Routing in Polygonal Domains*

Bahareh Banyassady[1], Man-Kwun Chiu[2,3], Matias Korman[4], Wolfgang Mulzer[1], André van Renssen[2,3], Marcel Roeloffzen[2,3], Paul Seiferth[1], Yannik Stein[1], Birgit Vogtenhuber[5], and Max Willert[1]

1 **Institut für Informatik, Freie Universität Berlin, Germany**
`{bahareh,mulzer,pseiferth,yannikstein,willerma}@inf.fu-berlin.de`
2 **National Institute of Informatics (NII), Tokyo, Japan**
`{chiumk,andre,marcel}@nii.ac.jp`
3 **JST, ERATO, Kawarabayashi Large Graph Project**
4 **Tohoku University, Sendai, Japan**
`mati@dais.is.tohoku.ac.jp`
5 **Institute of Software Technology, Graz University of Technology, Graz, Austria**
`bvogt@ist.tugraz.at`

Abstract

We consider the problem of routing a data packet through the visibility graph of a polygonal domain P with n vertices and h holes. We may preprocess P to obtain a *label* and a *routing table* for each vertex. Then, we must be able to route a data packet between any two vertices p and q of P, where each step must use only the label of the target node q and the routing table of the current node.

For any fixed $\varepsilon > 0$, we present a routing scheme that always achieves a routing path that exceeds the shortest path by a factor of at most $1 + \varepsilon$. The labels have $\mathcal{O}(\log n)$ bits, and the routing tables are of size $\mathcal{O}((\varepsilon^{-1} + h)\log n)$. The preprocessing time is $\mathcal{O}(n^2 \log n + hn^2 + \varepsilon^{-1}hn)$. It can be improved to $\mathcal{O}(n^2 + \varepsilon^{-1}n)$ for simple polygons.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems–Geometrical problems and computations

Keywords and phrases polygonal domains, routing scheme, small stretch, Yao graph

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2017.10

1 Introduction

Routing is a crucial problem in distributed graph algorithms [11, 22]. We would like to preprocess a given graph G in order to support the following task: given a data packet that lies at some *source* node p of G, route the packet to a given *target* node q in G that is identified by its *label*. We expect three properties from our routing scheme: first, it should be *local*, i.e., in order to determine the next step for the packet, it should use only information stored with the current node of G or with the packet itself. Second, the routing scheme should be *efficient*, meaning that the packet should not travel much more than the shortest

* MC, AvR and MR were supported by JST ERATO Grant Number JPMJER1201, Japan. BB was supported in part by DFG project MU/3501-2. MK was supported in part by KAKENHI Nos. 15H02665 and 17K12635, Japan. PS was supported in part by DFG project MU/3501-1. YS was supported by the DFG within the research training group 'Methods for Discrete Structures' (GRK 1408) and by GIF grant 1161.

28th International Symposium on Algorithms and Computation (ISAAC 2017).
Editors: Yoshio Okamoto and Takeshi Tokuyama; Article No. 10; pp. 10:1–10:12
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

path distance between p and q. The ratio between the length of the routing path and the shortest path in the graph is also called *stretch*. Third, it should be *compact*: the total space requirement should be as small as possible.

There is an obvious solution: for each node v of G, we store at v the complete shortest path tree for v. Thus, given the label of a target node w, we can send the packet for one more step along the shortest path from v to w. Then, the routing scheme will have perfect efficiency, sending each packet along a shortest path. However, this method requires that each node stores the entire topology of G, making it not compact. Thus, the challenge lies in finding the right balance between the conflicting goals of compactness and efficiency.

Thorup and Zwick introduced the notion of a *distance oracle* [30]. Given a graph G, the goal is to construct a compact data structure to quickly answer *distance queries* for any two nodes in G. A routing scheme can be seen as a distributed implementation of a distance oracle [24].

The problem of constructing a compact routing scheme for a general graph has been studied for a long time [1, 3, 7–9, 23, 24]. One of the most recent results, by Roditty and Tov, dates from 2016 [24]. They developed a routing scheme for a general graph G with n vertices and m edges. Their scheme needs to store a poly-logarithmic number of bits with the packet, and it routes a message from s to t on a path with length $\mathcal{O}(k\Delta + m^{1/k})$, where Δ is the shortest path distance between s and t and $k > 2$ is any fixed integer. The routing tables use $mn^{\mathcal{O}(1/\sqrt{\log n})}$ total space. In general graphs, any efficient routing scheme needs to store $\Omega(n^c)$ bits per node, for some constant $c > 0$ [22]. Thus, it is natural to ask whether there are better algorithms for specialized graph classes. For instance, trees admit routing schemes that always follow the shortest path and that store $\mathcal{O}(\log n)$ bits at each node [10, 25, 29]. Moreover, in planar graphs, for any fixed $\varepsilon > 0$, there is a routing scheme with a poly-logarithmic number of bits in each routing table that always finds a path that is within a factor of $1 + \varepsilon$ from optimal [28].

Another approach is called *geometric routing*. Here, the graph is embedded in a geometric space and the routing algorithm has to determine the next vertex for the data packet based on the knowledge of the source and target vertex, the current vertex, and its neighbourhood, see for instance [5, 6] and references therein. A recent result by Bose et al. [6] is very close to our setting. They show that under certain conditions, no geometric routing scheme can achieve stretch $o(\sqrt{n})$.

Here, we consider the class of visibility graphs of a polygonal domain P with h holes and n vertices. Two vertices p and q in P are connected by an edge if and only if they can *see* each other, i.e., if and only if the line segment between p and q is contained in the (closed) region P. The problem of computing a shortest path between two vertices in a polygonal domain has been well-studied in computational geometry [2, 4, 12, 13, 16, 17, 19–21, 26, 27, 31]. Nevertheless, to the best of our knowledge, prior to our work there have been no routing schemes for visibility graphs of polygonal domains that fall into our model. For any $\varepsilon > 0$, our routing scheme needs $\mathcal{O}((\varepsilon^{-1} + h) \log n)$ bits in each routing table, and for any two vertices s and t, it produces a routing path that is within a factor of $1 + \varepsilon$ of optimal. This constitutes a dramatic improvement over traditional geometric routing. Thus, we believe that it makes sense to look for compact routing schemes for geometrically defined graphs.

