Rectilinear Link Diameter and Radius in a Rectilinear Polygonal Domain

Elena Arseneva[1]
St. Petersburg State University, St. Petersburg, Russia
ea.arseneva@gmail.com
https://orcid.org/0000-0002-5267-4512

Man-Kwun Chiu[7,2]
Institut für Informatik, Freie Universität Berlin, Berlin, Germany
chiumk@zedat.fu-berlin.de

Matias Korman[3]
Tufts University, Boston, USA
matias.korman@tufts.edu

Aleksandar Markovic[4]
TU Eindhoven, Eindhoven, The Netherlands
a.markovic@tue.nl

Yoshio Okamoto[5]
University of Electro-Communications, Tokyo, Japan
RIKEN Center for Advanced Intelligent Project, Tokyo, Japan
okamotoy@uec.ac.jp
https://orcid.org/0000-0002-9826-7074

Aurélien Ooms[6]
Université libre de Bruxelles (ULB), Brussels, Belgium
aureooms@ulb.ac.be
https://orcid.org/0000-0002-5733-1383

André van Renssen[7]
University of Sydney, Sydney, Australia
andre.vanrenssen@sydney.edu.au
https://orcid.org/0000-0002-9294-9947

Marcel Roeloffzen[7]
TU Eindhoven, Eindhoven, The Netherlands
m.j.m.roeloffzen@tue.nl

Abstract

We study the computation of the diameter and radius under the rectilinear link distance within a rectilinear polygonal domain of n vertices and h holes. We introduce a *graph of oriented distances* to encode the distance between pairs of points of the domain. This helps us transform the problem so that we can search through the candidates more efficiently. Our algorithm computes both the

diameter and the radius in $O(\min(n^\omega, n^2 + nh \log h + \chi^2))$ time, where $\omega < 2.373$ denotes the matrix multiplication exponent and $\chi \in \Omega(n) \cap O(n^2)$ is the number of edges of the graph of oriented distances. We also provide an alternative algorithm for computing the diameter that runs in $O(n^2 \log n)$ time.

1 Introduction

Diameters and radii are popular characteristics of metric spaces. For a compact set S with a metric $d \colon S \times S \to \mathbb{R}^+$, its diameter is defined as $\mathrm{diam}(S) := \max_{p \in S} \max_{q \in S} d(p, q)$, and its radius is defined as $\mathrm{rad}(S) := \min_{p \in S} \max_{q \in S} d(p, q)$. The pair (p, q) and the point p that realize these distances are called the *diametral pair* and *center*, respectively. These terms are the natural extension of the same concepts in a disk and give some interesting properties of the environment, such as the worst-case response time or ideal location of a serving facility.

Much research has been devoted towards finding efficient algorithms to compute the diameter and radius for various types of sets and metrics. In computational geometry, one of the most well-studied and natural metric spaces is a polygon in the plane. This paper focuses on the computation of the diameter and the radius of a rectilinear polygon, possibly with holes (i.e., a *rectilinear polygonal domain*) under the *rectilinear link distance*. Intuitively, this metric measures the minimum number of links (segments) required in any rectilinear path connecting two points in the domain, where rectilinear indicates that we are restricted to horizontal and vertical segments only.

1.1 Previous Work

The ordinary link distance is a very natural metric and simple to describe. Initially, the interest was motivated by the potential robotics applications (i.e., having some kind of robot with wheels for which moving in a straight line is easy, but making turns is costly in time or energy). Since then, it has attracted a lot of attention from a theoretical point of view.

Indeed, many problems that are easy under the L_1 or Euclidean metric turn out to be more challenging under the link distance. For example, the shortest path between two points in a polygonal domain can be found in $O(n \log n)$ time for both Euclidean [9] and L_1 metrics [11, 12]. However, even approximating the shortest path within a factor of $(2 - \epsilon)$ under the link distance is 3-SUM hard [13], and thus it is unlikely that a significantly subquadratic-time algorithm is possible.

The problem of computing the diameter and radius is no exception to this rule: when polygons are simple (i.e., they do not have holes) and have n vertices, the diameter and center can be found in linear time for both Euclidean [1, 8] and L_1 metrics [4]. However, the best known algorithms for the link distance run in $O(n \log n)$ time [6, 17]. Lowering the running times or proving the impossibility of this is a longstanding open problem in the field. The only partial answer to this question was given by Nilsson and Schuierer [15, 16]; they showed that the diameter and center can be found in linear time under the rectilinear link distance (i.e., when we are only allowed to use rectilinear paths).

■ **Table 1** Summary of the best known results for computing the diameter and radius of a polygonal domain of n vertices and h holes under different metrics. In the table, $\omega < 2.373$ is the matrix multiplication exponent.

Metric	Simple polygon		Polygonal domain	
	Diameter	Radius	Diameter	Radius
Euclidean	$O(n)$ [8]	$O(n)$ [1]	$O(n^{7.73})$ [3]	$O(n^{11} \log n)$ [18]
L_1	$O(n)$ [4]	$O(n)$ [4]	$O(n^2 + h^4)$ [2]	$\tilde{O}(n^4 + n^2 h^4)$ [2]
Ordinary link	$O(n \log n)$ [17]	$O(n \log n)$ [6]	open	open
Rectilinear link	$O(n)$ [15]	$O(n)$ [16]	$O(n^2 \log n)$ (Thm. 10)	$O(n^\omega)$ (Thm. 12)
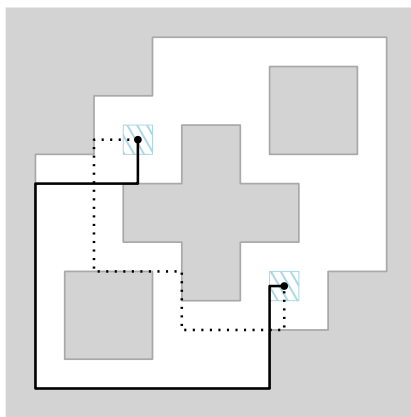
■ **Figure 1** An example showing no diametral pair lies on the boundary of the polygonal domain. The points in the dashed blue regions will have distance 6 from each other (out of the 4 shortest paths connecting them two are shown) whereas other pairs will have distance 5 or less.
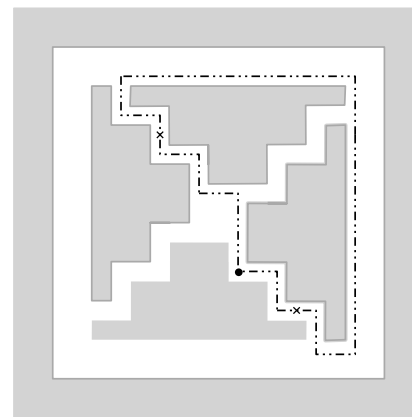
■ **Figure 2** Example with diameter 8 (crossed points) and radius 7 (dotted point). By increasing the number of bends in the holes the diameter and radius become arbitrarily close. Note that any point in the domain is either a center or belongs to a diametral pair.

