An Optimal Algorithm for the Geodesic L_1-Diameter and Center of a Simple Polygon*

Sven Schuierer†

Abstract

The diameter of a set S of points is the maximal distance between a pair of points in S. The center of S is the set of points that minimize the distance to their furthest neighbours. The problem of finding the diameter and center of a simple polygon for different distance measures has been studied extensively in recent years. $O(n \log n)$ algorithms have been given if the geodesic Euclidean or the link metric is used.

In this paper we consider the geodesic rectilinear case of this problem and give linear time algorithms to compute the L_1-diameter and the L_1-center of a simple rectilinear polygon which yields as a consequence a linear time solution for the geodesic L_1-radius problem.

1 Introduction

Rectilinear paths and rectilinear obstacles play an important role in many applications. VLSI design [Wid90] is one of the most prominent examples. A natural metric to measure the distance between two points in such a setting is the L_1-metric since it corresponds to the length of a shortest rectilinear path. In this paper we address the problem of locating the set of points inside a simple rectilinear polygon whose maximal L_1-distance, i.e., the length of a shortest rectilinear path, to any other point in the polygon is minimized. This set is known as the *center* of the polygon. A closely related problem is that of finding the maximal distance between two points inside a polygon, also known as the diameter problem. Note that we alway measure the distance between two points according to the length of a shortest rectilinear path inside the polygon.

Center and diameter problems have been considered in various other contexts and a number of results have been obtained. Pollack *et al.* [PSR89] give an $O(n \log n)$ algorithm to compute the geodesic center of a polygon if the Euclidean metric is used. It is not known up to date if the time complexity of this algorithm is optimal or whether it can be improved upon. The Euclidean diameter of a simple polygon can be computed in linear time by a result of Hershberger and Suri [HS93]. In this paper we show that the same time complexity can be achieved to compute the L_1-diameter and the L_1-center of a simple rectilinear polygon.

A metric with a somewhat different flavour is the *link metric* which counts the number of turns of a path. The problem of computing the *link center* has received a lot of attention [Ke89, DLS92, LPS*88] and again an $O(n \log n)$ upper bound could be achieved. As with the algorithm presented in this paper the upper can be improved if we restrict ourselves to rectilinear paths. Nilsson and Schuierer showed that in this case both the rectilinear link diameter and the rectilinear link center can be computed in linear time [NS91a, NS91b].

*This work was supported by the Deutsche Forschungsgemeinschaft under Grant No. Ot 64/8-1.

†Institut für Informatik, Universität Freiburg, Rheinstr. 10–12, D-79104 Freiburg, Fed. Rep. of Germany; email: schuierer@informatik.uni-freiburg.de

Most algorithms to compute the center of a polygon rely on the close relationship between the diameter and the *radius* of a polygon where the radius is defined as the distance between a point in the center and one of its furthest neighbours. For the link metric and the rectilinear link metric, for instance, it can be shown that the difference between the radius and half of the diameter is at most one link. In our problem we are even luckier since we can show that the radius equals half of the diameter. This is the key observation to computing the L_1-center of a polygon once the diameter and a pair of points that span the diameter have been computed.

The paper is organized as follows. Sections 2 and 3 introduce some notation and a number of basic observations. The main part of the paper is contained in Section 4 where it is shown how to employ a simple divide and conquer approach to compute the L_1-diameter. Section 5 then deals with the computation of the L_1-center.

