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 of P. 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 whose length exceeds the shortest path by a factor of at most $1 + \varepsilon$. The labels have $O(\log n)$ bits, and the routing tables are of size $O((\varepsilon^{-1} + h) \log n)$. The preprocessing time is $O(n^2 \log n)$. It can be improved to $O(n^2)$ for simple polygons.

1 Introduction

Routing is a crucial problem in distributed graph algorithms [23, 34]. 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 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 factor*. Third, it should be *compact*: the total space requirement should be as small as possible.

Here 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 q, we can send the packet for one more step along the shortest path from v to q. Then, the routing scheme will have perfect efficiency, sending each packet along a shortest path. However, this method requires that each node stores its entire shortest path tree, making it not compact. Thus, the challenge lies in finding the right balance between the conflicting goals of compactness and efficiency.

*A preliminary version appeared as B. Banyassady, M-K. Chiu, M. Korman, W. Mulzer, A. v. Renssen, M. Roeloffzen, P. Seiferth, Y. Stein, B. Vogtenhuber, and M. Willert. *Routing in Polygonal Domains.* Proc. 28th ISAAC, pp. 10:1–10.13. BB was supported in part by DFG project MU/3501-2. MC, AvR and MR were supported by JST ERATO Grant Number JPMJER1201, Japan. MK was supported in part by KAKENHI Nos. 15H02665 and 17K12635, Japan. WM was supported in part by ERC StG 757609. 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.

Thorup and Zwick introduced the notion of a *distance oracle* [42]. 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 [36].

The problem of constructing a compact routing scheme for a general graph has been studied for a long time [1,3,16,18,21,35,36]. One of the most recent results, by Roditty and Tov, dates from 2016 [36]. 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 p to q on a path with length $O(k\Delta + m^{1/k})$, where Δ is the shortest path distance between p and q and $k > 2$ is any fixed integer. The routing tables use $mn^{O(1/\sqrt{\log n})}$ total space. In general graphs, any routing scheme with constant stretch factor needs to store $\Omega(n^c)$ bits per node, for some constant $c > 0$ [34]. 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 $O(\log n)$ bits at each node [22,37,41]. 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 [40]. Similar results are also available for unit disk graphs [26], and for metric spaces with bounded doubling dimension [29].

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 location of the source and the target vertex, the current vertex, and its neighbourhood, see for instance [9,10] and the references therein. The most notable difference between geometric routing and our setting is that in geometric routing, vertices are generally not allowed to store routing tables, so that routing decisions are based solely on the geometric information available at the current vertex (and possibly information stored in the message). We note that the location of the source vertex may or may not be needed, depending on the routing algorithm. For example, the routing algorithm for triangulations by Bose and Morin [13] uses the line segment between the source and the target for its routing decisions. A recent result by Bose *et al.* [10] is very close to our setting. They show that when vertices do not store any routing tables (i.e., each vertex stores only the edges that can be followed from it), no geometric routing scheme can achieve stretch factor $o(\sqrt{n})$. This lower bound applies regardless of the amount of information that may be stores in the message.

Here, we consider the class of visibility graphs of a polygonal domain. Let P be such a polygonal domain 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. We note that this definition implies that the visibility graph contains the shortest path between any two vertices of the polygonal domain. The problem of computing a shortest path between two vertices in a polygonal domain has been well-studied in computational geometry [2,4,24,25,27,28,30,31,33,38,39,43]. 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.

