On-Line Search in a Simple Polygon *

Jon M. Kleinberg†

Abstract

We consider a number of search and exploration problems, from the perspective of robot navigation in a simple polygon. These problems are "on-line" in the sense that the robot does not have access to the map of the polygon; it must make decisions as it proceeds, based only on what it has seen so far. For the problem of exploring a simple rectilinear polygon (under the L_1 norm), Deng, Kameda, and Papadimitriou give a 2-competitive deterministic algorithm; we present a randomized exploration algorithm which is 5/4-competitive. Using similar techniques, we are able to give an algorithm for searching an arbitrary, unknown rectilinear polygon. No constant competitive ratio is attainable in this case, but our algorithm is within a constant factor of optimal in the worst case; in a sense, it is a generalization of some of the strategies of Baeza-Yates, Culberson, and Rawlins to a much more general class of search spaces. Finally, we examine a type of polygon for which competitive search is possible — the class of "streets" considered by Klein, who gave a $1 + \frac{3}{2}\pi$-competitive algorithm for the search problem in this case. We present a simple algorithm with a competitive ratio of at most $\sqrt{4+\sqrt{8}}$ (~ 2.61); in rectilinear streets it achieves the optimal competitive ratio of $\sqrt{2}$.

*This work was supported in part by the Sloan Fellowship and NSF PYI award of Éva Tardos. Author is supported by an ONR Graduate Fellowship

†Laboratory for Computer Science, MIT, Cambridge MA 02139 USA.

1 Introduction

A natural problem in robotics is that of searching for a goal in an unknown polygonal region. For example, a robot with vision is placed at a starting point in a simple polygon, and it must traverse a path to some *target point* in the polygon. Both the location of the target and and the geometry of the polygon are unknown, but the robot will recognize the target when it sees it. (Throughout this paper, we use the standard definition of visibility: a robot at x can *see* y if the segment $\overline{xy}$ is completely contained in the polygon P.)

Hence the problem is "on-line," in the sense that decisions must be made based only on what the robot has seen so far. The distance it travels, however, is compared to that of an *adversary* who knows both the map of the polygon and the location of the target, and hence can plan an optimal route. In one sense, such problems are as old as algorithmic maze-solving questions that reach back a century and more (see e.g. [O]). More recently, problems in this spirit have been addressed by Blum and Kozen in the context of automata theory [BK], and by Lumelsky, Stepanov, and others in robotics [LS]; they were first incorporated into the framework of on-line algorithms (cf. [ST, MMS]) by Papadimitriou and Yannakakis in [PY]. Since then, a number of papers have analyzed algorithms for navigation in an unknown environment in terms of their *competitive ratio*: the worst-case ratio of the distance traveled by the algorithm to the distance that an optimal algorithm with a map of the scene would have traveled. An algorithm whose competitive ratio is bounded by c can be termed c-competitive for the given problem; it is said simply to be *competitive* if it is c-competitive for some constant c.

Most of the work on the geometric version of the navigation problem has focused on finding shortest paths in an obstacle-filled environment [BRS, BBFY], and exploring a geometric scene [DKP]. Search algorithms, on the other hand, have generally been developed for situations in which the geometry is kept to a minimum — for example, the case of searching for a point on one of m rays emanating from a point [BCR, KRT], a line in the plane, or a point in an integer lattice [BCR]. Moreover, the structure of the space to be searched is assumed to be known — only the location of the target is unknown. [BCR] writes, "...these problems are (very simple) models of searching in the real-world. It is very often the case that we do not know many of the parameters that are usually taken for granted in designing search algorithms."

In this paper, we provide an on-line algorithm for the general problem of searching for a target point at an *unknown* location in an *unknown* and arbitrary simple rectilinear polygon. Thus, the robot must adapt its search pattern as it sees more and more of the polygon. For a given rectilinear polygon P, we identify the number of *essential cuts* of P [CN, DKP] as a fundamental parameter in determining the best competitive ratio attainable in searching P — it is easy to cast the problem of searching m concurrent rays as a search problem in a polygon with m essential cuts, whence the [BCR] lower bound of $2em - o(m)$ on competitive ratio applies (e is being used to denote the base of the natural logarithm here). Thus, the natural question is whether there is an algorithm which is $O(m)$-competitive for the problem of searching an *arbitrary* simple rectilinear polygon with m essential cuts. In Section 4, we present such an algorithm.

The algorithm is based on the problem — first considered in [DKP] — of exploring a simple rectilinear polygon P, starting from and returning to a fixed point s in P; we use an exploration algorithm iteratively as the search proceeds. We show in Section 3 how an adaptation of the technique in [DKP] gives a randomized algorithm which is 5/4-competitive in the L_1 norm when s is any point inside P.

