise neighbor of any vertex u starting from the vector $\overrightarrow{ut}$ such that the edge $(u, ccw'(u, t))$ intersects L_v. Similarly $cw'(u, t)$ is the first clockwise neighbor of any vertex u starting from the vector $\overrightarrow{ut}$ such that the edge $(u, cw'(u, t))$ intersects L_v.*

Without loss of generality we assume that $s \in H_L$. Here is the outline of the algorithm: initially the direction bit d is set to 0 and current vertex u is set to s. Our algorithm has two different phases:

Phase 1 (while $d = 0$): If $u \in (H_B \bigcap H_R)$, then our algorithm sends the message to $ccw'(u, t)$ to bring the message back to H_L. Otherwise, it repeatedly forwards the message to $u = ccw(u, t)$. If the message reaches a vertex in $H_T \bigcap H_R$, then the algorithm switches to Phase 2 by changing the direction bit d to 1.

Phase 2 (while $d = 1$): If $u \in (H_T \bigcap H_R)$, then our algorithm sends the message to $cw'(u, t)$ to bring the message back to H_L. Otherwise, it repeatedly forwards

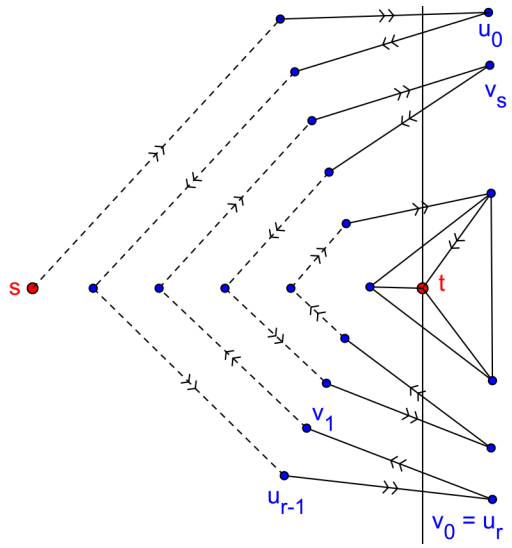


Figure 6.7: Schematic view of chains generated by Algorithm 2. Starting from the source vertex s, Algorithm 2 traverses a set of chains. The first and last vertex of every chain (except possibly C_0) are in the right half-plane R_H.

the message to $u = cw(u,t)$ until the message reaches a vertex in $H_R \bigcap H_B$. Then our algorithm switches to Phase 1 by changing the direction bit d to 0.

If $u \in V$ is the current vertex and d is a binary variable that contains either 0 or 1, then in each step, our updated algorithm (Algorithm 2) takes routing decisions by a rule $F(u,d)$, which is defined as follows:

$$F(u,d) = \begin{cases} ccw(u,t) & \text{if } d = 0 \text{ and } u \in H_L \\ ccw'(u,t) & \text{if } d = 0 \text{ and } u \in H_R \\ cw(u,t) & \text{if } d = 1 \text{ and } u \in H_L \\ cw'(u,t) & \text{if } d = 1 \text{ and } u \in H_R \end{cases}$$

Similar to Algorithm 1, the path followed by the message can be represented

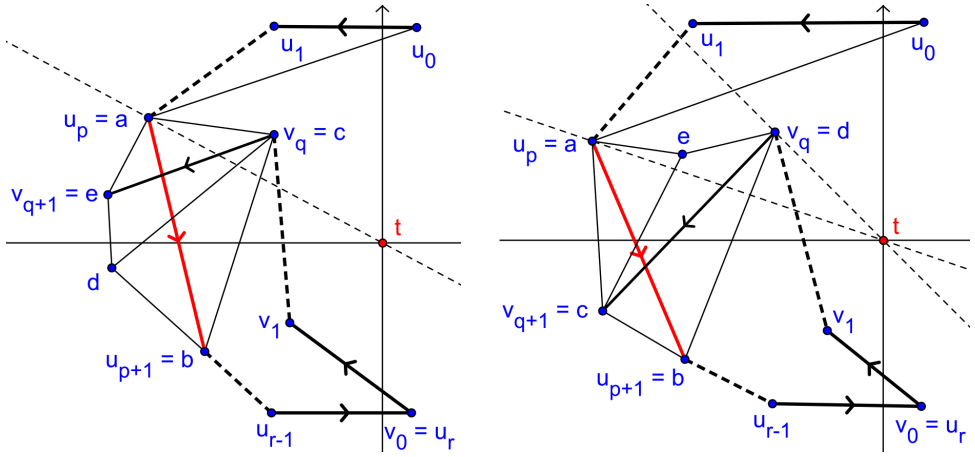
Algorithm 2 Routing algorithm for 2-augmented triangulation