When we relax the requirement on the length of the path, we enter the domain of spanners: given a graph G, a subgraph H of G is a *k-spanner* of G if for all pairs of vertices p and q in G, $d_H(p, q) \leq k \cdot d_G(p, q)$, for $k \geq 1$. The spanning properties of various geometric graphs have been studied extensively in the literature (see [15,32] for a comprehensive overview). We briefly mention the results that are most closely related to the approach we will take here, namely Yao-graphs [45] and Θ-graphs [17]. Intuitively, these graphs form geometric networks where each vertex connects to its nearest visible vertex in a certain number of different directions (a formal definition is given in Section 3). Both types of graphs are spanners, where the stretch factor depends on the number of cones used [5–8,14,19,20]. These graphs have also been considered for geometric routing purposes. For example, Bose *et al.* [9] gave an optimal geometric routing algorithm for the half-Θ_6-graph (the Θ-graph with six cones where edges are added in every other cone). When considering obstacles, Θ-graphs have recently been used to route on (subgraphs of) the visibility graph [10–12], though these algorithms do not provide a bound on the total length of the routing path, only on the number of edges followed by the routing scheme. However, as mentioned earlier, these geometric routing schemes cannot achieve a stretch factor of $o(\sqrt{n})$, as they are not allowed to store routing tables at the vertices.

We introduce a routing scheme that, for any $\varepsilon > 0$, needs $O((1/\varepsilon + h) \log n)$ bits in each routing table, and for any two vertices p and q, it produces a routing path that is within a factor of $1 + \varepsilon$ of optimal. This shows that by allowing a routing table at each vertex, we can do much better than in traditional geometric routing, achieving a stretch factor that is arbitrarily close to 1.

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 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 $d(p, q)$.

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 [26, 36, 44]. 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:

1. the *label size* $\max_{p \in V} |\ell(p)|$,
2. the *table size* $\max_{p \in V} |\rho(p)|$,
3. the *stretch factor* $\max_{p \neq q \in V} d_\rho(p, q)/d(p, q)$, and
4. 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 any 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 assume that P is in general position: no three vertices of P lie on a common line, and for each pair of vertices in P, the shortest path between them is unique. 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 $\overline{pq} \subset P$. By our general position assumption, $\overline{pq}$ touches ∂P only if $\overline{pq}$ is itself an edge of 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 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 $O(\log n)$ bits and routing tables of $O((1/\varepsilon + h)\log n)$ bits per vertex. 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 $O(n^2 \log n)$. If P is a simple polygon, the preprocessing time reduces to $O(n^2)$.*

3 Cones in Polygonal Domains

Let P be a polygonal domain with n vertices and h holes. Furthermore, let $t \geq 3$ be an integer parameter, to be determined later. Following Yao [45] and Clarkson [17], we subdivide the visibility polygon of each vertex in P into t *cones* with a small enough apex angle. This will allow us to construct compact routing tables that support a routing algorithm with small stretch factor.

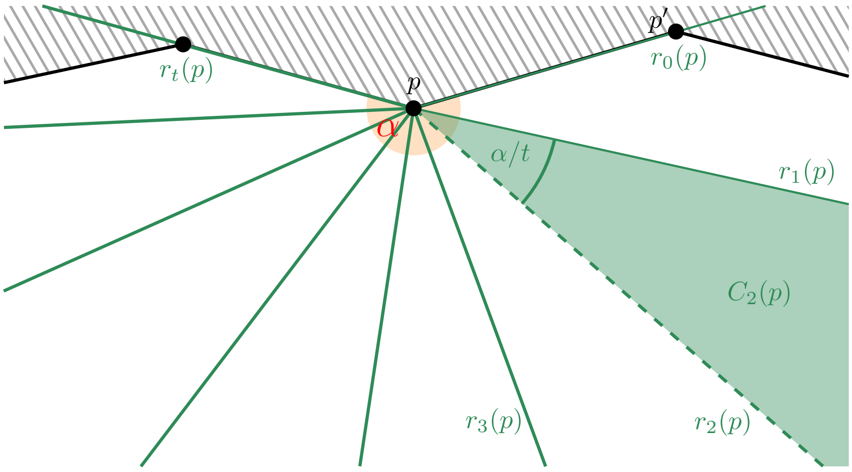

Figure 1: The cones and rays of a vertex p with apex angle α.