The paper begins, however, with a self-contained section in which we look at an intriguing notion — restricting the class of polygons and target points in such a way that a constant competitive ratio can be achieved for the search problem. The papers [IK, Kl] introduce the term *street* to define a class of general (not necessarily rectilinear) polygons with two distinguished points s and t, such that the two st boundary chains are mutually weakly visible (see below for an elaboration on this definition). In [Kl], Klein gives a $1 + \frac{3}{2}\pi$-competitive (~ 5.71-competitive) algorithm for finding t from s in an unknown street P. However, the algorithm and its analysis are quite involved. In Section 2, we give a simple algorithm with a competitive ratio of at most $\sqrt{4+\sqrt{8}}$ (~ 2.61). Moreover when P is rectilinear, it achieves the optimal ratio of $\sqrt{2}$. We believe it would be interesting in general to find other natural classes of polygons that can be searched competitively.

Finally, a word about the distance metrics used in this paper. The distance between two points as measured in the L_1 and L_2 (Euclidean) metrics differs at most by a factor of $\sqrt{2}$; thus, an algorithm which is c-competitive in L_1 is $c\sqrt{2}$-competitive in L_2. With the exception of the algorithm for traversing streets, which is analyzed directly in the Euclidean metric, we present our results in the conceptually neater framework of the L_1 metric. In view of the tight correspondence between L_1 and L_2, our final search algorithm is $O(m)$-competitive in both.

2 Traversing an Unknown Street

Let P be a simple polygon and s and t two distinguished points on the boundary. The removal of s and t would disconnect the boundary into two polygonal chains, L and R. We say that P is a *street* [IK, Kl] if each point on the boundary of P can see some point on the opposite boundary chain. The goal is for a robot with vision to travel from s to t; neither the map of P nor the coordinates of t are known. The cost incurred by the robot is the length of the path it generates, and its competitive ratio is taken with respect to the length of the shortest s-t path in P; distances are measured in the Euclidean metric.

Figure 1(a) can be completed to form a rectilinear street in which t could be just around the corner from either X or Y [Kl]. The robot will incur the best worst-case performance if it moves directly to segment $\overline{XY}$, then to t (it will see t when it reaches XY). This can be at most a factor of $\sqrt{2}$ longer than the shortest path. Curiously, this is the only known lower bound on the competitive ratio achievable for the problem. In [Kl], an algorithm with a competitive ratio of at most $1 + \frac{3}{2}\pi$ (~ 5.71) is presented. Below, we give an algorithm with competitive ratio at most $\sqrt{4+\sqrt{8}}$ (~ 2.61).

The example of Figure 1(a) is central to the proof technique we develop in this section, and it highlights a principle that will appear repeatedly in what follows — on-line algorithms hate making decisions. Specifically, it is useful in many navigation problems to adopt a strategy that preserves the robot's options for as long as possible. In the figure, suppose the robot

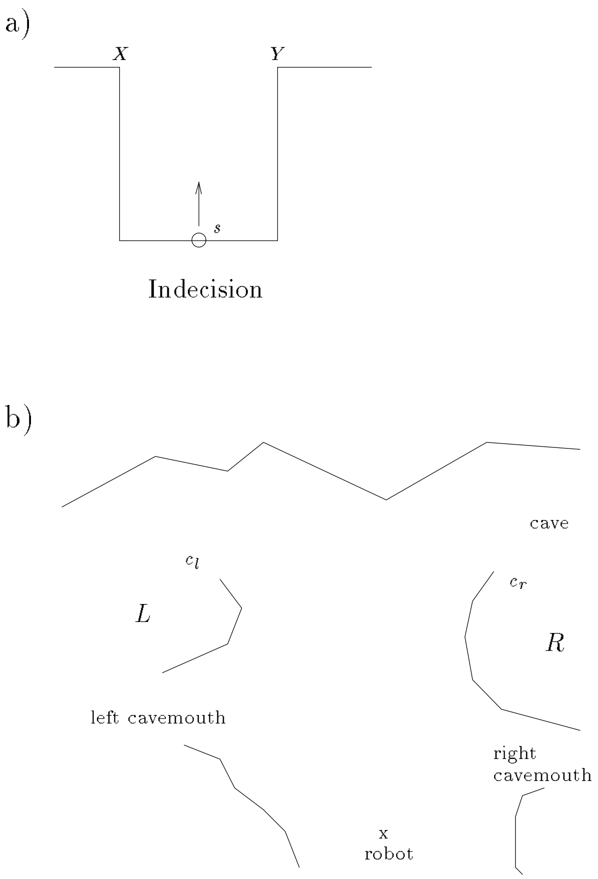


Figure 1: Streets and their views

at point s is moving towards segment $\overline{XY}$, but it has not yet decided whether it ultimately wants to visit point X or Y. Let us define a polygonal path to be *monotone* if the x- and y-coordinates of the points on the path change from their initial to final values monotonically. Then the key observation is that in the L_1 metric, any monotone path between two points is a shortest path, so the robot can defer its decision (X or Y) until it reaches segment $\overline{XY}$ and still have the option of traveling optimally to either point. The related fact for the Euclidean metric is that any monotone path between two points in the plane has length at most a factor of $\sqrt{2}$ times the straight-line distance between them. Thus, in this example, the robot can move to segment $\overline{XY}$ before making a decision and travel only $\sqrt{2}$ times too far (Euclidean distance) in the worst case.