We focus on polygons with holes. The addition of holes to the domain introduces significant difficulties to the problem. For example, the diameter and radius under the rectilinear link distance can be uniquely realized by points in the interior of a polygonal domain (see Figure 1). Hence, it does not suffice to determine the distance only between every pair of vertices of the domain. Other strange situations can happen, such as the diameter and radius being arbitrarily close (see e.g. Figure 2).

These difficulties have a clear impact in the runtime of the algorithms. In most metrics, the runtime changes from linear or slightly superlinear to large polynomial terms. The difference between the link distance and other metrics becomes even more significant: no algorithm for computing the diameter and radius under the link distance is known, not even one that runs in exponential time (or one that works for particular cases such as rectilinear polygons). A summary of the best running time for computing the diameter and center under different metrics can be found in Table 1.

In this paper we provide the first step towards understanding such a difficult metric. Similarly to the simple polygon case [15, 16], we start by considering the computation of both the diameter and radius under the rectilinear link distance. We hope that the ideas of this paper will motivate future research in solving the more difficult problem of computing the diameter and radius under the (ordinary) link distance.

1.2 Results

Several of the difficulties of the link distance disappear when restricting the problem to a rectilinear setting. For example, one can easily partition the domain into rectangular cells such that all points in a cell have the same distance to all points in another cell. With this partition, brute-force algorithms that find the diameter and radius in $O(n^3 \log n)$ time immediately follow. Alternatively, you can use a slightly coarser method to approximate either value: in $O(n^2 + nh \log h)$ or $O(n^2 \log \log n)$ time we can compute an estimate of the diameter (details of these methods are given in Section 2). This estimate will either be the exact diameter or will be the diameter plus one (i.e., the path computed may contain an additional link that is not needed).

In our work we improve this second approach. By using some geometric observations, we characterize exactly when the estimate is off by one unit. Thus, we can transform the approximation algorithm into an exact one by adding a verification step that checks whether or not the one additive error has actually happened.

We provide three different algorithms for making the above additional verification step. In Section 3 we characterize what we should look for to determine what the exact diameter is. This characterization then leads to a brute-force algorithm that runs in $O(n^2 + nh \log h + \chi^2)$ time, where χ is a parameter of the input that ranges from $\Theta(n)$ to $\Theta(n^2)$. To reduce running times when χ is large we present another algorithm to compute the diameter in Section 4. This algorithm, which runs in $O(n^2 \log n)$ time, exploits properties of the diameter. Specifically, we heavily use that this value is a maximum over a maximum of distances, hence it can only be used for the diameter (recall that we have a minimum-maximum alternation in the definition of the radius). For the radius we then present a third algorithm that uses matrix multiplication to speed up computation. This solution runs in time $O(n^\omega)$, where $\omega < 2.373$ is the matrix multiplication exponent (Le Gall [10] provided the best known bound on ω). This last solution can also be adapted to compute the diameter, but our second algorithm results in a faster method.

Another interesting benefit of our approach is that we may be able to obtain a *certificate*. In previous algorithms for computing the diameter or center in polygonal domains, the diameter is found via exhaustive search. Thus, even if somehow the points that realize the diameter or center are given, the only way to verify that the answer is correct is to run the whole algorithm. In our algorithm, knowing the diameter can reduce the time needed for verification. Although the reduction in computation time is not large (from $O(n^2 \log n)$ for computing to $O(n^2 \log \log n)$ for verifying the diameter, for example), we find it to be of theoretical interest.

Further note that, when comparing with the algorithms for other metrics, the running time for simple and polygonal domains differs by at least a cubic factor. In our case, running times only increase by a slightly superlinear factor when compared to the case of simple polygons [15, 16]. This is partially due to the fact that rectilinear link distance is much easier than the ordinary link distance, but also because we use this new verification approach. We believe this to be our main contribution and hope that it motivates a similar approach in other metrics.

1.3 Preliminaries

A *rectilinear simple polygon* (also called an orthogonal polygon) is a simple polygon that has horizontal and vertical edges only. A *rectilinear polygonal domain* P with h pairwise disjoint holes and n vertices is a connected and compact subset of $\mathbb{R}^2$ with h pairwise disjoint holes, in which the boundary of each hole is a simple closed rectilinear curve. Thus, the boundary ∂P of P consists of n line segments.

Each of the holes as well as the outer boundary of P is regarded as an *obstacle* that paths in P are not allowed to cross. A *rectilinear path* π from $p \in P$ to $q \in P$ is a path from p to q that consists of vertical and horizontal segments, each contained in P, and such that along π each vertical segment is followed by a horizontal one and vice versa. Recall that P is a closed set, so π can traverse the boundary of P (along the outer face and any of the h obstacles).