2 Definitions

We start off with some definitions we need throughout this paper. We define a *simple rectilinear polygon* to be a Jordan curve consisting of axis-parallel line segments such that no two consecutive segments are collinear. Since we are mainly interested in the interior points, we also use the term polygon to denote the union of a polygon with its interior. In the following let $\mathbf{P}$ be a simple rectilinear polygon with n edges. A (*rectilinear*) *path* $\mathcal{P}$ is a curve that consists of a finite number of (axis parallel) line segments inside $\mathbf{P}$. The line segments of $\mathcal{P}$ will be called *links*. The L_1*-length* of $\mathcal{P}$, denoted by $\lambda_{L_1}(\mathcal{P})$, is defined as the sum of the lengths of its links. The (*rectilinear*) *link length* $\lambda_{rl}(\mathcal{P})$ of $\mathcal{P}$ is defined as the number of links $\mathcal{P}$ consists of. The L_1 *(link) distance* between two points p and q in $\mathbf{P}$ is the L_1-length (link length) of the shortest rectilinear path between p and q. We will only consider rectilinear polygons and rectilinear paths in $\mathbf{P}$. Hence, whenever we talk of polygons, we mean simple rectilinear polygons and whenever we talk of paths, we mean rectilinear paths in $\mathbf{P}$.

In the rest of the paper the letter v with various subscripts and superscripts will always denote vertices of $\mathbf{P}$. In the same way the letters p, q, and r denote points of $\mathbf{P}$.

The fundamental structure upon which the results of this paper are based is the *histogram partition of* $\mathbf{P}$ w.r.t. a chord c which was first introduced by Levcopolous [Lev87]. But before we are in a position to state its definition, we need to take a look at histograms and some related concepts. A *histogram* is a monotone rectilinear polygon with one of the monotone chains forming a single straight line segment. The single straight line segment forming one of the monotone chains is called the *base* of the histogram. A *horizontal (vertical)* histogram is a histogram with a horizontal (vertical) base. An *upper* histogram is a horizontal histogram with its interior above the base. *Lower, left*, and *right* histograms are defined similarly. A *pyramid* is a histogram that is monotone w.r.t. both the x- and y-axis.

A *chord* of $\mathbf{P}$ is a line segment in $\mathbf{P}$ that intersects the boundary of $\mathbf{P}$ in exactly two points. Note that every chord divides $\mathbf{P}$ into two subpolygons. If we are given a horizontal chord c, we define the *maximal upper (lower) histogram* $\mathbf{H}$ of c in $\mathbf{P}$ to be the maximum area histogram interior to $\mathbf{P}$ above (below) c. For illustration refer to Figure 1a. Maximal left or right histograms of a vertical chord are defined similarly. Note that a chord has at most two maximal histograms.

A *window* is a segment of the boundary of a maximal histogram which is not also part of the boundary of $\mathbf{P}$. We do not consider the base to be a window of the histogram. Note that every window is a chord of $\mathbf{P}$ and, thus, splits $\mathbf{P}$ into two parts.

The histogram partition of $\mathbf{P}$ w.r.t. to a chord c can now be defined as follows. The *histogram partition* $\mathcal{H}(\mathbf{P}, c)$ of $\mathbf{P}$ w.r.t. c is based on an inductive construction. Let $\mathbf{H_1}$ and $\mathbf{H_2}$ be the