Let P be a street, and assume that the robot is currently located at a point x inside P. The robot maintains an *extended view* of P; this consists of all points on the boundary of P that it has seen so far. The robot's extended view will typically look like the example of Figure 1(b). We define a *cave* C to be a connected chain of the boundary of P such that the robot has seen the endpoints of the chain but no other points of it. At some point p' on the robot's path, these two endpoints were on the same line of sight from p'; call the one closer to p' the "mouth" of C. Each cavemouth v is a reflex vertex of P; in the neighborhood of v, P lies either to the left or right of the ray $p'v$. We accordingly refer to v as being either a left or right cavemouth.

Assume that t has not yet been seen, and the robot has maintained the invariant that the points in its extended view immediately to its left and right belong to L and R respectively. In view of these assumptions, we assemble some facts about extended views of a street before presenting the algorithm itself. The first is a standard fact about shortest paths inside any simple polygon.

Lemma 1 *If t is contained in a cave C, then the shortest x-t path touches the mouth of C.*

Lemma 2 *Let p be a point on boundary chain L (resp. R), and let Ψ be the boundary chain sp of P contained in L (R). If the robot moves from s to p, it will have seen every point on Ψ.*

Proof. Every point on boundary chain Ψ must be able to see some point on R; but all such lines of sight to R cross the robot's path from s to p. Thus the robot has seen every point on Ψ. ∎

Lemma 3 *If v is a left (right) cavemouth, it belongs to boundary chain L (R).*

Proof. Assume v is a left cavemouth, $v \in R$, and v was seen from point p'. The chain determined by a clockwise scan of the boundary from v to s (taken from point p') is entirely contained in R. Thus, if the robot were to walk directly from p' to v, it would have seen all of this chain, by Lemma 2. But since v is a cavemouth, it would not have seen any point on the boundary of P just around the corner from v, which belongs to this chain; this is a contradiction. ∎

Corollary 1 *In the extended view, all left cavemouths lie to the left of all right cavemouths.*

If the extended view contains any left cavemouths, we define c_l to be the rightmost one. The point c_r is defined analogously for right cavemouths. If both c_l and c_r are defined, the chain between the far endpoints of their respective caves must be completely visible in the extended view; otherwise, it would contain an additional left or right cavemouth. Combining this with Lemma 1 and the fact that t has not been seen,

Lemma 4 *The point t lies in the cave of either c_l or c_r. Consequently, the shortest path from x to t touches either c_l or c_r.*

Let $d(\cdot, \cdot)$ denote the length of the shortest path between two points in P. The shortest path from s to t, denoted by Γ, is a chain of line segments joined at reflex vertices of P. There is a natural order on the vertices of Γ, determined by traversing it from s to t. Our algorithm works iteratively, allowing the robot to move from a given vertex of Γ to a later one, with small "detour." The following theorem provides the main inductive step.

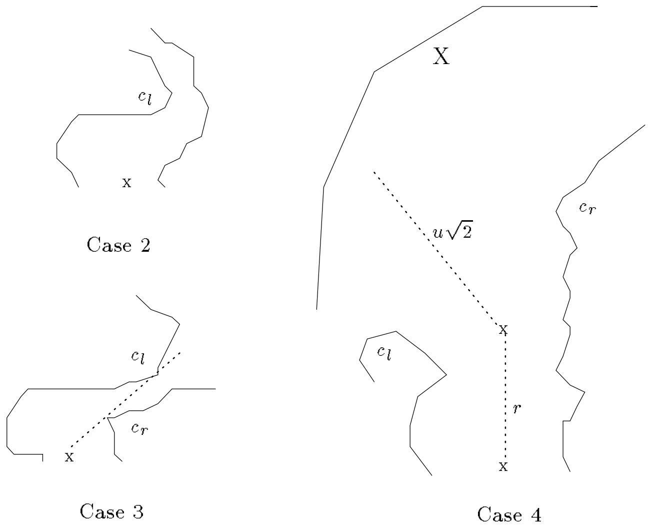


Figure 2: The algorithm at work

Theorem 1 *Assume that the robot is currently located at a vertex $x \in \Gamma$. Then it can move to a later vertex $x' \in \Gamma$ while traveling at most $(\sqrt{4+\sqrt{8}})d(x,x')$.*

Proof. We present an (on-line) algorithm for doing this. Based on the robot's extended view, there are four cases to consider. (See Figure 2.)

Case 1. If t is visible, the robot moves directly to t. The distance traveled is $d(x,t)$.