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 $\mathbf{r}(p)$ the *ray* from p through p'. To obtain our cones, we rotate $\mathbf{r}(p)$ by certain angles. Let α be the inner angle at p. For $j = 0, \dots, t$, we write $r_j(p)$ for the ray $\mathbf{r}(p)$ rotated clockwise by angle $j \cdot \alpha/t$.

Now, for $j = 1, \dots, t$, 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 $1 \leq j < t$, 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 $t \geq 3$, each cone is convex.

The following proof is similar to the one given by Clarkson [17] and Narasimhan and Smid [32], though the former shows only that the construction leads to an $O(1/\varepsilon)$-spanner instead of showing a more precise bound in terms of the number of cones.

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

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

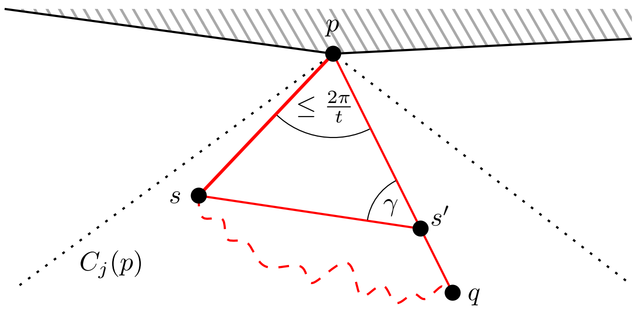

Figure 2: Illustration of Lemma 3.1. The points s and s' have the same distance to p. The dashed line represents the shortest path from s to q.

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

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

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

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

as claimed. ☐

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 factor.

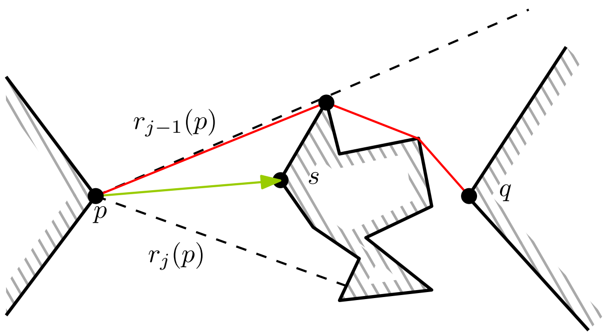

Figure 3: The idea of the routing scheme. The first edge on a shortest path from p to q (red) is contained in $C_j(p)$. The routing algorithm will route the packet from p to s (green), the closest vertex to p in C_j.

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. 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 completeness, we include a proof.

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 $d(p, x) > d(p, l)$.*

Proof. Suppose first that π_1 touches π_2 in a point x with $d(p, x) > d(p, l)$. The edges of T are line segments, so this can only happen if x is a vertex. But then T would contain a cycle, which is impossible.

Next, suppose that π_1 and π_2 cross in a point x with $d(p, x) > d(p, l)$. Suppose further that x lies on the edge $e_1 = (s_1, t_1)$ of π_1 and the edge $e_2 = (s_2, t_2)$ of π_2; see Figure 4. Without loss of generality, we have $d(l, s_1) + |\overline{s_1 x}| \leq d(l, s_2) + |\overline{s_2 x}|$. Since $x \in \operatorname{int} P$, there is a $\delta > 0$ such that the disk D with center x and radius δ is contained in P. Now consider the intersection y_1 of ∂D with $\overline{s_1 x}$ and the intersection y_2 of ∂D with $\overline{x t_2}$. We have $\overline{y_1 y_2} \subset D \subset P$, and the triangle inequality yields $|\overline{y_1 x}| + |\overline{x y_2}| > |\overline{y_1 y_2}|$. Hence, the path $s_1 y_1 y_2 t_2$ is a shortcut from l to t_2, a contradiction to π_2 being a shortest path. ☐