We define the *link length* of such a path to be the number of segments composing it. The *rectilinear link distance* between points $p, q \in P$ is defined as the minimum link length of a rectilinear path from p to q in P, and denoted by $\ell_P(p, q)$. It is well known that in rectilinear polygonal domains there always exists a rectilinear polygonal path between any two points $p, q \in P$, and thus the distance is well defined. Once the distance is defined, the definitions of *rectilinear link diameter* diam(P) and *rectilinear link radius* rad(P) directly follow.

For simplicity in the description, we assume that a pair of vertices do not share the same x- or y-coordinate unless they are connected by an edge. This general position assumption can be removed with classic symbolic perturbation techniques. Also notice that, since we are considering rectilinear polygons, no edge has length 0. However, for simplicity in the analysis we will allow edges in a rectilinear path to have length 0. These edges of length 0 are considered as edges and thus potentially contribute to the link distance (naturally, no shortest path will ever have such an edge). The reason for considering these is that we will consider oriented paths, where the first and last edge are forced to be horizontal or vertical, this enforcement may require edges of length 0. From now on, for ease of reading, we will refer to rectilinear simple polygons and rectilinear polygonal domains as "simple polygons" and "domains." Similarly, we will use the term "distance" to refer to the rectilinear link distance.

2 Graph of Oriented Distances

In this section we introduce the graph of oriented distances and show how it can be used to encode the rectilinear link distance between points of the domain. We note that, although we have not been able to find a reference to this graph in the literature, some properties are already known. For example, the horizontal and vertical decompositions (defined below) were used by Mitchell et al. [14] to compute minimum-link rectilinear paths.

For any domain P, we extend any horizontal segment of the domain to the left and right until it hits another segment of P, partitioning it into rectangles. We call this partition the *horizontal decomposition*. Let $\mathcal{H}(P)$ be the set containing those rectangles. Similarly, if we extend all the vertical segments up and down, we get the *vertical decomposition*. Let $\mathcal{V}(P)$ be the set of rectangles in this second decomposition. Observe that both decompositions have linear size and can be computed in $O(n \log n)$ time with a plane sweep.

The overlay of both subdivisions creates a finer subdivision that has the well-known property that pairwise cell distance is constant (that is, the distance between any pair of points in two fixed cells of this subdivision will remain constant). Thus, by computing the distance between all pairs of cells we can find both the diameter and center. The major problem of this approach is that the finer subdivision may have $\Omega(n^2)$ cells, and thus it is hard to obtain an algorithm that runs in subcubic time. Instead, we avoid the overlay and use both subdivisions separately to obtain the diameter.

Given two rectangles $i, j \in \mathcal{H}(P) \cup \mathcal{V}(P)$, we use $i \sqcap j$ to denote the boolean operation which returns *true* if and only if the rectangles i and j properly intersect (i.e. their intersection has non-zero area). This implies that one of i, j belongs to $\mathcal{H}(P)$, and the other to $\mathcal{V}(P)$.

▶ **Definition 1** (Graph of Oriented Distances). Given a rectilinear polygonal domain P, let $\mathcal{G}(P)$ be the unweighted undirected graph defined as $\mathcal{G}(P) = (\mathcal{H}(P) \cup \mathcal{V}(P), \{\, (h, v) \in \mathcal{H}(P) \times \mathcal{V}(P) \colon h \sqcap v \,\})$.

In other words, vertices of $\mathcal{G}(P)$ correspond to rectangles of the horizontal and the vertical decompositions of P. We add an edge between two vertices if and only if the corresponding rectangles properly intersect. Note that this graph is bipartite, and has $O(n)$ vertices. From now on, we make a slight abuse of notation and identify a rectangle with its corresponding vertex (thus, we talk about the neighbors of a rectangle $i \in \mathcal{H}(P)$ in $\mathcal{G}(P)$, for example).

The name *Graph of Oriented Distances* is explained as follows (see also the paragraph after Lemma 4). Consider a rectilinear path π between two points in P. Each horizontal edge of π is contained in a rectangle of $\mathcal{H}(P)$ and each vertical edge is contained in a rectangle of $\mathcal{V}(P)$. A bend in the path takes place in the intersection of the rectangles containing the two adjacent edges and corresponds to an edge of $\mathcal{G}(P)$. So every rectilinear path π has a corresponding walk π' in $\mathcal{G}(P)$ (and *vice versa*). Moreover, each bend of π is associated with an edge of π'.

▶ **Definition 2** (Oriented distance). Given a rectilinear polygonal domain P, let i and j be two vertices of $\mathcal{G}(P)$, let $\Delta(i, j)$ to be the length of the shortest path between i and j in graph $\mathcal{G}(P)$ plus one. We also define $\Delta(i, i) = 1$.

The reason why we add the extra unit is to make sure that the link distance and the oriented distance match (see Lemma 4 below). We first list some useful properties of the oriented distance, which directly follow from the definition. Then we show the relationship between the oriented distance $\Delta(\cdot, \cdot)$ in $\mathcal{G}(P)$ and the link distance $\ell_P(\cdot, \cdot)$ in P.

▶ **Lemma 3.** *Let i, j, i', j' be any (not necessarily distinct) rectangles in $\mathcal{H}(P) \cup \mathcal{V}(P)$ such that $i \sqcap i'$, and $j \sqcap j'$. Then, the following hold.*
(a) $\Delta(i, j) = \Delta(j, i)$.
(b) $\Delta(i', j) \in \{\, \Delta(i, j) - 1, \Delta(i, j) + 1 \,\}$.
(c) $\Delta(i', j') \in \{\, \Delta(i, j) - 2, \Delta(i, j), \Delta(i, j) + 2 \,\}$.