2 Preliminaries

Let $G = (V, E)$ be an *undirected*, *connected* and *simple* graph. In our model, G is embedded in the Euclidean plane: a *node* $p = (p_x, p_y) \in V$ corresponds to a point in the plane, and an edge $\{p, q\} \in E$ is represented by the line segment $\overline{pq}$. The *length* $|\overline{pq}|$ of an edge $\{p, q\}$ is given by the Euclidean distance between the points p and q. The length of a shortest path between two nodes $p, q \in V$ is denoted by $d(p, q)$.

Now, we formally define a *routing scheme* for G. Each node p of G is assigned a *label* $\ell(p) \in \{0, 1\}^*$ that identifies it in the network. Furthermore, we store with p a *routing table* $\rho(p) \in \{0, 1\}^*$. The routing scheme works as follows: the packet contains the label $\ell(q)$ of the target node q, and initially it is situated at the start node p. In each step of the routing algorithm, the packet resides at a current node $p' \in V$. It may consult the routing table $\rho(p')$ of p' and the label $\ell(q)$ of the target to determine the next node q' to which the packet is forwarded. The node q' must be a neighbor of p' in G. This is repeated until the packet reaches its destination q. The scheme is modeled by a *routing function* $f : \rho(V) \times \ell(V) \to V$.

In the literature, there are varying definitions for the notion of a routing scheme [15,24,32]. For example, we may sometimes store additional information in the *header* of a data packet (it travels with the packet and can store information from past vertices). Similarly, the routing function sometimes allows the use of an *intermediate* target label. This is helpful for recursive routing schemes. Here, however, we will not need any of these additional capabilities.

As mentioned, the routing scheme operates by repeatedly applying the routing function. More precisely, given a start node $p \in V$ and a target label $\ell(q)$, the scheme produces the sequence of nodes $p_0 = p$ and $p_i = f(\rho(p_{i-1}), \ell(q))$, for $i \geq 1$. Naturally, we want routing schemes for which every packet reaches its desired destination. More precisely, a routing scheme is *correct* if for any $p, q \in V$, there exists a finite $k = k(p, q) \geq 0$ such that $p_k = q$ (and $p_i \neq q$ for $0 \leq i < k$). We call $p_0, p_1, \ldots, p_k$ the *routing path* between p and q. The *routing distance* between p and q is defined as $d_\rho(p, q) = \sum_{i=1}^{k} |\overline{p_{i-1}p_i}|$.

The quality of the routing scheme is measured by several parameters: (i) the *label size* $L(n) = \max_{|V|=n} \max_{p \in V} |\ell(p)|$, (ii) the *table size* $T(n) = \max_{|V|=n} \max_{p \in V} |\rho(p)|$, (iii) the *stretch* $\zeta(n) = \max_{|V|=n} \max_{p \neq q \in V} d_\rho(p, q)/d(p, q)$, and (iv) the preprocessing time.

Let P be a polygonal domain with n vertices. The *boundary* ∂P of P consists of h pairwise disjoint simple closed polygonal chains: one *outer boundary* and $h - 1$ *hole boundaries*, or h *hole boundaries* with no outer boundary. All hole boundaries lie inside the outer boundary, and no hole boundary lies inside another hole boundary. In both cases, we say that P has h holes. The interior induced by a hole boundary and the exterior of the outer boundary are not contained in P. We denote the (open) *interior* of P by $\operatorname{int} P$, i.e., $\operatorname{int} P = P \setminus \partial P$. We make no general position assumption on P. Let n_i, $0 \leq i \leq h - 1$, be the number of vertices on the i-th boundary of P. For each boundary i, we number the vertices from 0 to $n_i - 1$, in clockwise order, if i is a hole boundary, or in counterclockwise order if i is the outer boundary. The kth vertex of the ith boundary is denoted by $p_{i,k}$.

Two points p and q in P can *see each other* in P if and only if $\overline{pq} \subset P$. In particular, note that the line segment $\overline{pq}$ may touch ∂P. The *visibility graph* of P, $\mathrm{VG}(P)$, has the same vertices as P and an edge between two vertices if and only if they see each other in P. We show the following main theorem:

▶ **Theorem 2.1.** *Let $\varepsilon > 0$, and let P be a polygonal domain with n vertices and h holes. There is a routing scheme for $\mathrm{VG}(P)$ with stretch $\zeta(n) = 1+\varepsilon$, label size $L(n) = \mathcal{O}(\log n)$ and routing table size $T(n) = \mathcal{O}((\varepsilon^{-1} + h) \log n)$. The preprocessing time is $\mathcal{O}(n^2 \log n + hn^2 + \varepsilon^{-1}hn)$. If P is a simple polygon, the preprocessing time can be improved to $\mathcal{O}(n^2 + \varepsilon^{-1}n)$.*

3 Cones in Polygonal Domains

Let P be a polygonal domain with n vertices and h holes. Furthermore, let $t > 2$ be a parameter, to be determined later. Following Yao [33], we subdivide the visibility polygon of each vertex in P into t *cones* with a small enough apex angle. This will allow us to achieve small stretch and compact routing tables.

