Routing in Simple Polygons

Matias Korman* Wolfgang Mulzer† André van Renssen‡,§ Marcel Roeloffzen‡,§ Paul Seiferth†
Yannik Stein† Birgit Vogtenhuber¶ Max Willert†

Abstract

A routing scheme $\mathcal{R}$ in a network $G = (V, E)$ is an algorithm that allows to send messages from one node to another in the network. We are first allowed a preprocessing phase in which we assign a unique *label* to each node $p \in V$ and a *routing table* with additional information. After this preprocessing, the routing algorithm itself must be local (i.e., we can only use the information from the label of the target and the routing table of the node that we are currently at).

We present a routing scheme for routing in simple polygons: for any $\varepsilon > 0$ the routing scheme provides a stretch of $1+\varepsilon$, labels have $\mathcal{O}(\log n)$ bits, the corresponding routing tables are of size $\mathcal{O}(\varepsilon^{-1} \log n)$, and the preprocessing time is $\mathcal{O}(n^2 + \varepsilon^{-1} n)$. This improves the best known strategies for general graphs by Roditty and Tov (Distributed Computing 2016).

1 Introduction

Routing is the act of sending a message from a node to its desired target. We would like to design a routing protocol that, for any network, can send messages from any node to any target. Although the problem is easy in small networks, it provides a significant challenge for large networks [1].

There are two key features of a routing scheme. First of all, it must be *local*: while the message is at a particular node, it can only use information stored in the memory of that node. In order to save storage, we would like that the amount of information that each node stores is relatively small. Second, the routing scheme should be *efficient*, meaning that the message should not travel much further than necessary.

A straight-forward solution is to explicitly store the whole network in each node. Routing can then be solved with a single source shortest path query, sending the message one step forward, and repeating the process until we eventually reach the final target. This is clearly local and efficient, but it requires a large amount of storage. Thus, the aim is to obtain some trade-off between the amount of information stored at each node, and the ratio between the distance travelled by messages and the shortest possible path for them in the network.

For general graphs, this problem has been well-studied since the 1980's [5, 6]. The most recent result is from Roditty and Tov [7] who developed a routing scheme for general graphs G with n nodes and m edges. The scheme has poly-logarithmic header size and routes a message from p to q on a path with length $\mathcal{O}(k\Delta + m^{1/k})$ for any integer $k > 2$, where Δ is the distance of the shortest path between p and q in G. Their routing tables use $mn^{\mathcal{O}(1/\sqrt{\log n})}$ total space, which is asymptotically optimal [5].

In this paper we provide a better algorithm, albeit for a specialized graph class: the visibility graphs of polygons. Given a simple polygon P of n vertices, we connect two vertices by an edge if and only if they can see each other (i.e., the segment connecting them is contained in P). Although many shortest path problems in polygons have been considered [2, 4], there are no routing schemes for visibility graphs of polygons. In this paper we present the first routing scheme for this graph class. For n vertices and any $\varepsilon > 0$, the routing scheme needs at most $\mathcal{O}(\varepsilon^{-1} \log n)$ bits for the routing table of each vertex and produces a routing path with stretch $1 + \varepsilon$. Therefore, for this class, our results provide a more efficient routing scheme than the one of Roditty and Tov [7].

2 Preliminaries

We model a network with an *undirected, connected* and *simple* graph $G = (V, E)$. We assume it is embedded in the Euclidean plane: a *node* $p = (p_x, p_y) \in V$ corresponds to a point and an edge $\{p, q\} \in E$ is represented by the segment $\overline{pq}$. We denote by $|\overline{pq}|$ the Euclidean distance between the points p and q and call $|\overline{pq}|$ the *weight* of the corresponding edge. The length of a shortest path in G connecting two points $p, q \in V$ is denoted by $d(p, q)$. From now on, we assume that G is the visibility graph of the n vertices of P (and thus, d encodes the *geodesic* distance in P).