▶ **Lemma 4.** *Let p and q be two points of the rectilinear polygonal domain P. The rectilinear link distance $\ell_P(p, q)$ between p and q can be characterized as follows. If p and q lie in the same vertical or horizontal rectangle of $\mathcal{V}(P)$ or $\mathcal{H}(P)$ then $\ell_P(p, q) = 1$ (if p and q share a coordinate) or $\ell_P(p, q) = 2$ (if both x- and y-coordinates of p and q are distinct). Otherwise, let $i \in \mathcal{H}(P)$, $i' \in \mathcal{V}(P)$, $j \in \mathcal{H}(P)$ and $j' \in \mathcal{V}(P)$ be vertices of the graph of oriented distances such that $p \in i \cap i'$ and $q \in j \cap j'$. Then*

$$\ell_P(p, q) = \min\{\, \Delta(i, j), \Delta(i, j'), \Delta(i', j), \Delta(i', j') \,\}.$$

Intuitively speaking, if we are given two disjoint rectangles $i, j \in \mathcal{H}(P)$, then $\Delta(i, j)$ denotes the minimum number of links needed to connect any two points $p \in i$ and $q \in j$ under the constraint that the first and the last segments of the path are horizontal. If we looked for rectangles in $\mathcal{V}(P)$, we would instead require that the path starts (or ends) with vertical segments. It follows that the link distance is the minimum of the four possible options.

In our algorithms we will often look for oriented distances between rectangles, so we compute it and store them in a preprocessing phase. Fortunately, a similar decomposition was used by Mitchell et al. [14]. Specifically, they show how to compute the distance from a single rectangle to all other rectangles in $O(n + h \log h)$ time with an $O(n)$-size data structure.[8]

---

[8] As a subproblem towards obtaining their main result, Mitchell et al. [14] show how to compute the

▶ **Lemma 5** ([14]). *Given the horizontal and vertical decompositions $\mathcal{H}(P)$ and $\mathcal{V}(P)$ we can compute for a single rectangle i in either decomposition the oriented distance $\Delta(i, j)$ to every other rectangle j in $O(n + h \log h)$ time.*

We construct this data structure for each of the $O(n)$ rectangles. This allows us to compute (and store) the $O(n^2)$ oriented distances in $O(n^2 + nh \log h)$ time. Alternatively, we can use a recent result by Chan and Skrepetos [5] to compute the same distances in $O(n^2 \log \log n)$ time.

3 Characterization via Boolean Formulas

Let $\hat{d} = \max_{i,j \in \mathcal{H}(P) \cup \mathcal{V}(P)} \Delta(i, j)$ be the largest distance between vertices of $\mathcal{G}(P)$. Similarly, we define $\hat{r} = \min_{i \in \mathcal{H}(P) \cup \mathcal{V}(P)} \max_{j \in \mathcal{H}(P) \cup \mathcal{V}(P)} \Delta(i, j)$. Note that these two values are the diameter and the radius of $\mathcal{G}(P)$ plus one (recall that we add one unit to the graph distance when defining Δ). We use $\hat{d}$ and $\hat{r}$ to approximate the diameter $\mathrm{diam}(P)$ and radius $\mathrm{rad}(P)$ of a domain P under the rectilinear link distance. First, we relate the distance between two points $p, q \in P$ to the oriented distances between the rectangles that contain p and q. Specifically, from Lemma 4, we know that $\ell_P(p, q) = \min\{\Delta(i, j), \Delta(i, j'), \Delta(i', j), \Delta(i', j')\}$, where $i, j \in \mathcal{H}(P)$ are the horizontal rectangles containing p and q, respectively, and $i', j' \in \mathcal{V}(P)$ are the vertical rectangles containing p and q. Similarly, we define $\hat{\ell}(p, q) = \max\{\Delta(i, j), \Delta(i, j'), \Delta(i', j), \Delta(i', j')\}$. It then follows from Lemma 3 that these two values differ by at most 2.

▶ **Lemma 6.** *For any two points $p, q \in P$, let $i, j \in \mathcal{H}(P)$ and $i', j' \in \mathcal{V}(P)$ be the rectangles containing p and q, i.e., $p \in i \cap i'$ and $q \in j \cap j'$. Then, it holds that $\hat{\ell}(p, q) - 2 \leq \ell_P(p, q) \leq \hat{\ell}(p, q) - 1$.*

This relation allows us to express the rectilinear link diameter of a domain in terms of $\hat{d}$.

▶ **Theorem 7.** *The rectilinear link diameter $\mathrm{diam}(P)$ of a rectilinear polygonal domain P satisfies $\mathrm{diam}(P) = \hat{d} - 1$ if and only if there exist $i, i', j, j' \in \mathcal{H}(P) \cup \mathcal{V}(P)$ with $i \sqcap i'$ and $j \sqcap j'$, such that $\Delta(i, j) = \hat{d}$ and $\Delta(i', j') = \hat{d}$. Otherwise, $\mathrm{diam}(P) = \hat{d} - 2$.*

Proof. Before giving our proof, we emphasize that the fact that $\mathrm{diam}(P) \in \{\hat{d} - 1, \hat{d} - 2\}$ is folklore (although we have found no reference, several researchers mentioned that they were aware of it). Our major contribution is the characterization of which of the two cases it is.

Now observe that for any pair of points $p, q \in P$ we have $\ell_P(p, q) \leq \hat{\ell}(p, q) - 1 \leq \hat{d} - 1$ by Lemma 6. Hence, the diameter of P is at most $\hat{d} - 1$. Similarly, by the definitions of $\hat{d}$ and $\hat{\ell}(\cdot, \cdot)$, there must be a pair of points $p, q \in P$ so that $\hat{\ell}(p, q) = \hat{d}$. Again by Lemma 6 it follows that $\mathrm{diam}(P) \geq \ell_P(p, q) \geq \hat{\ell}(p, q) - 2 = \hat{d} - 2$.