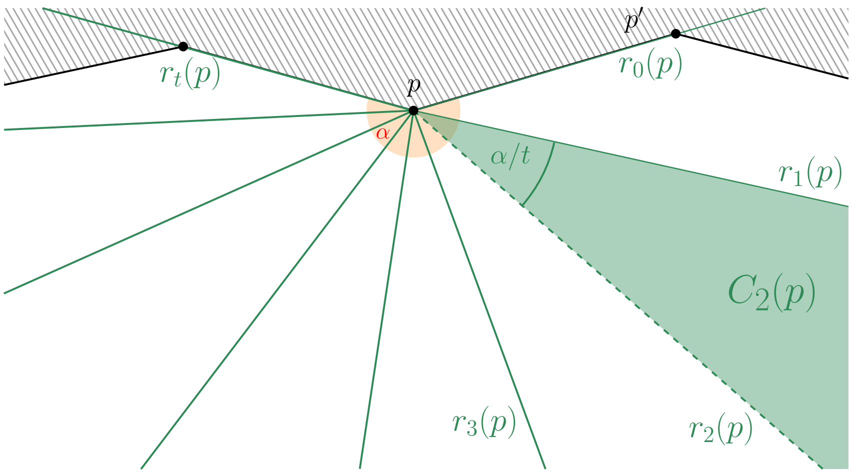

Figure 1 The cones and rays of a vertex p with apex angle α.

Let p be a vertex in P and p' the clockwise neighbor of p if p is on the outer boundary, or the counterclockwise neighbor of p if p lies on a hole boundary. We denote with $\mathbf{r}$ the *ray* from p through p'. To obtain our cones, we rotate $\mathbf{r}$ by certain angles. Let α be the inner angle at p. For $j = 0, \ldots, t$, we write $r_j(p)$ for the ray $\mathbf{r}$ rotated clockwise by angle $j \cdot \alpha/t$.

Now, for $j = 1, \ldots, t$, the cone $C_j(p)$ has apex p, boundary $r_{j-1}(p) \cup r_j(p)$, and opening angle α/t; see Figure 1. For technical reasons, we define $r_j(p)$ not to be part of $C_j(p)$, for $0 \leq j < t$, whereas we consider $r_t(p)$ to be part of $C_t(p)$. Furthermore, we write $\mathcal{C}(p) = \{C_j(p) \mid 1 \leq j \leq t\}$ for the set of all cones with apex p. Since the opening angle of each cone is $\alpha/t \leq 2\pi/t$ and since $t > 2$, each cone is convex.

▶ **Lemma 3.1.** *Let p be a vertex of P and let $\{p, q\}$ be an edge of* $\mathrm{VG}(P)$ *that lies in the cone $C_j(p)$. Furthermore, let s be a vertex of P that lies in $C_j(p)$, is visible from p, and that is closest to p. Then, $d(s, q) \leq |\overline{pq}| - (1 - 2\sin(\pi/t))\,|\overline{ps}|$.*

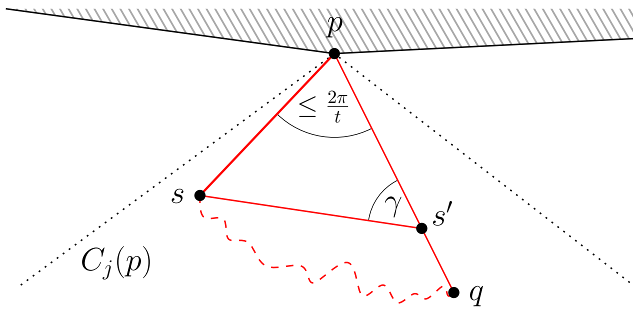

Figure 2 Illustration of Lemma 3.1. The points s and s' have the same distance to p. The dashed line represents the shortest path from s to q.

Proof. Let s' be the point on the line segment $\overline{pq}$ with $|\overline{ps'}| = |\overline{ps}|$; see Figure 2. Since p can see q, we have that p can see s' and s' can see q. Furthermore, s can see s', because p can see s and s' and we chose s to be closest to p, so the triangle $\Delta(p, s, s')$ cannot contain any vertices or (parts of) edges of P in its interior. Now, the triangle inequality yields $d(s, q) \leq |\overline{ss'}| + |\overline{s'q}|$. Let β be the inner angle at p between the line segments $\overline{ps}$ and $\overline{ps'}$. Since both segments lie in the cone $C_j(p)$, we get $\beta \leq 2\pi/t$. Thus, the angle between $\overline{s'p}$ and $\overline{s's}$ is $\gamma = \pi/2 - \beta/2$. Using the sine law and $\sin 2x = 2 \sin x \cos x$, we get

$$|\overline{ss'}| = |\overline{ps}| \cdot \frac{\sin\beta}{\sin\gamma} = |\overline{ps}| \cdot \frac{\sin\beta}{\sin((\pi/2) - (\beta/2))} = |\overline{ps}| \cdot \frac{2\sin(\beta/2)\cos(\beta/2)}{\cos(\beta/2)} \leq 2|\overline{ps}|\sin(\pi/t).$$

Furthermore, we have $|\overline{s'q}| = |\overline{pq}| - |\overline{ps'}| = |\overline{pq}| - |\overline{ps}|$. Thus, the triangle inequality gives

$$d(s, q) \leq 2|\overline{ps}|\sin(\pi/t) + |\overline{pq}| - |\overline{ps}| = |\overline{pq}| - (1 - 2\sin(\pi/t))\,|\overline{ps}|.$$

◀

4 The Routing Scheme

Let $\varepsilon > 0$, and let P be a polygonal domain with n vertices and h holes. We describe a routing scheme for $\mathrm{VG}(P)$ with stretch factor $1 + \varepsilon$. The idea is to compute for each vertex p the corresponding set of cones $\mathcal{C}(p)$ and to store a certain interval of indices for each cone $C_j(p)$ in the routing table of p. If an interval of a cone $C_j(p)$ contains the target vertex t, we proceed to the nearest neighbor of p in $C_j(p)$; see Figure 3. We will see that this results in a routing path with small stretch.