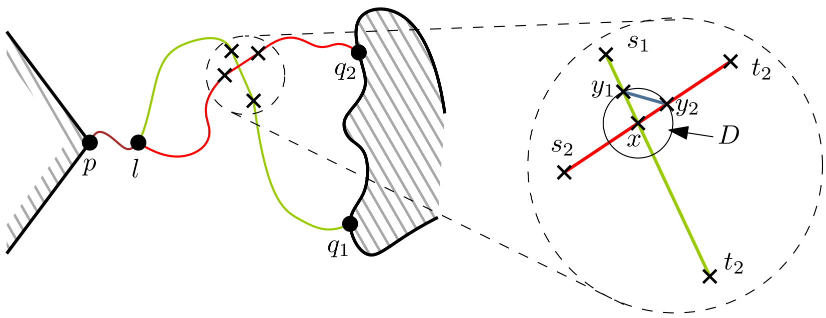

Figure 4: Two shortest paths that originate in p cannot cross.

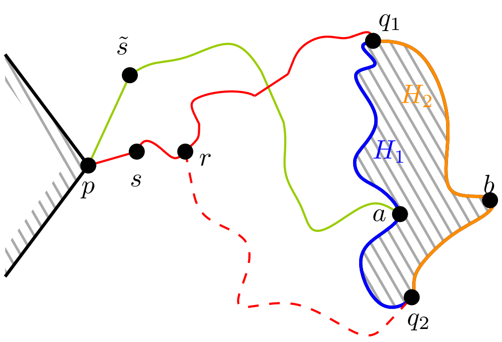

Figure 5: The shortest path from p to a (green) crosses the shortest path from p to q_1 (red). This gives a contradiction by Observation 4.1.

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 edges in T around p.*[1] *Then, the indices of the vertices in $I_s(i) \cup I_{s'}(i)$ are again an interval.*

[1] By this, we mean that there is no other edge of T incident to p in the cone that is spanned by e and f and that extends into the interior of P.

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 5). 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 let $\tilde{\pi}_1$ be the subpath of π_1 from r to q_1 and $\tilde{\pi}_2$ the subpath of π_2 from r to q_2. Consider the set $\mathcal{D}$ of (open) connected components of $P \setminus (\tilde{\pi}_1 \cup \tilde{\pi}_2)$. Any vertex of P that is on the boundary of two different components of $\mathcal{D}$ must lie on $\tilde{\pi}_1 \cup \tilde{\pi}_2$. Hence, p, a, and, b each lie on the boundary of exactly one component in $\mathcal{D}$, and the components D_a and D_b with a and b on the boundary are distinct. Suppose without loss of generality that $p \notin \partial D_a$. 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 crosses $\pi_1 \cup \pi_2$. 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 is very similar. 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 $b \in H_2$. Furthermore, we may assume that $a \neq p$ and $b \neq p$, because otherwise q_1 and q_2 would be the two vertices of P that share an edge with p, and thus q_1 and q_2 would be the only two children of p in T and $I_s(i) \cup I_{s'}(i)$ would be an interval. 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 lowest common ancestor r of q_1 and q_2 in T (now r might be p). Let $\tilde{\pi}_1$ be the subpath of π_1 from r to q_1 and $\tilde{\pi}_2$ the subpath of π_2 from r to q_2. Consider the set $\mathcal{D}$ of (open) connected components of $P \setminus (\tilde{\pi}_1 \cup \tilde{\pi}_2)$. As before, any vertex that lies on the boundaries of two distinct components of $\mathcal{D}$ must belong to $\tilde{\pi}_1 \cup \tilde{\pi}_2$, so a and b are on the boundaries of two uniquely defined distinct components in $\mathcal{D}$. We call these components D_a and D_b. Now, s and s' are consecutive around p, so at least one of D_a and D_b contains no other child of p in T on its boundary. Let it be D_a. Then, the shortest path from p to a must cross $\pi_1 \cup \pi_2$, contradicting Observation 4.1. ☐

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

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

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 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 $(i, k_1, k_2, \ell(r))$ to $\rho(p)$. This entry needs $2 \cdot \lceil \log h \rceil + 3 \cdot \lceil \log n \rceil$ bits.