There are several definitions for routing schemes for a network [3, 7]. In the following we introduce a restricted standard model that is comparable to the

*Tohoku University, Sendai, Japan. Partially supported by the ELC project (MEXT KAKENHI No. 12H00855 and 15H02665).

†Department of Computer Science, Freie Universität Berlin, Germany

‡National Institute of Informatics (NII), Tokyo, Japan.

§JST, ERATO, Kawarabayashi Large Graph Project.

¶Institute for Software Technology, Graz University of Technology, Graz, Austria.

other definitions. Each node in G has a unique tag, called *label*, that identifies the node in the network, as well as some additional information stored in a *routing table*. Starting at any node $p \in V$, the routing scheme uses only the information of p's routing table (and the target's label) to compute a node adjacent to p where the message is forwarded to. This process is repeated until it reaches the target.

Definition 2.1 *A routing scheme $\mathcal{R} = (l, \rho, f)$ of a graph G consists of the following elements: a label $l(p) \in \{0,1\}^*$ and a routing table $\rho(p) \in \{0,1\}^*$ for each node $p \in V$, as well as a routing function $f\colon V \times \{0,1\}^* \to V$.*

The transition function f models the behaviour of the routing scheme. For any two nodes $p, q \in V$, consider the sequence of points given by $p_0 = p$ and $p_i = f(p_{i-1}, l(q))$ for $i \geq 1$ (i.e., the nodes visited in the routing scheme from p to q). We say that a routing scheme $\mathcal{R}$ is *correct* if for any $p, q \in V$ there exists a $k = k(p,q) \geq 0$ such that $p_k = q$ and $p_i \neq q$ for $i < k$. 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:

- *label size* $L(n) = \max_{|V|=n} \max_{p \in V} |l(p)|$,
- *table size* $T(n) = \max_{|V|=n} \max_{p \in V} |\rho(p)|$,
- *stretch* $\zeta(n) = \max_{|V|=n} \max_{p \neq q \in V} \frac{d_\rho(p,q)}{d(p,q)}$,
- and *preprocessing time* (i.e., time spent in computing the labels and routing tables).

Let P be a polygon with n vertices (which need not be in general position). Two points $p, q \in P$ can *see each other* if and only if $\overline{pq} \subset P$. Note that p and q can see each other even if the line segment $\overline{pq}$ touches the boundary of P. The *visibility graph* $\mathrm{VG}(P)$ of P is the graph whose vertex set is the vertex set of P. Two vertices are connected with an edge in $\mathrm{VG}(P)$ if in P they see each other. In this paper we show that for any $\varepsilon > 0$ we can construct a routing scheme with $\zeta(n) = 1 + \varepsilon$, $L(n) = \mathcal{O}(\log n)$ and $T(n) = \mathcal{O}(\varepsilon^{-1} \log n)$. The preprocessing time is $\mathcal{O}(n^2 + \varepsilon^{-1} n)$.

3 Cones in Polygons

Let P be a polygon with n vertices and let $t > 2$. Our approach is inspired by the Yao Graph construction [8].

Let p be a vertex of the polygon, p' the clockwise next vertex of P, α be the inner angle at p, and $\varepsilon_0 := \frac{2\pi}{\alpha t}$. We denote with r the *ray* emanating from p through p'. Next, we rotate this ray as follows: let

$$r_i(p) := \text{rotate } r \text{ clockwise by angle } \alpha \cdot \min(i \cdot \varepsilon_0, 1)$$

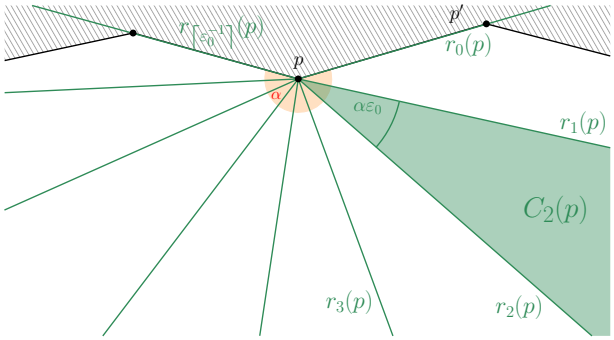

Figure 1: The cones and rays of a vertex p with inner angle α.