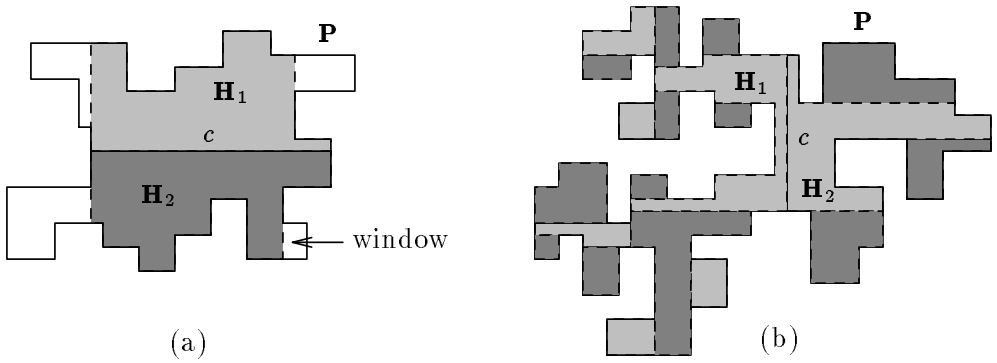

Figure 1: *The definition of maximal histograms and a histogram partition.*

two maximal histograms of c in $\mathbf{P}$. If $\mathbf{P} = \mathbf{H_1} \cup \mathbf{H_2}$, then $\mathcal{H}(\mathbf{P},c)$ is defined to be $\{\mathbf{H_1},\mathbf{H_2}\}$. Otherwise, let w_i, $1 \leq i \leq k$, be the windows of $\mathbf{H_1}$ and $\mathbf{H_2}$. Each window w_i is a chord in $\mathbf{P}$ that splits $\mathbf{P}$ into two subpolygons one of which does not contain $\mathbf{H_1}$ and $\mathbf{H_2}$. We denote this subpolygon by $\mathbf{P}_{w_i}$. The histogram partition is now given by $\mathcal{H}(\mathbf{P},c) = \bigcup_{1\leq i\leq k} \mathcal{H}(\mathbf{P}_{w_i}, w_i) \cup \{\mathbf{H_1},\mathbf{H_2}\}$. For illustration see Figure 1b. If a triangulation of the polygon is given (see [Cha90]), a histogram partition can be computed in linear time [Lev87].

In the following we give precise definitions of the L_1-diameter and -center of a polygon $\mathbf{P}$. The L_1-*diameter* of $\mathbf{P}$, denoted by $D(\mathbf{P})$, is defined as the maximum distance between two points in $\mathbf{P}$. Closely related is the notion of the radius $R(\mathbf{P})$ of $\mathbf{P}$ which is defined as the smallest value r such that there exists a point p in $\mathbf{P}$ with $L_1(p,q) \leq r$, for all points q of $\mathbf{P}$. The set of points p in $\mathbf{P}$ that satisfy $L_1(p,q) \leq R(\mathbf{P})$, for all $q \in \mathbf{P}$, is called the L_1-*center* of $\mathbf{P}$.

3 Chords and the L_1-distance in Polygons

The algorithms we present subsequently to compute the L_1-center and diameter of a simple polygon are mainly based on a divide and conquer approach that employs *balanced chords*, i.e., axis-parallel line segments that cut a polygon into two parts of almost equal complexity. So before we go into the algorithmic details, we need to investigate the distance relationsship between points and chords in $\mathbf{P}$. In the following let c be an axis-parallel chord in $\mathbf{P}$, $\mathcal{H}(\mathbf{P},c)$ the histogram partition of $\mathbf{P}$ w.r.t. c, and p a point in $\mathbf{P}$.

The first observation we make is that there is a *unique* point q on c such that $L_1(p,q) = L_1(p,c)$. We state this as a lemma.

Lemma 3.1 *For each point p in $\mathbf{P}$ and any axis-parallel chord c, there is one unique point on c that minimizes the distance from p to c.*

Proof: W.l.o.g. assume that c is vertical. We prove the claim by induction on the rectilinear link distance from p to c. If the link distance from p to c is one, then, clearly, the horizontal line segment from p to c is the (only) shortest path and the horizontal projection c_p of p onto c is the only point that minimizes the distance from p to c. If p has a link distance of $k > 1$, then there is a window w of one of the two maximal histograms of c such that p belongs to the part $\mathbf{P}_w$ of $\mathbf{P}$ that is cut off by w. Obviously, the rectilinear link distance from p to w is $k-1$. Let $\mathcal{P}$ be a shortest path from c to p. Since $\mathcal{P}$ crosses w, there is a first intersection point q of $\mathcal{P}$ with w. Consider the orthogonal projection c_q of q onto c. Clearly, c_q is the point on c that is

closest to q. Furthermore, c_q is independent of the location of q on w; hence, all shortest paths from c to p start in c_q and we have $L_1(p, c_q) = L_1(p, c)$. □

In the following we call the unique point on c that is closest to p the *projection of p on c* and denote it by $\pi_c(p)$. The importance of the projection of a point p onto a chord c is that we can easily compute the distance of all the other points of c to p as the next lemma shows.