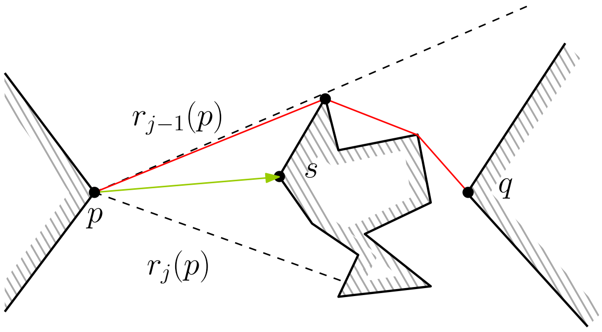

Figure 3 The idea of the routing scheme. The first edge on a shortest path from p to q (red) is contained in $C_j(p)$. The routing algorithm will route the packet from p to s (green), the closest vertex to p in C_j.

In the preprocessing phase, we first compute the label of each vertex $p_{i,k}$. The label of $p_{i,k}$ is the binary representation of i, concatenated with the binary representation of k, that is, $\ell(p_{i,k}) = (i, k)$. Thus, all labels are distinct binary strings of length $\lceil \log h \rceil + \lceil \log n \rceil$.

Let p be a vertex in P. Throughout this section, we will write $\mathcal{C}$ and C_j instead of $\mathcal{C}(p)$ and $C_j(p)$. The routing table of p is constructed as follows: first, we compute a shortest path tree T for p. For a vertex s of P, let T_s be the subtree of T with root s, and denote the set of all vertices on the i-th hole in T_s by $I_s(i)$. The following well-known observation lies at the heart of our routing scheme. For space reasons, we omit the proof from this extended abstract.

▶ **Observation 4.1.** *Let q_1 and q_2 be two vertices of P. Let π_1 be the shortest path in T from p to q_1, and π_2 the shortest path in T from p to q_2. Let l be the lowest common ancestor of q_1 and q_2 in T. Then, π_1 and π_2 do not cross or touch in a point x with $d(p, x) > d(p, l)$.*

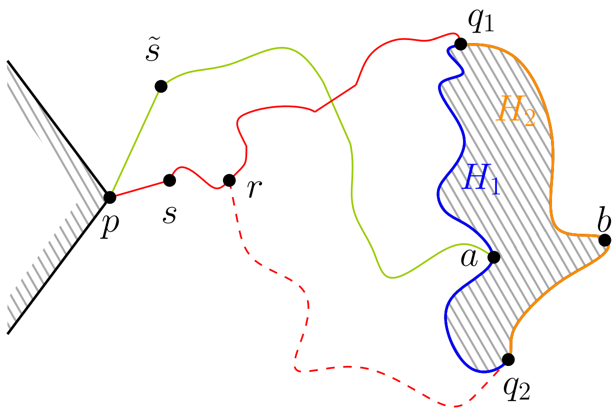

Figure 4 The shortest path from p to a (green) crosses the shortest path from p to q_1 (red). This gives a contradiction by Observation 4.1.

▶ **Lemma 4.2.** *Let $e = (p, s)$ be an edge in T. Then, the indices of the vertices in $I_s(i)$ form an interval. Furthermore, let $f = (p, s')$ be another edge in T, such that e and f are consecutive in the cyclic order around p in T. Then, the indices of the vertices in $I_s(i) \cup I_{s'}(i)$ are again an interval.*

Proof. For the first part of the lemma, suppose that the indices for $I_s(i)$ do not form an interval. Then, there are two vertices $q_1, q_2 \in I_s(i)$ such that if we consider the two polygonal chains H_1 and H_2 with endpoints q_1 and q_2 that constitute the boundary of hole i, there are two vertices $a, b \notin I_s(i)$ with $a \in H_1$ and $b \in H_2$ (see Figure 4). Let π_1 and π_2 be the shortest paths in T from s to q_1 and from s to q_2. Let r be the last common vertex of π_1 and π_2, and suppose without loss of generality that H_1, the subpath of π_1 from r to q_1, and the subpath of π_2 from r to q_2 bound a region inside P. Then, there has to be a child $\tilde{s}$ of p in T such that $a \in I_{\tilde{s}}(i)$ and such that the shortest path from $\tilde{s}$ to a intersects $\pi_1 \cup \pi_2$. Since p is the lowest common ancestor of a and q_1 and of a and q_2, this contradicts Observation 4.1.

The proof for the second part of the lemma is almost identical. We assume for the sake of contradiction that the indices in $I_s(i) \cup I_{s'}(i)$ do not form an interval, and we find vertices $q_1, q_2 \in I_s(i) \cup I_{s'}(i)$ such that if we split the boundary of hole i into two chains H_1 and H_2 between q_1 and q_2, there are two vertices $a, b \notin I_s(i) \cup I_{s'}(i)$ with $a \in H_1$ and $b \in H_2$. Again, let π_1 be the shortest path in T from s to q_1 and π_2 the shortest path in T from s to q_2, and consider the least common ancestor r of q_1 and q_2 in T. Without loss of generality, we assume that the region R bounded by H_1, the subpath of π_1 from r to q_1, and the subpath of π_2 from r to q_2 lies inside P. Now, the lowest common ancestor r may be p, but since s and s' are consecutive in the cyclic order around p, the other children of p are either all inside or all outside R. In either case, we can derive a contradiction to Observation 4.1 by noting that either the shortest path from s to a or the shortest path from s to b has to cross $\pi_1 \cup \pi_2$. ◀

Lemma 4.2 indicates how to construct the routing table $\rho(p)$ for p. We set

$$t = \pi / \arcsin\left(\frac{1}{2\,(1 + \varepsilon^{-1})}\right), \tag{1}$$

and we construct a set $\mathcal{C}$ of cones for p as in Section 3. Let $C_j \in \mathcal{C}$ be a cone, and let Π_i be a hole boundary or the outer boundary. We define $C_j \sqcap \Pi_i$ as the set of all vertices q on Π_i for which the first edge of the shortest shortest path from p to q lies in C_j. By Lemma 4.2, the indices of the vertices in $C_j \sqcap \Pi_i$ form a (possibly empty) cyclic interval $[k_1, k_2]$. If $C_j \sqcap \Pi_i = \emptyset$, we do nothing. Otherwise, if $C_j \sqcap \Pi_i \neq \emptyset$, there is a vertex $r \in C_j$ closest to p, and we add the entry (i, k_1, k_2, r) to $\rho(p)$. This entry needs $\lceil \log h \rceil + 3 \cdot \lceil \log n \rceil$ bits.