for $i \in \{0, 1, \ldots, \lceil \varepsilon_0^{-1} \rceil\}$. Let $C_i(p)$ be the closed *cone* with apex p and boundary $r_{i-1}(p)$ and $r_i(p)$ (see Figure 1). We also set $\mathcal{C}(p)$ as the set containing all such cones (that is, $\mathcal{C}(p) = \{C_i(p) \mid 1 \leq i \leq \lceil \varepsilon_0^{-1} \rceil\}$). By construction, the apex angle of each cone is at most $\alpha\varepsilon_0 = 2\pi/t$ and hence all cones are convex.

Lemma 3.1 *Let p be a vertex in P and $\{p, q\}$ an edge of $\mathrm{VG}(P)$ in the cone $C_i(p)$. Furthermore, let s be the closest vertex in $C_i(p)$ to p. Then the following inequality holds:*

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

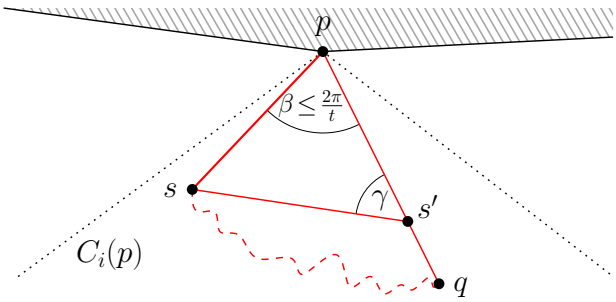

Figure 2: Illustration of Lemma 3.1. The points s and s' have the same distance to p. The dashed line represents the shortest path from s to q.

Proof. Let s' be the point on the line segment $\overline{pq}$ such that $|\overline{ps'}| = |\overline{ps}|$ (see Figure 2). Since p can see q, also s' can see q. Since s is the closest vertex to p in the cone, s can see s'. Now the triangle inequality yields $d(s,q) \leq |\overline{ss'}| + |\overline{s'q}|$. Let β and γ be the angles at p and s' in the triangle $\Delta(p,s,s')$. Since $\Delta(p,s,s')$ is in $C_i(p)$, we have $\beta \leq 2\pi/t$. Further, $\Delta(p,s,s')$ is isosceles and thus $\gamma \geq \pi/2 - \pi/t$. Applying the sine law and $\sin 2x = 2\sin x \cos x$, we get

$$|\overline{ss'}| \leq 2|\overline{ps}| \sin\frac{\pi}{t}.$$

Together with $|\overline{s'q}| = |\overline{pq}| - |\overline{ps'}| = |\overline{pq}| - |\overline{ps}|$, the triangle inequality from before gives the desired estimation. □

4 The Routing Scheme

Let $\varepsilon > 0$ and P be a polygon with vertices $p_1, \ldots, p_n$ in this order. We describe a routing scheme for $\mathrm{VG}(P)$ with stretch $1+\varepsilon$. For each each vertex p, we will partition the other vertices into intervals and assign one interval to each cone. Given a target vertex q, we look for the cone $C_i(p)$ whose associated interval contains q, and transmit the message to the nearest neighbour in $C_i(p)$.

Preprocessing In the preprocessing phase we do as follows. First, we assign to each vertex p_j of P the binary representation of j as its label, Which needs $\mathcal{O}(\log n)$ bits. For the routing table of a vertex p, we first compute the visibility polygon $\mathrm{vis}(p)$. This computation provides a sequence of consecutive points $v_0 v_1 \ldots v_k$ with $p = v_0 = v_{k+1}$. Each point of this sequence is either a vertex of P or the first proper intersection of a ray from p towards a reflex vertex. Notice that, as we walk clockwise along the visibility polygon, the angle spanned by the ray $r_0(p)$ and the edge $\{p, v_j\}$ increases monotonically. We set $t := \pi / \arcsin\left(0.5/\left(1+\varepsilon^{-1}\right)\right)$ and use the subdivision of $\mathrm{vis}(p)$ described in Section 3. This subdivision provides a set $\mathcal{C}(p)$ with a certain number of cones. The following obvious lemma specifies this number.