1: Let s be the source, t be the destination node.
2: $d \leftarrow 0$
3: **while** t is not in $N(u)$ **do** ▷ t is not a neighbor of u
4: **if** $u \in H_R$ **then**
5: $d \leftarrow 1 - d$ ▷ toggle the value of d
6: **end if**
7: Send the message to $F(u, d)$.
8: $u = F(u, d)$
9: **end while**
10: Forward the message to t.

as a set of clockwise (cw) and counterclockwise (ccw) ordered chains, where two consecutive chains are always oppositely directed. The properties of chains generated by this new algorithm are almost same as those given in Chapter 4. However, unlike the chains generated by Algorithm 1, two consecutive chains generated by Algorithm 2 share exactly one vertex in H_R (see Figure 6.7).

Proposition 2 *No two consecutive chains breach each other in 2-augmented triangulations.*

Proof. We know there is an underlying triangulation in 2-augmented triangulation. So, this Lemma follows by an argument analogous to that used to prove Lemma 2 in Chapter 4. ■



(a) One vertex $c \in RH(u_{p+1}, u_p)$ and two vertices $\{d, e\} \subset LH(u_{p+1}, u_p)$.

(b) Two vertices $\{d, e\} \subset RH(u_{p+1}, u_p)$ and one vertex $c \in RH(u_{p+1}, u_p)$.

Figure 6.8: (u_p, u_{p+1}) is an augmented edge and (v_q, v_{q+1}) is a triangulation edge.

Lemma 10 *No two consecutive chains, generated by Algorithm 2 in 2-augmented triangulations, cross.*

Proof. Let us consider two chains $C_i = \{u_0, u_1, u_2, \ldots, u_r\}$ and $C_{i+1} = \{v_0 = u_r, v_1, v_2, \ldots, v_s\}$, where C_i is clockwise. To derive a contradiction we assume that some edge (u_p, u_{p+1}) in C_i for some $0 \leq p \leq r-1$, is crossed by some edge (v_q, v_{q+1}) in C_{i+1} for some $0 \leq q \leq s-1$, for the first time while generating the chain C_{i+1}. We consider all possible formations of the envelope of an augmented edge and show that edge (v_q, v_{q+1}) does not cross edge (u_p, u_{p+1}) in any of these formations.

Case 1 Assume that (u_p, u_{p+1}) is an augmented edge and (v_q, v_{q+1}) is a triangulation edge, i.e., $u_p = a$ and $u_{p+1} = b$ in Figure 6.8. Consider the formations of vertices in $Env(a, b)$ in the following cases.

Case 1a: Consider that one vertex $c = v_q$ lies in $RH(b, a)$ and two vertices, d and e, lie in $LH(b, a)$ (see Figure 6.8a). That means, c is directly connected to both a and b. According to the definition of chain C_i, c does not lie in the cone $\angle tab$. So, c lies in $RH(t, a)$. Since edge (c, e) crosses edge (a, b), e does not lie in $LH(t, c)$, which contradicts that $e = ccw(v_q, t)$.