Case 2. If c_r (c_l) is not defined (there are no right (left) cavemouths), then by Lemma 4, Γ passes directly through c_l. Thus, the robot moves directly to c_l, following Γ the whole way.

Otherwise, both c_l and c_r are visible. The robot chooses a direction of motion such that c_l lies to its left and c_r lies to its right. We view this as a coordinate system in which the robot is the origin and it is moving in the direction of the positive y-axis; thus, c_l has negative x-coordinate and c_r has positive x-coordinate. The robot moves in this direction, updating its extended view and the points c_l, c_r, until one of of the above two cases applies, or one of the following two:

Case 3. The point c_l (or respectively c_r) "jumps" to the opposite side of the y-axis. At the moment when this happens, both c_l and c_r will lie on the same line of sight. If the robot moves in this direction until it hits the nearer one, it will once again be on a point $x' \in \Gamma$, having followed a path from x to x' that was monotone with respect to the chosen coordinate system. Thus, it has traveled no more than $\sqrt{2}$ times the distance from x to x' along Γ.

Case 4. If none of Cases 1, 2, or 3 applies, then there comes a point at which the robot's line of sight to c_l (c_r) is parallel to the x-axis. At the moment when this happens, the robot is "confused"; it can no longer follow a path guaranteed to be monotone to either c_l or c_r. However, the robot can adopt the following approach to return to the shortest path.

Let us translate the coordinate system so that the robot is again at the origin (so c_l now lies on the negative x-axis). Since t has not yet been seen, and since the second quadrant (i.e. $\{x \leq 0,\ y \geq 0\}$) is free of cavemouths, the robot can see all of the contiguous boundary chain X lying in this quadrant. Thus, X must be entirely contained in L or R; the robot can return to Γ once it discovers which of these cases holds. Define L_X to be the portion of boundary chain between c_l and the endpoint of X lying on the negative x-axis; define R_X to be the portion of boundary chain between c_r and the endpoint of X lying on the positive y-axis.

The robot begins moving in the direction of the vector $(-1, 1)$, updating its extended view and the points c_l, c_r, until it sees t or one of the following events occurs:

1. c_l has the same x-coordinate as the robot, or c_r has the same y-coordinate as the robot.

2. One of the chains L_X or R_X becomes completely visible.

If event (2) occurs first, then the robot will be able to move to the opposite cavemouth, thereby returning to Γ. Suppose event (1) occurs first; assume for the sake of concreteness that c_l has the same x-coordinate as the robot (the other case is strictly analogous). Then a point just around the corner from c_l can only see points lying on $X \cup L_X$; this implies that $X \subset R$, and so the robot can return to Γ by moving to c_l.

We must now bound how far the robot travels by implementing this strategy. Let r denote the distance traveled by the robot prior to becoming confused. First consider the case in which event (1) occurs, and c_r has the same y-coordinate as the robot. Suppose that the coordinates of c_r are (v, u); then the robot travels $r + u(1+\sqrt{2}) + v$, while we have $d(x, x') \geq \sqrt{(r+u)^2 + v^2}$. Thus, the worst-case ratio incurred by the robot is bounded by

$$\frac{r + u(1+\sqrt{2}) + v}{\sqrt{(r+u)^2 + v^2}}.$$

A somewhat lengthy but straightforward argument shows that this expression attains its maximum when $r = 0$ and $u = v(1+\sqrt{2})$, with a value of $\sqrt{4+\sqrt{8}}$.

Now suppose event (1) occurs, and c_l has the same x-coordinate as the robot. If the coordinates of c_l are $(-u, -v)$, then the robot travels $r + u(1+\sqrt{2}) + v$. Assume that when the robot first became confused, the coordinates of c_l were $(-u_1, 0)$ (recall that this point lies on the negative x-axis). Let $u_2 = u - u_1$; then we have

$$d(x, x') \geq \sqrt{r^2 + u_1^2} + \sqrt{v^2 + u_2^2}$$

In view of the simple inequality $\frac{a+b}{c+d} \leq \max(\frac{a}{c}, \frac{b}{d})$, the worst-case ratio is upper-bounded by the maximum value of

$$\frac{r + u_1(1+\sqrt{2})}{\sqrt{r^2 + u_1^2}}.$$

As above, this is maximized by taking $u_1 = r(1+\sqrt{2})$, with a value of $\sqrt{4+\sqrt{8}}$.

The case in which event (2) occurs first is similar. Suppose that all of L_X becomes visible and the goal has not been seen (the other case is analogous). Then we can set x' to be the current value of c_r. Suppose that the coordinates of c_r are (v, u); then since event (1) did not occur, the robot's y-coordinate is no more than u. Thus it can get to c_r having traveled at most $r + u(1+\sqrt{2}) + v$, while again $d(x, x') \geq \sqrt{(r+u)^2 + v^2}$.