Next we show that the diameter is $\hat{d} - 1$ if and only if the above condition holds. If $\Delta(i, j) = \hat{d}$ and $\Delta(i', j') = \hat{d}$, then by Lemma 3 and the fact that neither $\Delta(i, j')$ nor $\Delta(i', j)$ can be larger than $\hat{d}$, we know that $\Delta(i, j') = \Delta(i', j) = \hat{d} - 1$. It follows from Lemma 4 that a pair of points $p \in i \cap i'$ and $q \in j \cap j'$ has $\ell_P(p, q) = \hat{d} - 1$. Thus, the diameter is $\hat{d} - 1$.

---

distance from a single point to any other location in the domain with paths of fixed orientation. They call these the *h-h*-map, *v-v*-map, *v-h*-map and *h-v*-map and they correspond to our rectangular decompositions. Although their method considers a single starting point, it can be adapted to compute the distance from a rectangle as all points inside each rectangle we consider will have the same resulting distances to the other rectangles.

Now consider any pair p, q and the set of rectangles $i, j \in \mathcal{H}(P)$ and $i', j' \in \mathcal{V}(P)$ with $p \in i \cap i'$ and $q \in j \cap j'$. Recall that $\ell_P(p, q) = \min\{\Delta(i, j), \Delta(i, j'), \Delta(j', i), \Delta(i', j')\}$. By Lemma 3, $\Delta(i, j)$ and $\Delta(i', j')$ must differ by exactly one from $\Delta(i', j)$ and $\Delta(i, j')$. That implies that two distances may be $\hat{d} - 1$, but if the condition in the lemma is not satisfied, at most one can be $\hat{d}$ and the fourth must be $\hat{d} - 2$ or less. Therefore, if the condition is not satisfied for i, i', j, j', then the diameter is indeed $\hat{d} - 2$.   ◀

For the radius we can make a similar argument.

▶ **Theorem 8.** *The rectilinear link radius* $\mathrm{rad}(P)$ *of a rectilinear polygonal domain P satisfies* $\mathrm{rad}(P) = \hat{r} - 1$ *if and only if for all* $i, i' \in \mathcal{H}(P) \cup \mathcal{V}(P)$ *with* $i \sqcap i'$ *there exist* $j, j' \in \mathcal{H}(P) \cup \mathcal{V}(P)$ *with* $j \sqcap j'$ *such that* $\Delta(i, j) \geq \hat{r}$ *and* $\Delta(i', j') \geq \hat{r}$. *Otherwise,* $\mathrm{rad}(P) = \hat{r} - 2$.

Proof. We first show by contradiction that the real radius satisfies $\mathrm{rad}(P) \leq \hat{r} - 1$. Suppose the radius is greater than or equal to $\hat{r}$. Then, for all $p \in P$ there exists a point $q \in P$ such that $\ell_P(p, q) \geq \hat{r}$. Now consider a rectangle $i \in \mathcal{H}(P) \cup \mathcal{V}(P)$, a point $p \in i$ and a point q at distance $\hat{r}$ from p. Consider the two rectangles $j \in \mathcal{H}(P)$ and $j' \in \mathcal{V}(P)$ so that $q \in j \cap j'$. By Lemma 4 we know that $\Delta(i, j) \geq \ell_P(p, q) \geq \hat{r}$ and $\Delta(i, j') \geq \ell_P(p, q) \geq \hat{r}$. By Lemma 3b $\Delta(i, j)$ and $\Delta(i, j')$ differ by one, and thus one of them must be at least $\hat{r} + 1$. That is, for any rectangle i we can find a second rectangle at oriented distance $\hat{r} + 1$. This implies that $\hat{r} = \min_{i \in \mathcal{H}(P) \cup \mathcal{V}(P)} \max_{j \in \mathcal{H}(P) \cup \mathcal{V}(P)} \Delta(i, j) \geq \hat{r} + 1$, which is a contradiction. Therefore, our initial assumption that $\mathrm{rad}(P) \geq \hat{r}$ is false and we conclude that $\mathrm{rad}(P) \leq \hat{r} - 1$.

Next we show that $\mathrm{rad}(P) \geq \hat{r} - 2$. Consider any point p and a rectangle $i \in \mathcal{H}(P)$ that contains it. By definition of $\hat{r}$ there is a rectangle $j \in \mathcal{H}(P) \cup \mathcal{V}(P)$ so that $\Delta(i, j) \geq \hat{r}$. Let q be any point in j. From Lemma 6 we get that $\ell_P(p, q) \geq \hat{\ell}(p, q) - 2 \geq \Delta(i, j) - 2 \geq \hat{r} - 2$. Hence for any point p, there is a point q that is at distance at least $\hat{r} - 2$, which implies $\mathrm{rad}(P) \geq \hat{r} - 2$.

Now we show that if the above condition is satisfied, then it must hold that $\mathrm{rad}(P) = \hat{r} - 1$. Assume the condition holds and consider any point p and two rectangles $i, i' \in \mathcal{H}(P) \cup \mathcal{V}(P)$ so that $i \sqcap i'$ and $p \in i \cap i'$. There exist $j, j' \in \mathcal{H}(P) \cup \mathcal{V}(P)$ so that $j \sqcap j'$, $\Delta(i, j) \geq \hat{r}$, and $\Delta(i', j') \geq \hat{r}$. By Lemma 3 we know that $\Delta(i, j')$ and $\Delta(i', j)$ must be at least $\hat{r} - 1$. Therefore $\ell_P(p, q) \geq \hat{r} - 1$ for any point $q \in j \cap j'$. This shows that for any point p there is a point q whose link distance to p is at least $\hat{r} - 1$, giving a lower bound on the radius. Combining this with the upper bound shown above, we obtain that $\mathrm{rad}(P) = \hat{r} - 1$ as claimed.