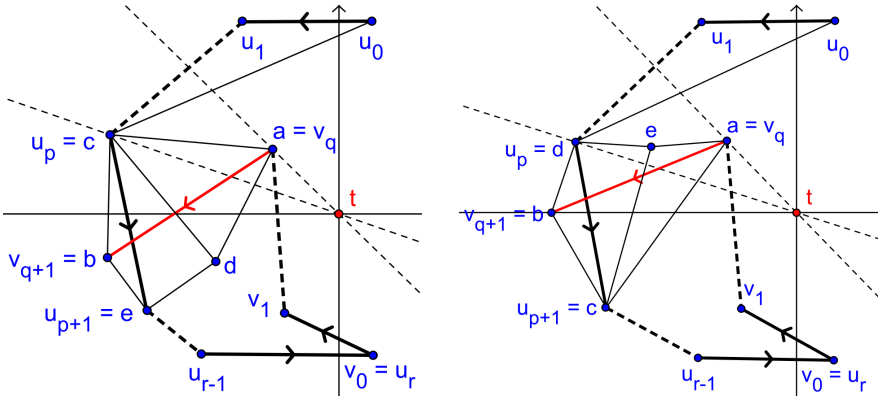
Case 1b: This case considers that two vertices, d and e, lie in $RH(b, a)$ and one vertex $c = v_{q+1}$ lies in $LH(b, a)$ (see Figure 6.8b). There are two possibilities for v_q:

$v_q = d$**:** Due to the envelope $Env(a, b)$, d is connected to e (see Figure 6.8b). According to the definition of chain C_i, $u_{p+1} = cw(u_p, t)$. So, vertex e does not lie in the cone $\angle tab$. If d lies in $LH(t, a)$, then c does not lie in the cone $\angle tde$. This contradicts that $c = ccw(v_q, t)$. Again if d lies in $RH(t, a)$, then due to the underlying triangulation, there must be at least one neighbor of d that lies in $RH(t, d)$. This also contradicts that $c = ccw(v_q, t)$.

$v_q = e$**:** Due to the envelope $Env(a, b)$, v_q is directly connected to u_p. So, the analogous argument shown in Case 1a contradicts that $c = ccw(v_q, t)$.

Case 2: Assume that (u_p, u_{p+1}) is a triangulation edge and (v_q, v_{q+1}) is an augmented edge, i.e., $v_q = a$ and $v_{q+1} = b$ in Figure 6.9. Again consider the formations of vertices in $Env(a, b)$ in the following cases.

Case 2a: In this case we consider that one vertex $c = u_p$ lies in $RH(a, b)$ and



(a) One vertex $c \in RH(v_q, v_{q+1})$ and two vertices $\{d, e\} \subset LH(v_q, v_{q+1})$.

(b) Two vertices $\{d, e\} \subset RH(v_q, v_{q+1})$ and one vertex $c \in LH((v_q, v_{q+1})$.

Figure 6.9: (u_p, u_{p+1}) is a triangulation edge and (v_q, v_{q+1}) is an augmented edge.

two vertices, d and e, lie in $LH(a, b)$ (see Figure 6.9a). So, u_p is directly connected to both v_q and v_{q+1}.

According to the definition of chain C_i, a does not lie in the cone $\angle tce$. So, a lies in $RH(t, c)$. Since edge (a, b) crosses edge (c, e), b must lie in $LH(t, a)$. Since there is an underlying triangulation, there must be at least one neighbor of a that lies in $RH(t, a)$. This contradicts that $b = ccw(v_q, t)$.

Case 2b: This case considers that two vertices, d and e, lie in $RH(a, b)$ and one vertex $c = u_{p+1}$ lies in $LH(a, b)$ (see Figure 6.9b). There are two possibilities for u_p:

$u_p = d$**:** Due to the envelop $Env(a, b)$, d is connected to e (see Figure 6.9b).

Analogous argument shown in Case 1b contradicts that $b = ccw(v_q, t)$.

$u_p = e$**:** Since a is directly connected to e, analogous argument shown in

Case 2a can be applied. This contradicts that $b = ccw(v_q, t)$.

Analogous arguments also sufficient if we consider (u_p, u_{p+1}) is an augmented edge with 1 crossing and (u_p, u_{p+1}) connects two vertices on the envelope of an augmented edge with 2 crossings. Therefore, C_{i+1} does not cross chain C_i. ■

Theorem 3 *Algorithm 2 guarantees delivery in 2-augmented triangulations.*

Proof. Let us consider a set of chain C generated by Algorithm 2. Lemma 10 proves that no two consecutive chains C_i and C_{i+1} cross each other by the augmented edge. Since there is an underlying triangulation, $C_i \neq C_{i+1}$ (analogous arguments to Lemma 1 in Chapter 4). Therefore, $R(C_{i+1}) \subset R(C_i)$. Lemma 9 proves that last chain C_m is connected to t. Therefore, Algorithm 2 guarantees delivery in 2-augmented triangulations. ■

Chapter 7

Conclusion

In this thesis we studied geometric local routing algorithms for some edge-augmented planar graphs. Section 7.1 summarizes the major contributions of this thesis and Section 7.2 proposes some related open research problems.