Finally, we should note that it is possible for the robot's motion to be stopped by the boundary of P. As in Lemma 2, however, it will have seen the entirety of one of the caves associated with c_l or c_r by the time this happens, so it can return to Γ; the preceding analysis is not affected. ■

Corollary 2 *For any street P, repeatedly applying the above algorithm produces a path from s to t that is at most $\sqrt{4+\sqrt{8}}$ times as long as the shortest s-t path in P.*

In the case in which P is rectilinear, the above algorithm can be implemented so that the robot's direction of motion is always along one of the coordinate axes. Then the key observation is that Case 4 cannot occur in a rectilinear street (since all angles are right angles, either a point just around the corner from c_l could not see R, or a point just around the corner from c_r could not see L). Thus we can show

Theorem 2 *When P is rectilinear, the above algorithm is $\sqrt{2}$-competitive, and this is optimal.*

3 Exploring a Simple Rectilinear Polygon

First, we present some basic definitions of [CN, DKP] on the structure of rectilinear polygons. In this and the remaining sections, distances will be measured in the L_1 metric.

Let P be a simple rectilinear polygon and s a distinguished point in P. An edge e of P is contained in a line ℓ; we say that an *extended edge* is a line segment $\hat{e} \subset \ell$ in P which shares one endpoint with e, and whose other endpoint is also on the boundary of P. Each edge e induces at most two extended edges. Also, note that an extended edge is either horizontal or vertical, depending on the orientation of its associated edge e. Call $\hat{e}$ a *horizon* if there is no path from s to e that does not cross $\hat{e}$. We can define a partial order on horizons as follows: if h and h' are horizons, then $h \preceq h'$ (h dominates h') if any path from s to h' must cross h, or if $h = h'$. The horizons of P which are maximal are called *essential cuts*; for an essential cut h, it is possible to start at s and follow a path which crosses every horizon except h. See Figure 3.

A special case of the following lemma is given in [DKP]; it is the underlying reason for the success of "greedy" exploration algorithms in rectilinear polygons.

Lemma 5 *Let $\alpha_1, \ldots, \alpha_n$ be a set of horizontal and vertical segments in P, and v a point in P. The (L_1) shortest path beginning at v and touching the α_i in order is generated by the greedy algorithm, which, from segment α_j, always chooses the shortest path to α_{j+1}.*

The problem we are considering here is that of traversing an *exploration route* in P: a path, starting and ending at s, such that every point of P can be seen from some point on the path. The robot does not know the map of P in advance. As noted in [CN, DKP] (see also the discussion above), a closed path through s has this property if and only if it touches all essential cuts in P. In [CN, DKP], it is observed that since any exploration route can

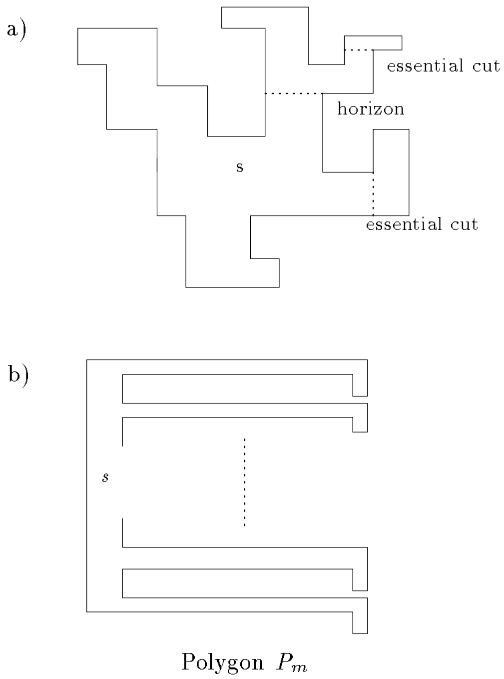


Figure 3: Some simple rectilinear polygons

be traversed (off-line) without self-crossings, the shortest exploration route will touch the essential cuts in clockwise order. Moreover, by Lemma 5, it will touch the cuts in this order using the greedy algorithm.

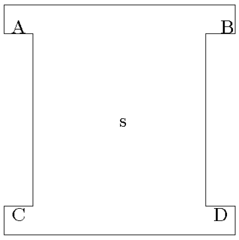
Consider the case in which the point s lies on the boundary of P, between the endpoints of essential cuts e and e'. The on-line algorithm given in [DKP] is essentially a greedy strategy which crosses each upcoming horizon as quickly as possible; it is shown in [DKP] that it will traverse the greedy path which touches the essential cuts in clockwise order, beginning with e. Consequently, this algorithm finds the *optimal* exploration route on-line; it is 1-competitive when s lies on the boundary of P.

When s does not lie on the boundary, the choice of which essential cut to start with becomes crucial, and the robot does not have enough information to make this choice.