Now, the routing function $f : \rho(V) \times \ell(V) \to V$ is quite simple. Given the routing table $\rho(p)$ for the current vertex p and a target label $\ell(q) = (i, k)$, indicating vertex k on hole i, we search $\rho(p)$ for an entry $(i, k_1, k_2, \ell(r))$ with $k \in [k_1, k_2]$. By construction, this entry is unique. We return r as the next destination for the packet (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 $|\mathcal{C}(p)| \in O(1/\varepsilon)$, for every vertex p.

Lemma 5.1. *We have $t \leq 2\pi\,(1 + 1/\varepsilon) + 1$.*

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 $z = 2(1 + 1/\varepsilon)$ 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 $i = 0, \dots, h-1$, 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 $|\rho(p)| \le O\left(\sum_{i=0}^{h-1} m(i) \log n\right)$ bits. We say that Π_i is *stretched for the cone* C_j if there are indices $0 \le j_1 < j < j_2 < t$ 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 $m(i) \le 2$. We prove the following lemma:

Lemma 5.2. *For every cone $C_j \in \mathcal{C}$, there is at most one boundary Π_i 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 6.

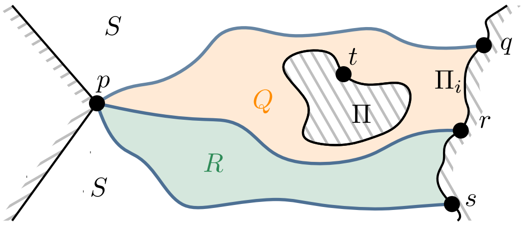

Figure 6: The shortest paths from p to q, r, s (blue). The hole Π contains t and lies in Q.

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 p 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. ☐

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

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

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 factor.

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, $d(s,q) \le d(p,q) - |\overline{ps}|/(1+\varepsilon)$.*

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

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

as desired. ☐

Lemma 5.3 immediately shows the correctness of the routing scheme: the distance to the target q decreases strictly in each step and there is a finite number of vertices, so there is a $k = k(p, q) \leq n$ so that after k steps, the packet reaches 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 $\pi = p_0 p_1 \dots p_k$ 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,

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

as claimed. ☐

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 $O(n^2 \log n + n/\varepsilon)$ for polygonal domains and $O(n^2 + n/\varepsilon)$ for simple polygons.*

Proof. Let p be a vertex of P. We compute the shortest path tree T for p. In polygonal domains, this takes $O(n \log n)$ time using the algorithm of Hershberger and Suri [25], and in simple polygons, this needs $O(n)$ time, using the algorithm of Guibas *et al.* [24]. We perform a circular sweep around p to find for each cone $C_j \in \mathcal{C}$ the set X_j of the children of p in T that lie in C_j. This requires $O(n + 1/\varepsilon)$ steps.

For each cone C_j, we find the child $r \in X_j$ that is closest to p. We traverse all subtrees of T that are rooted at some child in X_j, and we collect the set V_j of all their vertices. We group the vertices in V_j according to the hole boundaries they belong to. This takes $O(|V_j|)$ time, using the following *bucketing scheme*: once for the whole algorithm, we set up an array B of buckets with h entries, one for each hole boundary. Each bucket consists of a linked list, initially empty. This gives a one-time initialization cost of $O(h)$. When processing the vertices of V_j, we create a linked list N of *non-empty* buckets, also initially empty. For each $v \in V_j$, we add v into its corresponding bucket $B[i]$. If v is the first vertex in $B[i]$, we add i to N. This takes $O(|V_j|)$ time in total, and it leads to the desired grouping of V_j. Once we have processed V_j, we use N in order to reset all the buckets we used to empty, in another $O(|V_j|)$ steps.