Lemma 3.2 *If c is an axis-parallel chord in $\mathbf{P}$ and p a point in $\mathbf{P}$, then we have, for every point q on c, $L_1(p,q) = L_1(p, \pi_c(p)) + L_1(\pi_c(p), q)$.*

Proof: Again we assume w.l.o.g. that c is vertical. Consider the histogram partition $\mathcal{H}(\mathbf{P}, c)$ of $\mathbf{P}$ w.r.t. c. We prove the claim by induction on the rectilinear link distance from p to c. If the link distance is one, then the claim is obvious. So let the link distance of p be $k > 1$ and q be some point on c. Let $\mathcal{P}$ be a shortest path from p to q. There is one window w of the two maximal histograms of c such that $p \in \mathbf{P}_w$ and the link distance of w to p is $k-1$ or less. Hence, $\mathcal{P}$ intersects w. Let q_w be the intersection point of $\mathcal{P}$ and w. By the induction hypothesis we have $L_1(p, q_w) = L_1(p, \pi_w(p)) + L_1(\pi_w(p), q_w)$. Hence, we obtain

$$\begin{aligned} L_1(p,q) &= L_1(p,q_w) + L_1(q_w,q) \\ &= L_1(p,\pi_w(p)) + L_1(\pi_w(p),q_w) + L_1(q_w,q) \\ &= L_1(p,\pi_w(p)) + L_1(\pi_w(p),q_w) + L_1(q_w,\pi_c(q_w)) + L_1(\pi_c(q_w),q) \end{aligned}$$

Furthermore, since q_w lies on the window that cuts off the subpolygon p belongs to, we have $\pi_c(p) = \pi_c(q_w)$ as in the proof of Lemma 3.1. Thus, we have

$$\begin{aligned} L_1(p,q) &= L_1(p,w) + L_1(\pi_w(p),q_w) + L_1(q_w,\pi_c(p)) + L_1(\pi_c(p),q) \\ &\geq L_1(p,\pi_c(p)) + L_1(\pi_c(p),q) \end{aligned}$$

as claimed. □

An immediate consequence of Lemma 3.2 is that the furthest neighbour of a point in $\mathbf{P}$ is always a vertex. More precisely, we can show the following result.

Corollary 3.3 *For each point p in $\mathbf{P}$, all of its furthest neighbours are convex vertices of $\mathbf{P}$.*

Proof: Assume this is not the case. Let p be some point in $\mathbf{P}$ and q one of its furthest neighbours that is not a convex vertex. Then, there is one axis-parallel chord c through q that extends to both sides of q. Lemma 3.2 yields that one of the end points of c is farther away from p than q. □

An immediate consequence of the above lemma is that the diameter is always spanned by a pair of convex vertices.

3.1 Computing Projections

If we are given a chord c in $\mathbf{P}$, the diameter and center algorithms presented subsequently require us to compute the projection of all vertices of $\mathbf{P}$ onto c. The next lemma provides the main tool that allows us to do this in linear time.

Lemma 3.4 *Let c be an axis-parallel line segment in $\mathbf{P}$ and p a point in $\mathbf{P}$. If the link distance of p to c is greater than 1, then there exists a vertex v of $\mathbf{P}$ such that*

$$L_1(p,c) = L_1(p,v) + L_1(v,c).$$

Furthermore, v is one of the end points of the base of the histogram of $\mathcal{H}(\mathbf{P}, c)$ p belongs to.