Now, the routing function $f : \rho(V) \times \ell(V) \to V$ is quite simple. Given a current vertex p and a target label $\ell(t) = (i, k)$, we search the routing table $\rho(p)$ for an entry (i, k_1, k_2, r) with $k \in [k_1, k_2]$. By construction, this entry is unique. We then forward the packet from p to the neighbor r (see Figure 3).

5 Analysis

We analyze the stretch factor of our routing scheme and give upper bounds on the size of the routing tables and the preprocessing time. Let $\varepsilon > 0$ be fixed, and let $1 + \varepsilon$ be the desired stretch factor. We set t as in (1). First, we bound t in terms of ε. This immediately gives that $|\mathcal{C}(p)| \in \mathcal{O}(\varepsilon^{-1})$, for every vertex p.

▶ **Lemma 5.1.** *We have* $t \leq 2\pi\left(1 + \varepsilon^{-1}\right)$.

Proof. For $x \in (0, 1/2]$, we have $\sin x \leq x$, so for $z \in [2, \infty)$, we get that $\sin(1/z) \leq 1/z$. Applying $\arcsin(\cdot)$ on both sides, this gives $1/z \leq \arcsin(1/z) \Leftrightarrow 1/\arcsin(1/z) \leq z$. We set $z = 2(1 + \varepsilon^{-1})$ and multiply by π to derive the desired inequality. ◀

5.1 The Routing Table

Let p be a vertex of P. We again write $\mathcal{C}$ for $\mathcal{C}(p)$ and C_j instead of $C_j(p)$. To bound the size of $\rho(p)$, we need some properties of holes with respect to cones. For $i = 0, \ldots, h-1$, we write $m(i)$ for the number of cones $C_j \in \mathcal{C}$ with $\mathcal{C}_j \sqcap \Pi_i \neq \emptyset$. Then, $\rho(p)$ contains at most $|\rho(p)| \leq \mathcal{O}\left(\sum_{i=0}^{h-1} m(i) \log n\right)$ bits. We say that Π_i is *stretched for the cone* C_j if there are indices $0 \leq j_1 < j < j_2 < t$ such that $C_{j_1} \sqcap \Pi_i$, $C_j \sqcap \Pi_i$ and $C_{j_2} \sqcap \Pi_i$ are non-empty. If Π_i is not stretched for any cone of p, then $m(i) \leq 2$. We prove the following lemma:

▶ **Lemma 5.2.** *For every* $C_j \in \mathcal{C}$*, there is at most one boundary that is stretched for* C_j.

Proof. Let Π_i be a hole boundary that is stretched for C_j. There are indices $j_1 < j < j_2$ and vertices $q \in C_{j_1} \sqcap \Pi_i$, $r \in C_j \sqcap \Pi_i$, and $s \in C_{j_2} \sqcap \Pi_i$. We subdivide P into three regions Q, R and S: the boundary of Q is given by the shortest path from p to r, the shortest path from p to q, and the part of Π_i from r to q not containing s. Similarly, the region R is bounded by the shortest path from p to r, the shortest path from p to s and the part of Π_i between r and s that does not contain q. Finally, S is the closure of $P \setminus (Q \cup R)$. The interiors of Q, R, and S are pairwise disjoint; see Figure 5.

Suppose there is another boundary Π that is stretched for C_j. Then, Π must lie entirely in either Q, R, or S. We discuss the first case, the other two are symmetric. Since Π is stretched for C_j, there is an index $j' > j$ and a vertex $t \in C_{j'} \sqcap \Pi$. Consider the shortest path π from p to t. Since $j' > j$, the first edge of π lies in R or S, and π has to cross or touch the shortest path from p to q or from q to r. Furthermore, by definition, we have $C_j \cap C_{j'} = \{p\}$ and $C_{j_1} \cap C_{j'} = \{p\}$. Therefore, p is the lowest common ancestor of all three shortest paths, and Observation 4.1 leads to a contradiction. ◀

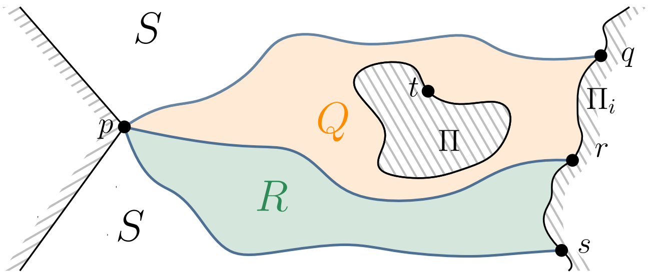

Figure 5 The shortest paths from p to q, r, s (blue). The hole Π contains t and lies in Q.

For $i = 0, \ldots, h-1$, let $s(i)$ be the number of cones in $\mathcal{C}$ for which Π_i is stretched. By Lemma 5.2, we get $\sum_{i=0}^{h-1} s(i) \leq |\mathcal{C}(p)| \in \mathcal{O}(\varepsilon^{-1})$. Since $m(i) \leq s(i) + 2$, we conclude

$$\begin{aligned} |\rho(p)| &\in \mathcal{O}\left(\sum_{i=0}^{h-1} m(i) \log n\right) = \mathcal{O}\left(\sum_{i=0}^{h-1} (s(i)+2) \log n\right) \\ &= \mathcal{O}\left((|\mathcal{C}(p)| + 2h) \log n\right) = \mathcal{O}\left((\varepsilon^{-1} + h) \log n\right). \end{aligned}$$

5.2 The Stretch Factor

Next, we bound the stretch factor. First, we prove that the distance to the target decreases after the first step. This will then give the bound on the overall stretch.