Now, for each hole i, let $V_{j,i}$ be the set of all vertices on Π_i that lie in V_j. By Lemma 4.2, $V_{j,i}$ is a cyclic interval. To determine its endpoints, it suffices to identify one vertex on hole i that is not in $V_{j,i}$ (if it exists). After that, a simple scan over $V_{j,i}$ gives the desired interval endpoints in $O(|V_{j,i}|)$ additional time. To find this vertex in $O(|V_{j,i}|)$ time, we use prune and search: let $L = \{p_{i,k} \in V_{j,i} \mid k < \lceil n_i/2 \rceil\}$ and $R = V_{j,i} \setminus L$. We determine $|L|$ and $|R|$ by scanning $V_{j,i}$, and we distinguish three cases. First, if $|L| = \lceil n_i/2 \rceil$ and $|R| = \lfloor n_i/2 \rfloor$, all vertices of hole i lie in the V_j, and we are done. Second, if $|L| < \lceil n_i/2 \rceil$ and $|R| < \lfloor n_i/2 \rfloor$, then at least one of $p_{i,0}$, $p_{i,\lceil n_i/2 \rceil - 1}$, $p_{i,\lceil n_i/2 \rceil}$, and p_{i,n_i-1} is not in $V_{j,i}$. Another scan over $V_{j,i}$ reveals which one it is. In the third case, exactly one of the two sets L, R contains all possible vertices, whereas the other one does not. We recurse on the latter set. This set contains at most $|V_{j,i}|/2$ elements, so the overall running time for the recursion is $O(|V_{j,i}|)$.

It follows that we can handle a single cone C_j in time $O(|V_j|)$, so the total time for processing p is $O(n \log n + 1/\varepsilon)$ in polygonal domains and $O(n + 1/\varepsilon)$ in simple polygons. Since we repeat for each vertex of P, the claim follows. ☐

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

Theorem 2.1. *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 $O(\log n)$ bits and routing tables of $O((1/\varepsilon + h) \log n)$ bits per vertex. 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 $O(n^2 \log n)$. If P is a simple polygon, the preprocessing time reduces to $O(n^2)$.*

Proof. First, note that we may assume that $\varepsilon = \Omega(1/n)$, otherwise, the theorem follows trivially from storing a complete shortest path tree in each routing table. Thus, $1/\varepsilon = O(n)$, and by Lemma 5.5, the preprocessing time is $O(n^2 \log n)$ for polygonal domains, and $O(n^2)$ for simple polygons. The claim on the label size follows from the discussion at the beginning of Section 4, the size of the routing tables is given in Section 5.1, and the stretch factor is proved in Lemma 5.4. ☐

6 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. This scenario occurs for routing in a wireless network: here, the main overhead is caused by forwarding a packet at a base station, whereas the distance that the packet has to cross is negligible for the travel time. For our routing scheme, we can construct examples where the stretch factor is $\Omega(n)$; see Figure 7. Moreover, it would be interesting to improve the preprocessing time or the size of the routing tables, perhaps using a recursive strategy.

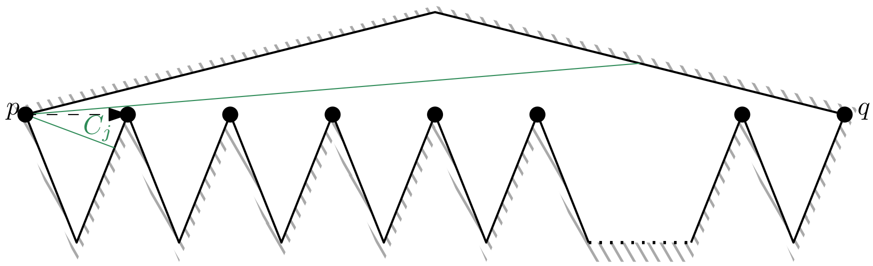

Figure 7: In this polygon, p and q can see each other, so their hop-distance is 1. Our routing scheme routes from one spire to the next, giving stretch factor $\Theta(n)$.

A final open question concerns routing schemes in general: how do we model the time needed by a data packet to travel through the graph, including the processing times at the vertices? In particular, it would be interesting to consider a model in which each vertex has a fixed *processing time* until it knows the next vertex for the current packet. This would lead to a sightly different, but important, measure for routing schemes.