Proposition 1 *No deterministic algorithm for exploring a simple rectilinear polygon can be better than* $5/4$*-competitive.*

Proof. Consider Figure 4. All the long edges of the polygon P have length 2, the short edges have some length ε much less than 2, and s is at the center. A robot exploring P crosses either the upper or lower horizon first; assume the former case. At this point, it will see two tiny "caves" at points A and B, both of which must be visited. Assume that it visits A before visiting B or crossing the lower horizon (other cases are similar).

We now add an extra cave at C but not at D. Even if the robot now had the map of P, it would have to travel a distance of 8 to visit the caves at B and C and return to s. It



Add small "caves" at
three of the four points
A, B, C, D

Figure 4: Lower bound construction

has traveled a distance of 2 to reach A; thus its total distance is 10. On the other hand, the greedy exploration route which visits C first travels a distance of 8. ∎

The algorithm given in [DKP] is 2-competitive when s is an arbitrary starting point in P, and this is the best known deterministic ratio. In the remainder of this section, we give a simple randomized algorithm whose performance matches the deterministic lower bound of Proposition 1.

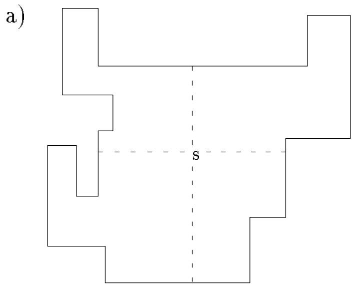
Theorem 3 *There is 5/4-competitive randomized exploration algorithm when s is an arbitrary starting point in P.*

Proof. Consider first the following construction. If the robot standing at s were to imagine a thin "needle" of boundary extending from the real boundary of P to s, it would then be on the boundary of this new polygon and could explore optimally. If we restrict ourselves to horizontal or vertical segments, then there are four possible needles that can be inserted in P. See Figure 5.

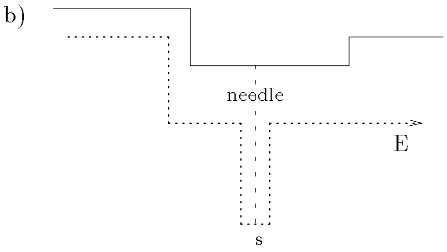
Let $L(\cdot)$ denote the length of a path in P, E denote the optimal exploration route in P, and P_i denote polygon P with the i^{th} needle inserted, $i = 1, 2, 3, 4$. Finally, we denote by T_i the (optimal) exploration route generated by the robot starting from s in P_i (s is on the boundary of each P_i). Of course, we are not really interested in the performance of T_i in P_i; we must show that T_i is also not far from optimal in the original polygon P. Set $d_i = L(T_i) - L(E)$.

We claim that $\sum_{i=1}^{4} d_i \leq L(E)$; the proof is as follows. Since E visits the essential cuts of P in clockwise order, it meets each needle in at most one point. E can be traversed so as to avoid the i^{th} needle (it takes a detour through s); let us denote this longer route by E_i. Since E_i is an exploration route for P_i and T_i is optimal in this polygon, we have $L(T_i) \leq L(E_i)$.

Let $d'_i = L(E_i) - L(E)$. Consider the four points at which E hits the needles (some of these points may be s); connect these by shortest paths to form a closed path $\tilde{T}$. $\tilde{T}$ is not necessarily an exploration route for P, but we have $L(\tilde{T}) \leq L(E)$ and $L(\tilde{T}) = \sum_{i=1}^{4} d'_i$. Since



Inserting needles in the polygon



The exploration route E avoiding a needle

Figure 5: Polygon with needles

$d_i \le d'_i$ for each i,

$$\sum_{i=1}^{4} d_i \le \sum_{i=1}^{4} d'_i \le L(E).$$

Thus, if the robot chooses any needle, the exploration route T_i it generates will have length at most $2L(E)$ (and simple examples show that, standing at s, there is no way to choose a needle guaranteeing a performance better than this). However, the expected value of the quantity d_i is bounded by $L(E)/4$, so if the robot chooses one of the four needles uniformly at random, the expected length of the exploration route it generates is at most $\frac{5}{4}L(E)$. ∎

4 Searching an Unknown Rectilinear Polygon

Using the exploration algorithm of the preceding section, we can give an $O(m)$-competitive algorithm for the general search problem described in the introduction. A point t is hidden in a simple rectilinear polygon P; however, the point t can be recognized when it is first seen. A robot starting at s, and without knowledge of the map of P, must travel to t; its distance traveled is compared to $d(s,t)$, the length of the shortest s-t path. As mentioned in the introduction, $O(m)$ is the best bound possible on the competitive ratio: if we take the polygon P_m of Figure 3, make the m "arms" extremely long, and introduce tiny bends to limit visibility, then the [BCR] lower bound for searching m concurrent rays applies — no deterministic algorithm can be better than $(2em - o(m))$-competitive. (An $\Omega(m)$ lower bound clearly holds for randomized algorithms as well, with a somewhat smaller constant.)