▶ **Lemma 5.3.** *Let p and q be two vertices in P. Let s be the next vertex computed by the routing scheme for a data packet from p to q. Then, $d(s, q) \leq d(p, q) - |\overline{ps}|/(1 + \varepsilon)$.*

Proof. By construction of $\rho(p)$, we know that the next vertex q' on the shortest path from p to q lies in the same cone as s. Hence, by the triangle inequality and Lemma 3.1, we obtain

$$\begin{aligned} d(s, q) &\leq d(s, q') + d(q', q) \leq |\overline{pq'}| - \left(1 - 2\sin\frac{\pi}{t}\right)|\overline{ps}| + d(q', q) \\ &= d(p, q) - \left(1 - 2\sin\frac{\pi}{t}\right)|\overline{ps}| = d(p, q) - \left(1 - \frac{1}{1+\varepsilon^{-1}}\right)|\overline{ps}| \qquad \text{(definition of } t\text{)} \\ &= d(p, q) - |\overline{ps}|/(1+\varepsilon). \end{aligned}$$

◀

Lemma 5.3 immediately implies the correctness of the routing scheme: since the distance to the target q decreases strictly in each step and since there is a finite number of vertices, there is a $k = k(p, q) \leq n$ such that after k steps, the packet arrives at q. Using this, we can now bound the stretch factor of the routing scheme.

▶ **Lemma 5.4.** *Let p and q be two vertices of P. Then, $d_\rho(p, q) \leq (1 + \varepsilon)d(p, q)$.*

Proof. Let $\pi = p_0 p_1 \ldots p_k$ be the routing path from $p = p_0$ to $q = p_k$. By Lemma 5.3, we have $d(p_{i+1}, q) \leq d(p_i, q) - |\overline{p_i p_{i+1}}|/(1 + \varepsilon)$. Thus,

$$\begin{aligned} d_\rho(p, q) &= \sum_{i=0}^{k-1} |\overline{p_i p_{i+1}}| \leq (1+\varepsilon) \sum_{i=0}^{k-1} \left(d(p_i, q) - d(p_{i+1}, q)\right) \\ &= (1+\varepsilon)\left(d(p_0, q) - d(p_k, q)\right) = (1+\varepsilon) d(p, q). \end{aligned}$$

◀

5.3 The Preprocessing Time

Finally, we discuss the details of the preprocessing algorithm and its time complexity.

▶ **Lemma 5.5.** *The preprocessing time for our routing scheme is* $\mathcal{O}(n^2 \log n + hn^2 + \varepsilon^{-1}hn)$.

Proof. Let p be a vertex of P. We compute the shortest path tree T for p. Using the algorithm of Hershberger and Suri [13], this can be done in time $\mathcal{O}(n \log n)$. Now, we perform a post-order traversal of T to compute the intervals for each child of p. Given a node q, the post-order traversal provides at most h different intervals. For each hole, we compute the union of the intervals among the children. Lemma 4.2 shows that the union of these intervals is again an interval, and it can be found in time $\mathcal{O}(h\,\mathrm{outdeg}(q))$, where $\mathrm{outdeg}(q)$ is the number of q's children in T. In total, the post-order traversal needs $\mathcal{O}(hn)$ time.

Let $q_1, \dots, q_k$ be the children of p, and let $\alpha_1, \dots, \alpha_k$ be the angles between the ray $r_0(p)$ and the edges (p, q_i), $i = 1, \dots, k$. By construction, the q_i are sorted by increasing angle α_i. Into this sorted sequence, we insert the rays $r_j(p)$, and we call the resulting sequence L. By Lemma 5.1, the sequence L has $\mathcal{O}(\varepsilon^{-1} + \mathrm{outdeg}(p))$ elements. We scan through L, and between each two consecutive rays $r_{j-1}(p)$ and $r_j(p)$, we join all the corresponding intervals for each hole. Again by Lemma 4.2, this gives a set of intervals. Finally, we compute the vertex closest to p in each cone, and we store the appropriate entries in the routing table $\rho(p)$. This last step takes time $\mathcal{O}(h(\varepsilon^{-1} + \mathrm{outdeg}(p))) = \mathcal{O}(h\varepsilon^{-1} + hn)$. Thus, the preprocessing time for p is $\mathcal{O}(n \log n + hn + h\varepsilon^{-1})$, for a total of $\mathcal{O}(n^2 \log n + hn^2 + \varepsilon^{-1}hn)$. ◀

Combining the last two lemmas with Section 4, we get the following theorem.

▶ **Theorem 5.6.** *Let P be a polygonal domain with n vertices and h holes. For any $\varepsilon > 0$ we can construct a routing scheme for $\mathrm{VG}(P)$ with labels of $\mathcal{O}(\log n)$ bits and routing tables of $\mathcal{O}((\varepsilon^{-1} + h) \log n)$ bits. For any two sites $p, q \in P$, the scheme produces a routing path with stretch factor at most $1 + \varepsilon$. The preprocessing time is $\mathcal{O}(n^2 \log n + hn^2 + \varepsilon^{-1}hn)$.*

6 Improvement for Simple Polygons

We show how to improve the preprocessing time for polygons without holes. Let P be a simple polygon with n vertices, and let $1+\varepsilon$, $\varepsilon > 0$, be the stretch factor. The previous section computes a shortest path tree for each vertex, which leads to $\mathcal{O}(n^2 \log n)$ preprocessing time. In simple polygons, we can use a different technique to avoid this large overhead in the preprocessing phase. The routing function, the vertex labels, and the structure of the routing tables remain unchanged.