References

[1] I. Abraham and C. Gavoille. On approximate distance labels and routing schemes with affine stretch. In *Proc. 25th Int. Symp. Dist. Comp. (DISC)*, pages 404–415, 2011.

[2] T. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai. Visibility of disjoint polygons. *Algorithmica*, 1(1–4):49–63, 1986.

[3] B. Awerbuch, A. Bar-Noy, N. Linial, and D. Peleg. Improved routing strategies with succinct tables. *J. Algorithms*, 11(3):307–341, 1990.

[4] R. Bar-Yehuda and B. Chazelle. Triangulating disjoint Jordan chains. *Internat. J. Comput. Geom. Appl.*, 4(04):475–481, 1994.

[5] L. Barba, P. Bose, M. Damian, R. Fagerberg, W. L. Keng, J. O'Rourke, A. van Renssen, P. Taslakian, S. Verdonschot, and G. Xia. New and improved spanning ratios for Yao graphs. *J. of Computational Geometry*, 6(2):19–53, 2015.

[6] L. Barba, P. Bose, J.-L. De Carufel, A. van Renssen, and S. Verdonschot. On the stretch factor of the theta-4 graph. In *Proc. 13th Int. Sympos. Algorithms and Data Structures (WADS)*, pages 109–120, 2013.

[7] P. Bose, J.-L. D. Carufel, P. Morin, A. van Renssen, and S. Verdonschot. Towards tight bounds on theta-graphs: More is not always better. *Theoret. Comput. Sci.*, 616:70–93, 2016.

[8] P. Bose, M. Damian, K. Douïeb, J. O'Rourke, B. Seamone, M. Smid, and S. Wuhrer. $\pi/2$-angle Yao graphs are spanners. *Internat. J. Comput. Geom. Appl.*, 22(1):61–82, 2012.

[9] P. Bose, R. Fagerberg, A. van Renssen, and S. Verdonschot. Optimal local routing on Delaunay triangulations defined by empty equilateral triangles. *SIAM J. Comput.*, 44(6):1626 – 1649, 2015.

[10] P. Bose, R. Fagerberg, A. van Renssen, and S. Verdonschot. Competitive local routing with constraints. *J. of Computational Geometry*, 8(1):125–152, 2017.

[11] P. Bose, M. Korman, A. van Renssen, and S. Verdonschot. Constrained routing between non-visible vertices. In *Proc. 23rd Annu. Int. Computing and Combinatorics Conf.*, pages 62–74, 2017.

[12] P. Bose, M. Korman, A. van Renssen, and S. Verdonschot. Routing on the visibility graph. In *Proc. 28th Annu. Internat. Sympos. Algorithms Comput. (ISAAC)*, pages 18:1–18:12, 2017.

[13] P. Bose and P. Morin. Competitive online routing in geometric graphs. *Theoret. Comput. Sci.*, 324(2):273–288, 2004.

[14] P. Bose, P. Morin, A. van Renssen, and S. Verdonschot. The θ_5-graph is a spanner. *Comput. Geom. Theory Appl.*, 48(2):108–119, 2015.

[15] P. Bose and M. Smid. On plane geometric spanners: A survey and open problems. *Comput. Geom. Theory Appl.*, 46(7):818–830, 2013.

[16] S. Chechik. Compact routing schemes with improved stretch. In *Proc. ACM Symp. Princ. Dist. Comp. (PODC)*, pages 33–41, 2013.

[17] K. L. Clarkson. Approximation algorithms for shortest path motion planning. In *Proc. 19th Annu. ACM Sympos. Theory Comput. (STOC)*, pages 56–65, 1987.

[18] L. J. Cowen. Compact routing with minimum stretch. *J. Algorithms*, 38(1):170–183, 2001.