Lemma 4.1 *We have* $t \leq 2\pi\left(1+\varepsilon^{-1}\right)$.

The ray $r_i(p)$ intersects $\mathrm{vis}(p)$ at one or more points. Let z_i be the intersection point that is closest to p and e_i the edge of P containing it. All points z_i and edges e_i can be obtained by going once through the sequence $v_0 v_1 \ldots v_k$ (see Figure 3).

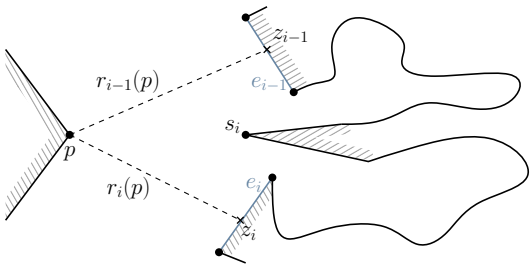

Figure 3: The boundaries of $C_i(p)$ hit the boundary of P in the points z_{i-1} and z_i. The vertex s_i is the point in $C_i(p)$ with shortest distance to p.

For all cones $C_i(p) \in \mathcal{C}(p)$, we start from z_{i-1}, walk clockwise along the boundary of P until we meet z_i, and collect all the visited vertices. If we take the indices modulo n, the collection forms an interval called $I(i)$. Among $I(i)$ we look for the vertex s_i with smallest distance to p. We store the interval boundaries of $I(i)$ together with the label of the vertex s_i in the routing table of p. This requires $\mathcal{O}(\log n)$ bits as well. Hence, by Lemma 4.1, the size of each routing table is $\mathcal{O}(\varepsilon^{-1} \log n)$ bits.

Routing phase The routing strategy is simple: when routing from a vertex p to a target q, we search in the routing table of p for the index i whose associated interval $I(i)$ contains the label of q, and then transmit the message to s_i. The following lemma proves that the algorithm is well defined.

Lemma 4.2 *Let p, q be two vertices of P and (p, q') the first edge on the shortest path from p to q. If $q \in I(i)$, then $q' \in C_i(p)$.*

Proof. Suppose that $q' \notin C_i(p)$. Since q is in $I(i)$, the shortest path π from p to q has to cross $\overline{pz_{i-1}}$ or $\overline{pz_i}$ at least twice. The first intersection is p itself. Let z be the last intersection and let π' be the subpath of π from p to z via q'. By the triangle inequality, $|\overline{pz}|$ is strictly smaller than the length of π' (see Figure 4). Thus, we can find a shortcut from p to z and hence a shorter path from p to q. □

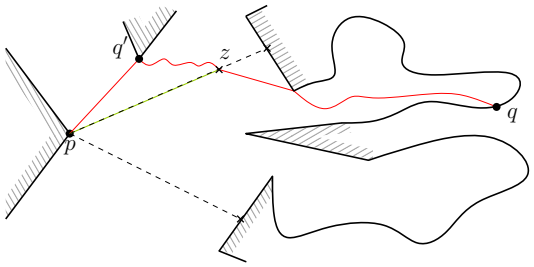

Figure 4: The red line is the "shortest" path from p to q with q' as first step, whereas the green dashed line represents a shortcut from p to z.

5 Analysis

The aim of this section is to show that for any polygon P and $\varepsilon > 0$ the stretch $1+\varepsilon$ is always preserved. First, we show that our routing strategy decreases the distance to the target.