Let p be a vertex of P. We compute the visibility polygon $\mathrm{vis}(p)$ for p. This gives a sequence V of points $v_0, v_1, \dots v_m$ with $p = v_0 = v_m$. Some points of V may not be vertices of P. We assume that V is sorted clockwise. Then, the sequence $\alpha_1, \alpha_2, \dots, \alpha_{m-1}$ of the angles α_j between the ray $r_0(p)$ and the edges $\{p, v_j\}$, $j = 1, \dots, m-1$, is increasing. For $j = 1, \dots, t-1$, let w_j be the intersection point of $r_j(p)$ and $\mathrm{vis}(p)$ that is closest to p. The sequence of edges e_j of P that contain the points w_j can be found in $O(n)$ time by traversing the sorted sequence V; see Figure 6.

Next, let $C_j \in \mathcal{C}$ be a cone. Recall that C_j is bounded by the rays $r_{j-1}(p)$ and $r_j(p)$. The vertices related to C_j are determined as follows: starting from w_{j-1}, we walk along the boundary of P, until we meet w_j. During the walk, we collect all the visited vertices. This set forms a (possibly empty) interval $I(j)$. We let s be the vertex in $I(j)$ with the smallest distance to p. As before, we add the endpoints of $I(j)$ together with s to $\rho(p)$. This

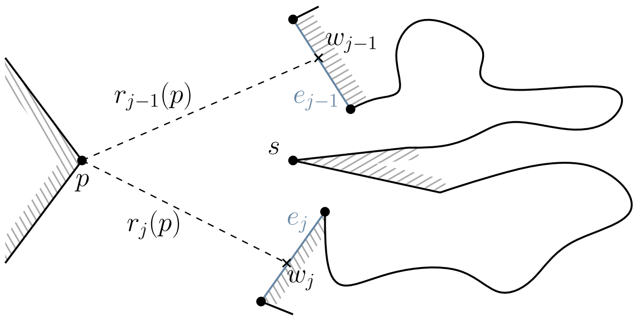

Figure 6 The boundaries of C_j hit ∂P in the points w_{j-1} and w_j. The vertex s is the vertex in C_j with smallest distance to p.

needs $3 \cdot \lceil \log n \rceil$ bits. By Lemma 5.1, the routing table $\rho(p)$ has $\mathcal{O}(\varepsilon^{-1} \log n)$ bits, as in the previous section. To show correctness, we need the following lemma.

▶ **Lemma 6.1.** *Let p and q be two vertices of P, and let (p, q') bet the first edge on the shortest path from p to q. If $q \in I(j)$, then $q' \in C_j$.*

Proof. Suppose that $q' \notin C_j$. Since $q \in I(j)$, the shortest path π from p to q has to meet $\overline{pw_{i-1}}$ or $\overline{pw_i}$ at least twice. The first intersection is p itself. Let $a \neq p$ be the second intersection, and π' the subpath of π from p to a. By the triangle inequality $|\overline{pa}|$ is strictly smaller than the length of π'; see Figure 7. This contradicts the fact that π is a shortest path from p to q. ◀

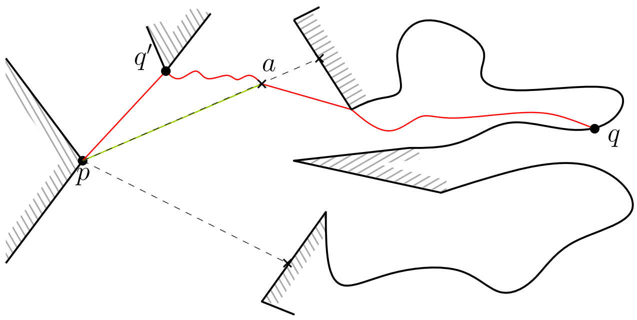

Figure 7 The red curve is the "shortest" path from p to q with q' as first step, whereas the green dashed line represents a shortcut from p to a.

Thus, we obtain our main theorem for simple polygons.

▶ **Theorem 6.2.** *Let P be a simple polygon with n vertices. For any $\varepsilon > 0$, we can construct a routing scheme for $\mathrm{VG}(P)$ with labels of $\lceil \log n \rceil$ bits and routing tables of $\mathcal{O}(\varepsilon^{-1} \log n)$ bits. For any two vertices $p, q \in P$, the scheme produces a routing path with stretch $1 + \varepsilon$. The preprocessing time is $\mathcal{O}(n^2 + \varepsilon^{-1} n)$.*

Proof. Let p be a vertex of P. First, we compute the visibility polygon of the vertex p. This needs time $\mathcal{O}(n)$ [14, 18]. Let V be the vertices of $\mathrm{vis}(p)$, sorted by increasing angle. Using

V, we can find in time $\mathcal{O}(n + \varepsilon^{-1})$ all the intersection points w_j and the edges e_j of P that contain them. Finally, let C_j be a cone. We can find in constant time the endpoints of $I(j)$ and in $\mathcal{O}(|I(j)|)$ time the vertex s in $I(j)$ with the smallest distance to p. This step costs $\mathcal{O}(n + \varepsilon^{-1})$ time in total over all cones. The total running time is $\mathcal{O}(n^2 + \varepsilon^{-1}n)$. ◀

7 Conclusion

We gave an efficient routing scheme for the visibility graph of a polygonal domain. Our scheme produces routing paths whose length can be made arbitrarily close to the optimum.

Several open questions remain. First of all, we would like to obtain an efficient routing scheme for the *hop-distance* in polygonal domains P, where each edge of $\mathrm{VG}(P)$ has unit weight. For our routing scheme, we can easily construct examples where the stretch is $\Omega(n)$; see Figure 8. Moreover, it would be interesting to improve the preprocessing time or the size of the routing tables, perhaps using a recursive strategy.