[19] M. Damian and N. Nelavalli. Improved bounds on the stretch factor of y_4. *Comput. Geom. Theory Appl.*, 62:14–24, 2017.

[20] M. Damian and K. Raudonis. Yao graphs span Theta graphs. *Discrete Mathematics, Algorithms and Applications*, 4(02):1250024, 2012.

[21] T. Eilam, C. Gavoille, and D. Peleg. Compact routing schemes with low stretch factor. *J. Algorithms*, 46(2):97–114, 2003.

[22] P. Fraigniaud and C. Gavoille. Routing in trees. In *Proc. 28th Internat. Colloq. Automata Lang. Program. (ICALP)*, pages 757–772, 2001.

[23] S. Giordano and I. Stojmenovic. Position based routing algorithms for ad hoc networks: A taxonomy. In *Ad hoc wireless networking*, pages 103–136. Springer-Verlag, 2004.

[24] L. J. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2:209–233, 1987.

[25] J. Hershberger and S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM J. Comput.*, 28(6):2215–2256, 1999.

[26] H. Kaplan, W. Mulzer, L. Roditty, and P. Seiferth. Routing in unit disk graphs. *Algorithmica*, 80(3):830–848, 2018.

[27] S. Kapoor and S. Maheshwari. Efficient algorithms for Euclidean shortest path and visibility problems with polygonal obstacles. In *Proc. 4th Annu. Sympos. Comput. Geom. (SoCG)*, pages 172–182, 1988.

[28] S. Kapoor, S. Maheshwari, and J. S. Mitchell. An efficient algorithm for Euclidean shortest paths among polygonal obstacles in the plane. *Discrete Comput. Geom.*, 18(4):377–383, 1997.

[29] G. Konjevod, A. W. Richa, and D. Xia. Scale-free compact routing schemes in networks of low doubling dimension. *ACM Trans. Algorithms*, 12(3):27:1–27:29, 2016.

[30] J. S. Mitchell. A new algorithm for shortest paths among obstacles in the plane. *Annals of Mathematics and Artificial Intelligence*, 3(1):83–105, 1991.

[31] J. S. Mitchell. Shortest paths among obstacles in the plane. *Internat. J. Comput. Geom. Appl.*, 6(03):309–332, 1996.

[32] G. Narasimhan and M. Smid. *Geometric spanner networks*. Cambridge University Press, 2007.

[33] M. H. Overmars and E. Welzl. New methods for computing visibility graphs. In *Proc. 4th Annu. Sympos. Comput. Geom. (SoCG)*, pages 164–171, 1988.

[34] D. Peleg and E. Upfal. A trade-off between space and efficiency for routing tables. *J. ACM*, 36(3):510–530, 1989.

[35] L. Roditty and R. Tov. New routing techniques and their applications. In *Proc. ACM Symp. Princ. Dist. Comp. (PODC)*, pages 23–32, 2015.

[36] L. Roditty and R. Tov. Close to linear space routing schemes. *Distributed Computing*, 29(1):65–74, 2016.

[37] N. Santoro and R. Khatib. Labelling and implicit routing in networks. *The Computer Journal*, 28(1):5–8, 1985.

[38] M. Sharir and A. Schorr. On shortest paths in polyhedral spaces. *SIAM J. Comput.*, 15(1):193–215, 1986.

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

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

[41] M. Thorup and U. Zwick. Compact routing schemes. In *Proc. 13th ACM Symp. Par. Algo. Arch. (SPAA)*, pages 1–10, 2001.

[42] M. Thorup and U. Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005.

[43] E. Welzl. Constructing the visibility graph for n-line segments in $\mathcal{O}(n^2)$ time. *Inform. Process. Lett.*, 20(4):167–171, 1985.

[44] C. Yan, Y. Xiang, and F. F. Dragan. Compact and low delay routing labeling scheme for unit disk graphs. *Comput. Geom. Theory Appl.*, 45(7):305–325, 2012.

[45] A. C.-C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Comput.*, 11(4):721–736, 1982.