7.1 Summary

In Chapter 4, we proposed an origin-oblivious predecessor-oblivious routing algorithm (Algorithm 1) that requires only 1 state bit passed with the message. Algorithm 1 utilizes geometric information to guarantee delivery of messages in convex subdivisions. It follows a sequence of chains in the half-plane (defined by the vertical line through the target vertex t) that contains the source vertex s.

Algorithm 1 is simple and straightforward. It is easy to implement and efficient in computational time and space requirements. At each node u in the route, it only requires computing $cw(u, t)$ and $ccw(u, t)$, which takes only constant time.

It is known that no memoryless origin-oblivious predecessor-oblivious routing algorithm guarantees delivery in convex subdivisions [6]. One of the best known routing algorithms is face routing, which is predecessor aware [5]. Any predecessor aware routing algorithm can be thought of as a predecessor-oblivious routing algorithm by using the coordinates of the predecessor vertex as message overhead. However, doing so requires $\Omega(\log n)$ bits of memory. As there is no constant-memory origin-oblivious predecessor-oblivious routing algorithm known that guarantees delivery in convex subdivisions, Algorithm 1 fills this gap. Moreover, the memory bound of the algorithm is tight because it requires only one bit of memory to carry with the message.

In Chapter 5 we proved that Algorithm 1 is powerful enough to guarantee delivery in quasi planar convex subdivisions. So far, the best known algorithm that guarantees delivery in any quasi planar convex subdivision is provided by Kranakis et al. [21]. Their algorithm requires $\Omega(\log n)$ bits of memory as message overhead in the worst case. However, 1 bit of memory is sufficient for our algorithm (Algorithm 1) to achieve the same result.

In Chapter 6 we studied routing algorithms in 2-augmented triangulations. We showed that a small modification to Algorithm 1 gives us a guaranteed delivery routing algorithm (Algorithm 2) for 2-augmented triangulations. To the best of our knowledge, no algorithm for 2-augmented triangulation is known that guarantees delivery in 2-augmented triangulations. Therefore, Algorithm 2 also gives an upper bound on the memory that is required for any algorithm to guarantee delivery in 2-augmented triangulations.

Algorithm 2 follows the same sequence of chains as does Algorithm 1, but adds one

additional vertex in each chain. Therefore, it guarantees delivery in all three classes of graphs: convex subdivisions, quasi planar convex subdivisions and 2-augmented triangulations. Algorithm 2 is clearly origin oblivious, predecessor oblivious and requires one bit of memory.

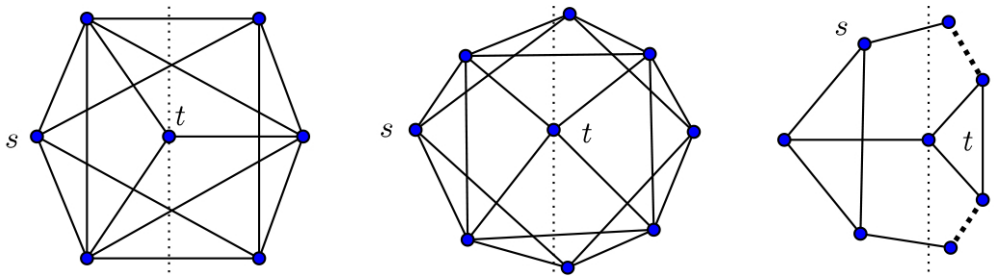
7.2 Future Work and Open Problems

Algorithm 2 requires only 1 bit of memory but it is origin oblivious, predecessor oblivious, and guarantees delivery in many different classes of graphs: convex subdivision, k-quasi planar graph and 2-augmented triangulation. However, we do not know the **competitive ratio** of Algorithm 2. The route generated by face routing is a sequence of edges from the intersecting faces with line st. Therefore, face routing might give a better competitive ratio than our algorithm does. So, one future research problem could be improving the competitive ratio of Algorithm 2.