We first assemble some basic lemmas that will be useful in analyzing the exploration algorithm. Because P is a simple polygon, we have the following fact.

Lemma 6 *Let h and h' be horizons with the same orientation such that for some point v in P, every path from s to v must cross both h and h'. Then h and h' are comparable with respect to $\preceq$ ($h \preceq h'$ or $h' \preceq h$).*

Based on this lemma, we can represent the partial order $\preceq$ restricted to the horizontal segments by a directed tree T_h in which the root is the point s and the other nodes are horizontal horizons. For vertical horizons, there is the analogous representation as a tree T_v.

Lemma 7 *Each of T_h and T_v has at most m leaves.*

Proof. A leaf of T_h corresponds either to a horizontal essential cut, or a horizon h that dominates some vertical essential cut h'. In the latter case, no other leaf of T_h can dominate h', by Lemma 6, so h can be uniquely charged to h'. The same analysis holds for T_v, giving us the stated bound. ∎

Lemma 8 *Let v be a point in P such that s cannot see v. Then there is some horizon h in P that separates s from v, such that for any path from s to h, the point v can be seen from some point on this path. (I.e. no matter how the robot gets to h, it will have seen v.)*

Proof. By analogy with the construction for streets, consider the "view" of P from point v. Define a cave, as before, to be a connected chain of the boundary of P such that v can see the endpoints but no other points of the chain. Since s cannot see v, s must lie in some cave C. Let u be the cavemouth of C; then there is a horizon with endpoint u such that by the time the robot reaches this horizon, it will have crossed the ray $\vec{vu}$ and seen v. ∎

Lemma 9 *Suppose the robot is moving perpendicularly towards an extended edge e, and no other extended edge of the same orientation lies between the robot and e. Then the robot can see both endpoints of e.*

Proof. Suppose the robot could not see the right endpoint of e; let x be the rightmost point on e that it can see. The line of sight from the robot to x must meet a reflex vertex of P; it is easily verified that this is the endpoint of a parallel extended edge e' lying between the robot and e. ∎

Finally, the following lemma is the key to designing the search algorithm; it is also an interesting fact in itself. As before, let $d(u, v)$ denote the length of an L_1 shortest u-v path in P.

Lemma 10 *Let P be a simple polygon (not necessarily rectilinear), and consider a robot traversing some path in P. If points u and v are both visible from this path, then the robot can determine $d(u, v)$ without seeing the rest of P.*

Proof. In fact, it can compute a shortest path between u and v. Since we are dealing with the L_1 metric, the shortest u-v path in P will not generally be unique. However, some shortest u-v path is polygonal (consists of a finite number of line segments). Consider the extended view of P that the robot has generated. By definition, the line segment joining the two endpoints of a cave in this view is completely contained in P; let us call it a "pseudo-edge." Let P' denote the truncated polygon whose boundary consists of the edges and pseudo-edges of the extended view of P. Thus, the robot has seen all of the boundary of P'.

We claim that there is a shortest u-v path in P that does not leave P'. The result will follow since the robot can obviously compute a shortest u-v path in P'. Let T be a polygonal shortest u-v path in P, which may enter a cave of the extended view, crossing pseudo-edge e at the point x. Since v lies in P', the path must re-enter P'; let us say that it next does so by crossing pseudo-edge e' at point y. Since P is a simple polygon, $e = e'$. Thus we can form a new u-v path which goes directly from x to y when T enters this cave; this operation does not increase the length. Proceeding in this way, we eliminate the (finitely many) places at which T enters a cave, producing a u-v path T' in P' with $L(T') \leq L(T)$. ∎

Theorem 4 *There is an algorithm for searching simple rectilinear polygons, which is $O(m)$-competitive on the class of such polygons with m essential cuts.*

Proof. For any $\delta > 0$, let P^δ denote the rectilinear polygon obtained by "truncating" P at horizons that are more than δ away from s. There is some small δ_0 such that all points within δ_0 of s are visible from s. By rescaling, we can assume that $\delta_0 = 1$. For fixed values of δ, the robot will simulate the algorithm of [DKP] in P^δ, as follows. The [DKP] algorithm has the property that at all times, the robot is moving along the coordinate axes, perpendicularly towards an extended edge, and it maintains this direction until it reaches the closest such extended edge e. By Lemma 9, it can see both of the endpoints of the edge e; thus, by Lemma 10, it can tell whether e lies in P^δ or not. If e lies in P^δ, the robot moves towards it, as in the [DKP] algorithm; otherwise, e is treated as a wall in the simulation. In this way, the exploration route generated by the simulation is the same as the route that the [DKP] algorithm generates in P^δ.

Initially, δ is set to 1. Whenever t is first seen, the robot moves directly to it; if the robot explores P^δ without seeing t, then δ is doubled and the next iteration begins.