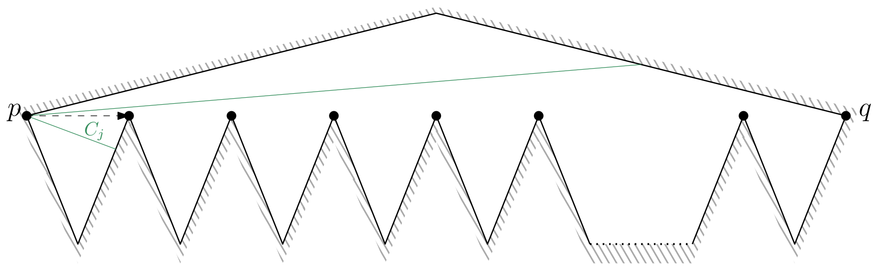

Figure 8 In this polygon, p and q can see each other, so their hop-distance is 1. Our routing scheme routes from one spire to the next, giving stretch $\Theta(n)$.

A final open question concerns routing schemes in general: what is the time needed by a data packet to travel through the graph? In particular, it would be interesting to see how much time a data packet needs at one single vertex until it knows the vertex where it is forwarded. It would be a sightly different, but important measure for routing schemes.

References

1. Ittai Abraham and Cyril Gavoille. On approximate distance labels and routing schemes with affine stretch. In *Proc. 25th DISC*, pages 404–415, 2011.
2. Takao Asano, Tetsuo Asano, Leonidas Guibas, John Hershberger, and Hiroshi Imai. Visibility of disjoint polygons. *Algorithmica*, 1(1–4):49–63, 1986.
3. Baruch Awerbuch, Amotz Bar-Noy, Nathan Linial, and David Peleg. Improved routing strategies with succinct tables. *JALG*, 11(3):307–341, 1990.
4. Reuven Bar-Yehuda and Bernard Chazelle. Triangulating disjoint Jordan chains. *IJCGA*, 4(04):475–481, 1994.
5. Prosenjit Bose, Rolf Fagerberg, André van Renssen, and Sander Verdonschot. Optimal local routing on Delaunay triangulations defined by empty equilateral triangles. *SICOMP*, 44(6):1626 – 1649, 2015.
6. Prosenjit Bose, Rolf Fagerberg, André van Renssen, and Sander Verdonschot. Competitive local routing with constraints. *JoCG*, 8(1):125–152, 2017.
7. Shiri Chechik. Compact routing schemes with improved stretch. In *Proc. PODC*, pages 33–41, 2013.
8. Lenore J Cowen. Compact routing with minimum stretch. *JALG*, 38(1):170–183, 2001.

9 Tamar Eilam, Cyril Gavoille, and David Peleg. Compact routing schemes with low stretch factor. *JALG*, 46(2):97–114, 2003.

10 Pierre Fraigniaud and Cyril Gavoille. Routing in trees. In *Proc. 28th ICALP*, pages 757–772, 2001.

11 Silvia Giordano and Ivan Stojmenovic. Position based routing algorithms for ad hoc networks: A taxonomy. In *Ad hoc wireless networking*, pages 103–136. Springer-Verlag, 2004.

12 Leonidas Guibas, John Hershberger, Daniel Leven, Micha Sharir, and Robert E Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1-4):209–233, 1987.

13 John Hershberger and Subhash Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SICOMP*, 28(6):2215–2256, 1999.

14 Barry Joe and Richard B Simpson. Corrections to Lee's visibility polygon algorithm. *BIT Numerical Mathematics*, 27(4):458–473, 1987.

15 Haim Kaplan, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Routing in unit disk graphs. In *Proc. 12th LATIN*, pages 536–548, 2016.

16 Sanjiv Kapoor and SN Maheshwari. Efficient algorithms for Euclidean shortest path and visibility problems with polygonal obstacles. In *Proc. 4th SoCG*, pages 172–182, 1988.

17 Sanjiv Kapoor, SN Maheshwari, and Joseph SB Mitchell. An efficient algorithm for Euclidean shortest paths among polygonal obstacles in the plane. *DCG*, 18(4):377–383, 1997.

18 Der-Tsai Lee. Visibility of a simple polygon. *CGVIP*, 22(2):207–221, 1983.

19 Joseph SB Mitchell. A new algorithm for shortest paths among obstacles in the plane. *AMAI*, 3(1):83–105, 1991.

20 Joseph SB Mitchell. Shortest paths among obstacles in the plane. *IJCGA*, 6(03):309–332, 1996.

21 Mark H Overmars and Emo Welzl. New methods for computing visibility graphs. In *Proc. 4th SoCG*, pages 164–171, 1988.

22 David Peleg and Eli Upfal. A trade-off between space and efficiency for routing tables. *J. ACM*, 36(3):510–530, 1989.

23 Liam Roditty and Roei Tov. New routing techniques and their applications. In *Proc. PODC*, pages 23–32, 2015.

24 Liam Roditty and Roei Tov. Close to linear space routing schemes. *Distributed Computing*, 29(1):65–74, 2016.

25 Nicola Santoro and Ramez Khatib. Labelling and implicit routing in networks. *The Computer Journal*, 28(1):5–8, 1985.

26 Micha Sharir and Amir Schorr. On shortest paths in polyhedral spaces. *SICOMP*, 15(1):193–215, 1986.

27 James A Storer and John H Reif. Shortest paths in the plane with polygonal obstacles. *J. ACM*, 41(5):982–1012, 1994.

28 Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, 2004.

29 Mikkel Thorup and Uri Zwick. Compact routing schemes. In *Proc. 13th SPAA*, pages 1–10, 2001.

30 Mikkel Thorup and Uri Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005.

31 Emo Welzl. Constructing the visibility graph for n-line segments in $\mathcal{O}(n^2)$ time. *IPL*, 20(4):167–171, 1985.

32 Chenyu Yan, Yang Xiang, and Feodor F Dragan. Compact and low delay routing labeling scheme for unit disk graphs. *CGTA*, 45(7):305–325, 2012.

33 Andrew Chi-Chih Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SICOMP*, 11(4):721–736, 1982.