Open Problem 1. Find the competitive ratio of Algorithm 2 and improve it.

We know that greedy-compass routing is oblivious but fails in some 2-augmented triangulations. On the other hand, Algorithm 2 shows that an upper bound of one bit on the additional memory necessary. So, the natural question is whether Algorithm 2 is tight.

Open Problem 2. Find an oblivious routing algorithm that guarantees delivery in 2-augmented triangulations. If it is not possible, then show that Algorithm 2 is tight in terms of message overhead.



(a) Loop in a 3-augmented triangulation.

(b) Loop in another 3-augmented triangulation.

(c) Loop in a 1-augmented convex subdivision.

Figure 7.1: Algorithm 2 fails both in some 3-augmented triangulation and in a 1-augmented convex subdivision.

Algorithm 2 fails in some 3-augmented triangulations. Figure 7.1a and Figure 7.1b give two such examples. In Algorithm 2, instead of constructing chains in one half-plane, if we apply the forwarding rule $cw(u, t)$ and $ccw(u, t)$ in both L_R and H_R, then we can show that the message gets stuck in a loop in some 3-augmented triangulations. Note that if we delete all the crossing edges from a 3-augmented triangulation, then the target vertex t (in both examples) becomes disconnected from the graph. Therefore, if an algorithm avoids all the crossing edges, it will definitely fail in these two examples. Therefore, finding an oblivious algorithm for 3-augmented triangulation is a challenging task.

Open Problem 3. Find a constant-memory origin-oblivious predecessor-oblivious routing algorithm that provides guaranteed delivery in any 3-augmented triangulation or show that no such algorithm is possible.

To the best of our knowledge, there is no constant-memory origin-oblivious predecessor-oblivious routing algorithm known for k-augmented triangulations, for any finite value of k. Moreover, It is natural to ask if there is any bound of k for any k-augmented triangulation, where no algorithm guarantees delivery.

Open Problem 4. Find a constant-memory origin-oblivious predecessor-oblivious routing algorithm that guarantees delivery in any k-augmented triangulation for all fixed k or find a bound k such that no such algorithm exists.

To the best of our knowledge, no constant-memory origin-oblivious predecessor-oblivious routing algorithm is known that guarantees delivery in k-augmented convex subdivisions for a fixed value of k. Our algorithm fails in a 1-augmented convex subdivision that is shown in Figure 7.1c.

Open Problem 5. Find a constant-memory origin-oblivious predecessor-oblivious routing algorithm that provides guaranteed delivery in any k-augmented convex subdivision or find a bound k such that no such algorithm exists.

Several routing algorithms guarantee delivery in planar graphs and use $O(\log n)$ bits of memory as message overhead (see Section 3.1.3). Therefore, the upper bound on the message overhead required is $O(\log n)$. However, the lower bound on the memory required by any geometric local routing algorithm to guarantee delivery in planar graphs is still unknown.

Open Problem 6. Prove that the lower bound of memory required by any algorithm to guarantee delivery in any planar graph is $\Omega(\log n)$.

Bibliography

[1] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, and Barry A. Woodward. A distance routing effect algorithm for mobility (dream). In *Proceedings of the fourth annual ACM/IEEE international conference on Mobile computing and networking*, pages 76–84, 1998. (Cited on page 28.)

[2] Mark de Berg, Marc Van Kreveld, Rene Van Oostrum, and Mark Overmars. Simple traversal of a subdivision without extra storage. *International Journal of Geographical Information Science*, 11(4):359–373, 1997. doi: 10.1080/136588197242310. URL http://www.tandfonline.com/doi/abs/10.1080/136588197242310. (Cited on pages 26 and 27.)

[3] Prosenjit Bose and Pat Morin. An improved algorithm for subdivision traversal without extra storage. In Gerhard Goos, Juris Hartmanis, Jan Leeuwen, D.T. Lee, and Shang-Hua Teng, editors, *Algorithms and Computation*, volume 1969 of *Lecture Notes in Computer Science*, pages 444–455. Springer Berlin Heidelberg, 2000. ISBN 978-3-540-41255-7. doi: 10.1007/3-540-40996-3_38. URL http://dx.doi.org/10.1007/3-540-40996-3_38. (Cited on page 27.)