If the condition is not true, then we know there exist rectangles $i, i' \in \mathcal{H}(P) \cup \mathcal{V}(P)$ so that $i \sqcap i'$, and for every $j, j' \in \mathcal{H}(P) \cup \mathcal{V}(P)$ with $j \sqcap j'$ the above statement is not true. Now consider a point $p \in i \cap i'$. We argue that p has distance at most $\hat{r} - 2$ to any other point $q \in P$. Consider any point q and let $j, j' \in \mathcal{H}(P) \cup \mathcal{V}(P)$ be the rectangles containing q. We perform a case analysis on the value of $\Delta(i, j)$. First consider the case $\Delta(i, j) \geq \hat{r} + 1$. In this case $\Delta(i', j) \geq \hat{r}$ and $\Delta(i, j') \geq \hat{r}$ which contradicts our assumption that the above statement is true for every (j, j'). If $\Delta(i, j) = \hat{r}$, then by Lemma 3 and the assumption that not both $\Delta(i, j) \geq \hat{r}$ and $\Delta(i', j') \geq \hat{r}$ we find that $\Delta(i', j') = \hat{r} - 2$ which implies that $\ell_P(p, q) \leq \hat{r} - 2$. If $\Delta(i, j) = \hat{r} - 1$, then by Lemma 3, both $\Delta(i, j')$ and $\Delta(i', j)$ differ from $\Delta(i, j)$ by 1, but by our assumption that not both $\Delta(i, j') \geq \hat{r}$ and $\Delta(i', j) \geq \hat{r}$, one of them must be $\hat{r} - 2$. Lastly, if $\Delta(i, j) \leq \hat{r} - 2$, we can already conclude that $\ell_P(p, q) \leq \hat{r} - 2$. This shows that from p any other point q is at most distance $\hat{r} - 2$ away, hence the radius is at most $\hat{r} - 2$. Combining this with the lower bound of $\hat{r} - 2$ (shown above), we conclude that the radius must be $\hat{r} - 2$.   ◀

With the above characterization, we can naively compute the diameter and the radius by checking all $O(n^4)$ quadruples $(i, i', j, j') \in \mathcal{H}(P) \times \mathcal{V}(P) \times \mathcal{H}(P) \times \mathcal{V}(P)$. However, the approach can be improved by using $\mathcal{G}(P)$.

▶ **Corollary 9.** *The rectilinear link diameter* diam(P) *and radius* rad(P) *of a rectilinear polygonal domain P consisting of n vertices and h holes can be computed in $O(n^2 + nh \log h + \chi^2)$ time, where χ is the number of edges of $\mathcal{G}(P)$ (i.e., the number of pairs of intersecting rectangles of $\mathcal{H}(P)$ and $\mathcal{V}(P)$).*

As we discuss later, this method is only useful when χ is very small, i.e. almost linear size or smaller.

Remark on the interior realization of the diameter/radius

Theorems 7 and 8 together with Lemma 3b imply that a necessary condition for the diameter to be uniquely realized by pairs of interior points is that diam$(P) = \hat{d} - 1$. Similarly, for all centers to be determined by points in the interior we must have rad$(P) = \hat{r} - 1$. However, neither condition is sufficient. This transformation of the problem into a search of quadruples of rectangles allows us to handle the interior cases in the same way as the boundary cases.

4 Computing the Diameter Faster

We present a faster method for computing the diameter. This method uses the fact that the diameter is defined as a maximum over maxima which allows us to reduce the running time to $O(n^2 \log n)$. Recall that the radius is a minimum over maxima, thus the algorithm of this section does not trivially extend to the computation of the radius. The rest of this section is the proof of the following statement.

▶ **Theorem 10.** *The rectilinear link diameter* diam(P) *of a rectilinear polygonal domain P of n vertices can be computed in $O(n^2 \log n)$ time.*

By Theorem 7, after we compute the oriented diameter $\hat{d}$, we only need to consider $\hat{d} - 1$ or $\hat{d} - 2$ as candidates to be diam(P). The following corollary of Theorem 7 can be obtained by applying Lemma 3c.

▶ **Corollary 11.** *The diameter* diam(P) *equals $\hat{d} - 2$ if and only if for all rectangles i and j with $\Delta(i,j) = \hat{d}$, and for all rectangles i' and j' with $i \sqcap i'$ and $j \sqcap j'$, we have $\Delta(i', j') = \hat{d} - 2$. Otherwise,* diam$(P) = \hat{d} - 1$.

This condition can be checked in $O(n^4)$ time in a brute-force manner as follows. We iterate over every pair (i, j) with $\Delta(i, j) = \hat{d}$. For each such pair we find the sets cover$(i) = \{i' : i \sqcap i'\}$ and cover$(j) = \{j' : j \sqcap j'\}$. Then for each pair $(i', j') \in$ cover$(i) \times$ cover(j) we check if $\Delta(i', j') = \hat{d} - 2$. If there is a pair for which this is not the case, then by the above corollary the diameter is $\hat{d} - 1$. Since each of the covers may have linear size, the running time is $\Omega(n^4)$.

The key observation that allows us to reduce this to $O(n^2 \log n)$ time is that in the end there are only $O(n^2)$ unique pairs to test. Indeed, what we are checking is the distance of every pair (i', j') in the set

$$\mathcal{T} = \{(i', j') \colon \exists i, j \text{ such that } (i' \sqcap i, j \sqcap j', \Delta(i, j) = \hat{d})\}$$

which clearly has only quadratic size. Next we show that this set has more structure than just being an arbitrary set of rectangles, which allows us to compute it more quickly.

First, instead of iterating over every pair (i, j) with $\Delta(i, j) = \hat{d}$ and computing all pairs in cover$(i) \times$ cover(j), we iterate over i and compute all pairs in cover$(i) \times \bigcup_{j \colon \Delta(i,j)=\hat{d}}$ cover(j).

For a rectangle $i \in \mathcal{H}(P) \cup \mathcal{V}(P)$, let $\mathcal{S}_i$ denote the set of rectangles at oriented distance $\hat{d}$ from i. Now let

$$\mathcal{T} = \bigcup_i \mathcal{T}_i = \bigcup_i \{(i', j') \colon \exists j \text{ such that } (i' \sqcap i, j' \sqcap j, j \in \mathcal{S}_i)\}.$$

Note that the rectangles fulfilling the role of i' are easily found (i.e., they must intersect i and must have different orientation), but naively computing the ones that fulfill the role of j' leads to a quadratic runtime. That is, if we were to compute for each $j \in \mathcal{S}_i$ its cover, then this may take $\Omega(n^2)$ time. However, there are only $O(n)$ rectangles that can fulfill the role of j' and we show how to find them in $O(n \log n)$ time.