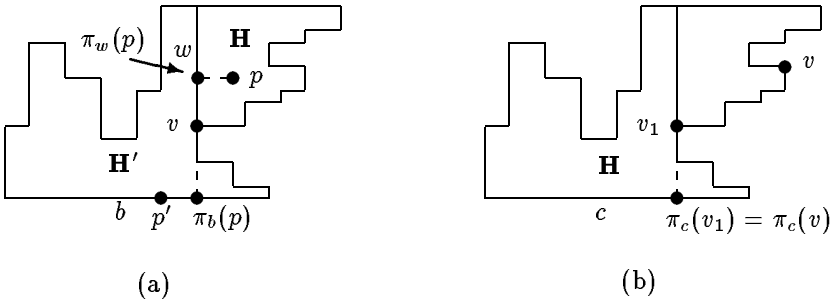

Figure 2: *The computation of projections.*

Proof: Consider the histogram $\mathbf{H}$ of $\mathcal{H}(\mathbf{P},c)$ p belongs to. Let w be the base of $\mathbf{H}$. There is one histogram $\mathbf{H}'$ in $\mathcal{H}(\mathbf{P},c)$ such that w is a window of $\mathbf{H}'$. Let v be the end point of w that is closer to the base b of $\mathbf{H}'$. Again we use induction on the link distance from p to c. So first assume that p is two links away from c. Then, $b = c$ and $\mathbf{H}'$ is a maximal histogram of c. Since any shortest path from p to c has to cross w, we have $L_1(p,c) = L_1(p,\pi_w(p)) + L_1(\pi_w(p),c)$. By our choice of v we have $L_1(\pi_w(p),c) = L_1(\pi_w(p),v) + L_1(v,c)$ which proves the claim for $k = 2$ (see Figure 2a).

Now assume that the link distance from p to c is $k > 2$. Let $\mathcal{P}$ be a shortest path from p to c and assume that $\mathcal{P}$ crosses b in a point p'. Since p' is on a shortest path from p to c we have $\pi_c(p) = \pi_c(p')$. As we just proved before we have

$$L_1(p,b) \quad = \quad L_1(p,v) + L_1(v,b).$$

Lemma 3.2 yields

$$L_1(p,p') \quad = \quad L_1(p,\pi_b(p)) + L_1(\pi_b(p),p').$$

Together with $\pi_b(p) = \pi_b(v)$ we obtain

$$\begin{aligned} L_1(p,c) &= L_1(p,p') + L_1(p',c) \\ &= L_1(p,\pi_b(p)) + L_1(\pi_b(p),p') + L_1(p',c) \\ &= L_1(p,v) + L_1(v,b) + L_1(\pi_b(p),p') + L_1(p',c) \\ &\geq L_1(p,v) + L_1(v,c) \end{aligned}$$

The reverse inequality follows by the triangle inequality which proves the claim. □

If c is an axis-parallel chord in $\mathbf{P}$, the above lemma can be employed for a linear time algorithm to compute $\pi_c(v)$, for each vertex v of $\mathbf{P}$, together with the distance $L_1(v,c)$ in the following way. We first compute the histogram decomposition $\mathcal{H}(\mathbf{P},c)$ of $\mathbf{P}$ and compute the points $\pi_c(v)$, for all the vertices v that are contained in one of the maximal histograms of c. Let $\mathbf{H}$ be one of these histograms. For each window w of $\mathbf{H}$, let $\mathbf{H}_w$ be the histogram of $\mathcal{H}(\mathbf{P},c)$ that has its base on w. One of its end point, say v_1, satisfies Lemma 3.4 and, hence, by computing $L_1(v_1,v)$, for all vertices v of $\mathbf{H}_w$, we obtain the distance from v to c and $\pi_c(v)$ which equals $\pi_c(v_1)$ in time proportional to the number of vertices of $\mathbf{H}_w$. For illustration refer to Figure 2b. Continuing recursively with the windows of $\mathbf{H}_w$ in this way, we can compute the projection and distance of each vertex to c as claimed. Since the complexity of a histogram decomposition is linear, the whole algorithm takes only linear time.

4 A Simple Algorithm for the L_1-Diameter of a Polygon

We first concern ourselves with the problem of computing the L_1-diameter of $\mathbf{P}$. Computing the diameter is also a first step to solving the L_1-center problem since the middle point on a path between