[4] Prosenjit Bose and Pat Morin. Online routing in triangulations. In *Proceedings of*

the 10th International Symposium on Algorithms and Computation, ISAAC '99, pages 113–122, London, UK, UK, 1999. Springer-Verlag. ISBN 3-540-66916-7. URL http://dl.acm.org/citation.cfm?id=646342.686747. (Cited on pages 8, 22, 24, and 27.)

[5] Prosenjit Bose, Pat Morin, Ivan Stojmenović, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*, DIALM '99, pages 48–55, New York, NY, USA, 1999. ACM. ISBN 1-58113-174-7. doi: 10.1145/313239.313282. URL http://doi.acm.org/10.1145/313239.313282. (Cited on pages 5, 13, 21, 25, 26, and 72.)

[6] Prosenjit Bose, Andrej Brodnik, Svante Carlsson, ErikD. Demaine, Rudolf Fleischer, Alejandro Lpez-Ortiz, Pat Morin, and J.Ian Munro. Online routing in convex subdivisions. 1969:47–59, 2000. doi: 10.1007/3-540-40996-3_5. URL http://dx.doi.org/10.1007/3-540-40996-3_5. (Cited on pages 8, 24, 31, 58, and 72.)

[7] Prosenjit Bose, Paz Carmi, and Stephane Durocher. Bounding the locality of distributed routing algorithms. In *Proceedings of the 28th ACM symposium on Principles of distributed computing*, PODC '09, pages 250–259, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-396-9. doi: 10.1145/1582716.1582756. URL http://doi.acm.org/10.1145/1582716.1582756. (Cited on pages 12 and 29.)

[8] Mark Braverman. On ad hoc routing with guaranteed delivery. *CoRR*, abs/0804.0862, 2008. (Cited on page 27.)

[9] E. Chávez, S. Dobrev, E. Kranakis, J. Opatrny, L. Stacho, and J. Urrutia. Route discovery with constant memory in oriented planar geometric networks. *Netw.*, 48(1):7–15, August 2006. ISSN 0028-3045. doi: 10.1002/net.v48:1. URL http://dx.doi.org/10.1002/net.v48:1. (Cited on page 30.)

[10] Dan Chen, Luc Devroye, Vida Dujmović, and Pat Morin. Memoryless routing in convex subdivisions: Random walks are optimal. *Computational Geometry: Theory and Applications*, 45(4):178–185, May 2012. ISSN 0925-7721. doi: 10.1016/j.comgeo.2011.12.005. URL http://dx.doi.org/10.1016/j.comgeo.2011.12.005. (Cited on page 25.)

[11] Jinhee Chun, Akiyoshi Shioura, TruongMinh Tien, and Takeshi Tokuyama. A unified view to greedy geometric routing algorithms in ad hoc networks. In Amotz Bar-Noy and MagnsM. Halldrsson, editors, *Algorithms for Sensor Systems*, volume 7718 of *Lecture Notes in Computer Science*, pages 54–65. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-36091-6. doi: 10.1007/978-3-642-36092-3_7. URL http://dx.doi.org/10.1007/978-3-642-36092-3_7. (Cited on page 23.)

[12] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*, chapter 9, pages 183–207. Springer, second edition, 2008. (Cited on page 17.)

[13] Gregory G. Finn. Routing and Addressing Problems in Large Metropolitan-Scale Internetworks, 1987. (Cited on pages 3 and 22.)

[14] Maia Fraser. Tight linear lower memory bound for local routing in planar digraphs. In *Canadian Conference on Computational Geometry*, Charlottetown, PEI, 2012. (Cited on page 30.)

[15] Maia Fraser, Evangelos Kranakis, and Jorge Urrutia. Memory requirements for local geometric routing and traversal in digraphs. In *Canadian Conference on Computational Geometry*, 2008. (Cited on page 30.)

[16] Xiaoyang Guan. *Face routing in wireless ad-hoc networks*. Phd thesis, University of Toronto, 2009. (Cited on page 29.)

[17] Elliot D. Kaplan and Christopher J. Hegarty. *Understanding GPS: principles and applications*. Library of Congress Cataloging-in-Publication Data, 2nd edition, 2006. (Cited on page 2.)