For this purpose we use an orthogonal segment intersection reporting data structure, derived from a known dynamic ray shooting data structure [7]. The data structure we use stores horizontal line segments. It allows to add or remove horizontal line segments in $O(\log n)$ time per segment. The structure reports the first segment hit by a query ray in $O(\log n)$ time. By repeatedly using the structure, we can find all z horizontal line segments intersected by a vertical line segment in $O((z+1) \log n)$ time. While performing the query, we also remove all the reported segments from the data structure in the same time complexity.

For a rectangle k, we define the *middle segment* ℓ_k of k. If k is a horizontal rectangle, ℓ_k is the line segment connecting the midpoints of its left and right boundary; if k is a vertical rectangle, ℓ_k is the segment connecting the midpoints of its top and bottom boundary.

We fix a rectangle i, and assume without loss of generality that the rectangles in $\mathcal{S}_i$ are vertical. Insert the middle segments of all horizontal rectangles in $\mathcal{H}(P)$ into the intersection reporting data structure. Then, for each rectangle $j \in \mathcal{S}_i$, we query its corresponding middle segment. By the definition of middle segments, each reported horizontal segment corresponds to a rectangle j' intersecting j. Since we remove each segment as we find it, no rectangle is reported twice. Repeating this for all $j \in \mathcal{S}_i$ finds the set $\mathcal{C}_i = \{\, j' \colon j' \sqcap j, j \in \mathcal{S}_i \,\}$ of all horizontal rectangles that intersect at least one rectangle in $\mathcal{S}_i$. Each query can be charged either to the horizontal segment that is deleted from the data structure or, in case $z = 0$, to the rectangle $j \in \mathcal{S}_i$ that we are querying. Hence, the total query time sums to $O(n \log n)$.

For each rectangle in the set $\mathcal{C}_i$, we should check the distance to every rectangle i' such that $i' \sqcap i$. Doing this explicitly takes $O(n^2)$ time. Thus, summing over all rectangles i, we get the total running time of $O(n^3)$.

To bring the running time down to $O(n^2 \log n)$, we create a reverse map of the map $i \mapsto \mathcal{C}_i$. For each rectangle k, we build a collection $\mathcal{L}_k$ that contains i if and only if k belongs to $\mathcal{C}_i$. Given a rectangle j', we need to check the distance between j' and i' for any (i, i') with $i \in \mathcal{L}_{j'}$ and $i \sqcap i'$. Using the intersection reporting data structure, we compute for each rectangle j' the set $\mathcal{D}_{j'}$, which is the set of all rectangles intersecting those in $\mathcal{L}_{j'}$. For each rectangle $i' \in \mathcal{D}_{j'}$, we test if $\Delta(i', j') = \hat{d} - 2$. Again recall that if we find a pair with $\hat{d}$, then the diameter must be $\hat{d} - 1$ (otherwise, the diameter is $\hat{d} - 2$). This proves Theorem 10.

5   Computation via Matrix Multiplication

In this section we provide an alternative method to compute the radius. This method also uses the condition in Theorem 8, but instead exploits the behavior of matrix multiplication on (0,1)-matrices. Recall that, given two (0,1)-matrices A and B, their product is $(AB)_{i,j} = \sum_k (A_{i,k} \cdot B_{k,j}) = |\{\, k \colon A_{i,k} = 1 \wedge B_{k,j} = 1 \,\}|$.

We define a (0,1)-matrix I, which is used to compute both the diameter and radius:

$$I_{i,j} = \begin{cases} 1 & \text{if } i \sqcap j, \\ 0 & \text{otherwise.} \end{cases}$$

In other words, for each pair i, j of rectangles in $\mathcal{H}(P) \cup \mathcal{V}(P)$, the matrix I indicates whether i and j intersect and have different orientations (one horizontal, one vertical). Note that, for ease of explanation, we have slightly abused the notation and identified rectangles of $\mathcal{H}(P) \cup \mathcal{V}(P)$ with indices in the matrix.

5.1 Computing the Radius

We use Theorem 8 to compute the radius. Thus, we need to determine if there exist four rectangles in $\mathcal{H}(P) \cup \mathcal{V}(P)$ that satisfy the condition of Theorem 8. If so, the radius will be $\hat{r} - 1$; otherwise, $\hat{r} - 2$. In order to do so, we define the (0,1)-matrix R that which indicates whether a pair of rectangles is at oriented distance at least $\hat{r}$ from each other:

$$R_{i,j} = \begin{cases} 1 & \text{if } \Delta(i,j) \geq \hat{r}, \\ 0 & \text{otherwise.} \end{cases}$$

By multiplying I and R, we obtain

$$(IR)_{i,j'} = |\{\, i' \colon (i \sqcap i') \wedge (\Delta(i',j') \geq \hat{r}) \,\}|.$$

In other words, the entry at (i, j') of the product IR counts the number of rectangles in $\mathcal{H}(P) \cup \mathcal{V}(P)$ that intersect rectangle i and are oriented differently from it, and at the same time are at oriented distance at least $\hat{r}$ from rectangle j'.

We construct the (0,1)-matrix N that indicates whether the corresponding entry of IR is non-zero, as follows:

$$N_{i,j} = \begin{cases} 1 & \text{if } (IR)_{i,j} > 0, \\ 0 & \text{otherwise.} \end{cases}$$

We now look at the product RN. Note that $(RN)_{i,i'} > 0$ if and only if there are two rectangles j and j' with $j \sqcap j'$ such that $\Delta(i,j) \geq \hat{r}$ and $\Delta(i',j') \geq \hat{r}$

The quantifier on j' and the condition on its intersection with j can be moved just to the right of the quantifier on j without altering the meaning of the formula, since both of them are existential quantifiers.