Lemma 5.1 *Let p and q be two vertices in P, and let s be the next vertex computed by the routing scheme from p to q. Then we have $d(s,q) \leq d(p,q) - |\overline{ps}|/(1+\varepsilon)$.*

Proof. Our routing strategy routes from p to a vertex $s = s_i$ that is the closest in some cone $C_i(p)$. By Lemma 4.2, we know that the next vertex q' on the shortest path from p to q is also contained in $C_i(p)$. Thus, by Lemma 3.1 and the triangle inequality we obtain

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

□

Since the distance to the target decreases at each step, this implies that our routing scheme terminates. We now bound the stretch of the routing scheme.

Lemma 5.2 *Let p and q be two vertices of P. Then we have $d_\rho(p,q) \leq (1+\varepsilon)d(p,q)$, where $d_\rho(p,q)$ is the length of the routed path.*

Proof. Let $\pi = p_0 p_1 \dots p_k$ be the path from $p = p_0$ to $q = p_k$ computed by the routing scheme. By Lemma 5.1 we have $d(p_{i+1},q) \leq d(p_i,q) - |\overline{p_i p_{i+1}}|/(1+\varepsilon)$. Thus, we have

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

since $p_0 = p$ and $p_k = q$. This finishes the proof for the bound on the stretch. □

Thereby, we obtain our main theorem.

Theorem 5.3 *Let P be a simple polygon with n vertices. For any $\varepsilon > 0$ we can preprocess P into a routing scheme for $\mathrm{VG}(P)$ with labels of $\mathcal{O}(\log n)$ 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 $\leq (1+\varepsilon)d(p,q)$. The preprocessing time is $\mathcal{O}(n^2 + \varepsilon^{-1}n)$.*

Proof. The stretch and size bounds follow from previous arguments. We focus on the preprocessing time bound. For any vertex $p \in P$ let L be the sequence of vertices $v_0 v_1 \dots v_k$ of $\mathrm{vis}(p)$ calculated in time $\mathcal{O}(n)$. Using L, we can find all intersection points z_i and corresponding edges e_i of P in time $\mathcal{O}(n + \varepsilon^{-1})$. Thus, for each ray $r_i(p)$, we can find the boundaries of e_i in amortized constant time. Once the edges e_i are computed, we can find the interval boundaries of $I(i)$ in constant time. The point within the interval $I(i)$ with smallest distance to p can be found by going once through $I(i)$ in $\mathcal{O}(|I(i)|)$ time. For all cones of one vertex, this step takes $\mathcal{O}(n+\varepsilon^{-1})$ in total. In the end, we do the same procedure for all vertices and obtain the running time $\mathcal{O}(n^2 + \varepsilon^{-1}n)$. □

6 Conclusion

We still have various open questions for the routing schemes for polygons. First of all, it would be interesting, if there is a routing scheme that approximates the hop-distance in polygons, where each pair of adjacent vertices has edge weight 1. Using our routing scheme we can find examples, where the stretch is in $\Omega(n)$. Further, it would be interesting to know whether the preprocessing time or the size of the routing table can be improved, perhaps using a recursive strategy.

The routing scheme extends to polygonal domains with h holes: we now need to apply the same strategy to each vertex p, cone $C_i(p)$ and the $h+1$ simple polygons that form the boundary of P. This increases the size of the routing table to $\mathcal{O}(\varepsilon^{-1}h\log n)$ and preprocessing time to $\mathcal{O}(n^2\log n + hn^2 + \varepsilon^{-1}hn)$. This is impractical for large values of h, and thus we wonder whether a better strategy exists for this case.

References

[1] 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.

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

[3] Haim Kaplan, Wolfgang Mulzer, Liam Roditty, and Paul Seiferth. Routing in unit disk graphs. In *Proc. 12th Latin American Symp. Theoretical Inf. (LATIN)*, pages 536–548, 2016.

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

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

[6] Liam Roditty and Roei Tov. New routing techniques and their applications. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 23–32. ACM, 2015.

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

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