[18] Brad Karpand and H. T. Kung. Gpsr: greedy perimeter stateless routing for wireless networks. In *MobiCom'00: Proceedings of the 6-th annual international conference on Mobile computing and networking*, pages 243–254, 2000. (Cited on page 28.)

[19] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. In *Proceedings of the fourth annual ACM/IEEE international conference on Mobile computing and networking*, pages 66–75, 1998. (Cited on page 28.)

[20] Evangelos Kranakis, Harvinder Singh, and Jorge Urrutia. Compass routing on geometric networks. In *11-th Canadian Conference on Computational Geometry*, pages 51–54, Vancouver, August 1999. (Cited on pages 22, 23, and 24.)

[21] Evangelos Kranakis, Tim Mott, and Ladislav Stacho. Constant memory routing in quasi-planar and quasi-polyhedral graphs. volume 156, pages 3430–3442, 2008. doi: http://dx.doi.org/10.1016/j.dam.2008.01.027. URL http://www.sciencedirect.com/science/article/pii/S0166218X08000516. (Cited on pages 27 and 72.)

[22] Fabian Kuhn, Roger Wattenhofer, Yan Zhang, and Aaron Zollinger. Geometric ad-hoc routing: of theory and practice. In *Proceedings of the Twenty-second Annual Symposium on Principles of Distributed Computing (PODC'03)*, pages 63–72. ACM Press, 2003. (Cited on page 28.)

[23] Fabian Kuhn, Rogert Wattenhofer, and Aaron Zollinger. Ad-hoc networks beyond unit disk graphs. In *Proceedings of the 2003 joint workshop on Foundations of mobile computing*, DIALM-POMC '03, pages 69–78, New York, NY, USA, 2003. ACM. ISBN 1-58113-765-6. doi: 10.1145/941079.941089. URL http://doi.acm.org/10.1145/941079.941089. (Cited on page 29.)

[24] Kevin M. Lillis, Sriram V. Pemmaraju, and Imran A. Pirwani. Topology control and geographic routing in realistic wireless networks. In *Proceedings of the 6th international conference on Ad-hoc, mobile and wireless networks*, ADHOC-NOW'07, pages 15–31, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-

540-74822-9. URL http://dl.acm.org/citation.cfm?id=1771050.1771053. (Cited on page 29.)

[25] X. Lin and I. Stojmenovi. Geographic distance routing in ad hoc wireless networks. Technical report tr-98-10, site, University of Ottawa, December 1998. (Cited on page 22.)

[26] D. W. Matula and R. R. Sokal. Properties of gabriel graphs relevant to geographic variation research and the clustering of points in the plane. *Geographical Analysis*, 12:205–222, 1980. doi: 10.1111/j.1538-4632.1980.tb00031.x. (Cited on page 17.)

[27] Patric Ryan Morin. *Online routing in geometric graph*. Phd thesis, Carleton University, 2001. (Cited on pages 1, 2, 25, and 26.)

[28] Rajeev Motwani and Prabhakar Raghavan. Randomized algorithms. *ACM Comput. Surv.*, 28(1):33–37, March 1996. ISSN 0360-0300. doi: 10.1145/234313.234327. URL http://doi.acm.org/10.1145/234313.234327. (Cited on page 25.)

[29] Weisheng Si and A.Y. Zomaya. Midpoint routing algorithms for delaunay triangulations. In *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, pages 1–7, 2010. doi: 10.1109/IPDPS.2010.5470471. (Cited on page 23.)

[30] S. Toumpis and D. Toumpakaris. Wireless ad hoc networks and related topologies: applications and research challenges. *Elektrotechnik und Informationstech-*

nik, 123(6):232–241, 2006. ISSN 0932-383X. doi: 10.1007/s00502-006-0348-9. URL http://dx.doi.org/10.1007/s00502-006-0348-9. (Cited on page 1.)

[31] Jorge Urrutia, Ladislav Stacho, Evangelos Kranakis, Jaroslav Opatrny, Tefan Dobrev, and Edgar Chvez. Traversal of a quasi-planar subdivision without using mark bits. *Journal of Interconnection Networks*, 05(04):395–407, 2004. doi: 10.1142/S0219265904001234. URL http://www.worldscientific.com/doi/abs/10.1142/S0219265904001234. (Cited on page 27.)