Recall the partial orders T_h and T_v on the horizontal and vertical horizons of P. In P^δ, these naturally restrict to partial orders T_h^δ and T_v^δ, which are obtained simply by deleting all nodes corresponding to horizons more than δ away from s. Both T_h^δ and T_v^δ are still directed trees rooted at s, and by Lemma 7, each has at most m leaves. Since each essential cut of P^δ is a leaf of one of these trees, P^δ has at most $2m$ essential cuts. Lemma 8 implies that one way to explore P^δ would be to travel to each essential cut individually; this would require a total distance of $2m(2\delta) = 4m\delta$. Since the robot is following a 2-competitive algorithm, it travels at most $8m\delta$ in exploring P^δ.

Given this, we can complete the proof as follows. If the robot sees t when $\delta = 1$, then it can travel to it optimally. Otherwise, assume it first sees t in P^δ, with $\delta = 2^k$. Since it did not see t when exploring $P^{(2^{k-1})}$, we have by Lemma 8 that $d(s,t) \geq 2^{k-1}$.

Now, we must bound how far the robot has traveled before reaching t. Until it sees t, it has traveled at most

$$\sum_{j=0}^{k} 8m2^j \leq 8m(2^{k+1}) \leq 32md(s,t).$$

Since $\delta = 2^k \leq 2d(s,t)$, it is easily verified that traveling to t can require no more than an additional $3d(s,t)$. Thus the robot's path is $O(m)$ times $d(s,t)$, as required. ∎

5 Conclusion and Open Problems

We have considered the problem of exploration and search in simple polygons, ultimately giving a general search algorithm for rectilinear polygons which is within a constant factor of optimal in the worst case. Some immediate questions are to narrow the gaps between the upper and lower bounds given here. Any progress on the deterministic lower bound or randomized upper bound for exploration would be particularly interesting, since it would show that randomization is strictly more powerful for this problem. We are also interested in the more general problem of defining natural classes of polygons, such as streets, that can be searched competitively.

6 Acknowledgements

The author wishes to thank Éva Tardos for numerous helpful discussions and ideas on the topic of this paper. Thanks also to the robotics and vision groups at Cornell for their feedback, and to Rolf Klein for pointing out an error in an earlier version of the street algorithm.

References

[BBFY] E. Bar-Eli, P. Berman, A. Fiat, P. Yan, "On-Line Navigation in a Room," *Proc. 3rd ACM-SIAM Symposium on Discrete Algorithms*, 1992, pp. 237–249.

[BCR] R. Baeza-Yates, J. Culberson, G. Rawlins, "Searching in the Plane," *Inf. and Computation*, 106(1993), pp. 234–252.

[BRS] A. Blum, P. Raghavan, B. Schieber, "Navigating in Unfamiliar Geometric Terrain," *Proc. 23rd ACM Symposium on Theory of Computing*, 1991.

[BK] M. Blum, D. Kozen, "On the Power of the Compass (or, Why Mazes are Easier to Search than Graphs)," *Proc. 19th IEEE Symposium on Foundations of Computer Science*, 1978, pp. 132–142.

[CN] W. Chin, S. Ntafos, "Shortest Watchman Routes in Simple Polygons," *Discrete and Computational Geometry*, 6(1991), pp. 9–31.

[DKP] X. Deng, T. Kameda, C. Papadimitriou, "How to Learn an Unknown Environment I: The Rectilinear Case," Technical Report CS-93-04, Department of Computer Science, York University. (Preliminary version in *Proc. 32nd IEEE Symposium on Foundations of Computer Science*, 1991, pp. 298–303.)

[IK] C. Icking, R. Klein, "The Two Guards Problem," *Proc. 7th ACM Symposium on Computational Geometry.*

[Kl] R. Klein, "Walking an Unknown Street with Bounded Detour," *Computational Geometry: Theory and Applications*, 1(1992), pp. 325–351. (Preliminary version in *Proc. 32nd IEEE Symposium on Foundations of Computer Science*, 1991, pp. 304–313.)

[KRT] M. Kao, J. Reif, S. Tate, "Searching in an Unknown Environment: An Optimal Randomized Algorithm for the Cow-Path Problem," *Proc. 4th ACM-SIAM Symposium on Discrete Algorithms*, 1993, pp. 441–447.

[LS] V. Lumelsky, A. Stepanov, "Path-Planning Strategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape," *Algorithmica*, 2(1987) pp. 403–430.

[MMS] M. Manasse, L. McGeoch, D. Sleator, "Competitive Algorithms for Server Problems," *J. Algorithms*, 11(1990), pp.208–230.

[O] O. Ore *Theory of Graphs*, Providence: AMS, 1962.

[PY] C. Papadimitriou, M. Yannakakis, "Shortest Paths Without a Map," *Theoretical Computer Science*, 84(1991), pp. 127–150.

[ST] D. Sleator, R. Tarjan, "Amortized Efficiency of List Update and Paging Rules," *Comm. ACM*, 23(1985).