Therefore, the condition on Theorem 8 is satisfied if and only if for each 1-entry in I the corresponding entry in RN is non-zero. This condition can be checked by iterating over the entries of the matrices in quadratic time once the matrix RN has been computed.

Note that the time taken by the computation of the various matrix products dominates the time taken by the other loops and operations. Each matrix has $O(n)$ rows and columns, and the product of two $O(n) \times O(n)$ matrices can be computed in $O(n^\omega)$ time. A similar method can be applied using Theorem 7 to compute the diameter instead. We summarize the results of this section in the following theorem.

▶ **Theorem 12.** *The rectilinear link radius* $\mathrm{rad}(P)$ *or diameter* $\mathrm{diam}(P)$ *of a rectilinear polygonal domain P consisting of n vertices can be computed in $O(n^\omega)$ time.*

References

1 Hee-Kap Ahn, Luis Barba, Prosenjit Bose, Jean-Lou De Carufel, Matias Korman, and Eunjin Oh. A Linear-Time Algorithm for the Geodesic Center of a Simple Polygon. *Discrete & Computational Geometry*, 56(4):836–859, 2016. `doi:10.1007/s00454-016-9796-0`.

2 Sang Won Bae, Matias Korman, Joseph S. B. Mitchell, Yoshio Okamoto, Valentin Polishchuk, and Haitao Wang. Computing the L_1 Geodesic Diameter and Center of a Polygonal Domain. *Discrete & Computational Geometry*, 57(3):674–701, 2017. `doi:10.1007/s00454-016-9841-z`.

3 Sang Won Bae, Matias Korman, and Yoshio Okamoto. The Geodesic Diameter of Polygonal Domains. *Discrete & Computational Geometry*, 50(2):306–329, 2013. `doi:10.1007/s00454-013-9527-8`.

4 Sang Won Bae, Matias Korman, Yoshio Okamoto, and Haitao Wang. Computing the L_1 geodesic diameter and center of a simple polygon in linear time. *Computational Geometry: Theory and Applications*, 48(6):495–505, 2015. `doi:10.1016/j.comgeo.2015.02.005`.

5 Timothy M. Chan and Dimitrios Skrepetos. All-Pairs Shortest Paths in Geometric Intersection Graphs. In *Proc. 16th Algorithms and Data Structures Symposium*, pages 253–264, 2017.

6 Hristo Djidjev, Andrzej Lingas, and Jörg-Rüdiger Sack. An $O(n \log n)$ Algorithm for Computing the Link Center of a Simple Polygon. *Discrete & Computational Geometry*, 8:131–152, 1992. `doi:10.1007/BF02293040`.

7 Yoav Giyora and Haim Kaplan. Optimal dynamic vertical ray shooting in rectilinear planar subdivisions. *ACM Transactions on Algorithms*, 5(3):28:1–28:51, 2009. `doi:10.1145/1541885.1541889`.

8 John Hershberger and Subhash Suri. Matrix Searching with the Shortest-Path Metric. *SIAM Journal on Computing*, 26(6):1612–1634, 1997. `doi:10.1137/S0097539793253577`.

9 John Hershberger and Subhash Suri. An Optimal Algorithm for Euclidean Shortest Paths in the Plane. *SIAM Journal on Computing*, 28(6):2215–2256, 1999. `doi:10.1137/S0097539795289604`.

10 François Le Gall. Powers of tensors and fast matrix multiplication. In Katsusuke Nabeshima, Kosaku Nagasaka, Franz Winkler, and Ágnes Szántó, editors, *Proc. 25th International Symposium on Symbolic and Algebraic Computation, ISSAC*, pages 296–303. ACM, 2014. `doi:10.1145/2608628.2608664`.

11 Joseph S. B. Mitchell. An optimal algorithm for shortest rectilinear paths among obstacles. In Proc. *1st Canadian Conference on Computational Geometry, CCCG*, 1989.

12 Joseph S. B. Mitchell. L_1 shortest paths among polygonal obstacles in the plane. *Algorithmica*, 8(1):55–88, 1992.

13 Joseph S. B. Mitchell, Valentin Polishchuk, and Mikko Sysikaski. Minimum-link paths revisited. *Computational Geometry: Theory and Applications*, 47(6):651–667, 2014. `doi:10.1016/j.comgeo.2013.12.005`.

14 Joseph S. B. Mitchell, Valentin Polishchuk, Mikko Sysikaski, and Haitao Wang. An Optimal Algorithm for Minimum-Link Rectilinear Paths in Triangulated Rectilinear Domains. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Proc. Automata, Languages, and Programming - 42nd International Colloquium, ICALP*, volume 9134 of *Lecture Notes in Computer Science*, pages 947–959. Springer, 2015. `doi:10.1007/978-3-662-47672-7_77`.

15 Bengt J. Nilsson and Sven Schuierer. Computing the Rectilinear Link Diameter of a Polygon. In *Proc. Computational Geometry – Methods, Algorithms and Applications, International Workshop on Computational Geometry CG'91*, pages 203–215, 1991. `doi:10.1007/3-540-54891-2_15`.

16   Bengt J. Nilsson and Sven Schuierer. An Optimal Algorithm for the Rectilinear Link Center of a Rectilinear Polygon. *Computational Geometry: Theory and Applications*, 6:169–194, 1996. `doi:10.1016/0925-7721(95)00026-7`.

17   Subhash Suri. On some link distance problems in a simple polygon. *IEEE Transactions on Robotics and Automation*, 6(1):108–113, 1990. `doi:10.1109/70.88124`.

18   Haitao Wang. On the geodesic centers of polygonal domains. *Journal of Computational Geometry*, 9(1):131–190, 2018.