Algorithms for On-Line Navigation*

Jon M. Kleinberg

TR 92-1316
November 1992

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*This work was sponsored in part by the Sloan Fellowship of Eva Tardos.
Algorithms for On-Line Navigation *

Jon M. Kleinberg

Department of Computer Science
Cornell University, Ithaca, NY 14853, U.S.A.

Abstract

We consider a number of problems faced by a robot trying to navigate inside a simple polygon. Such 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. Specifically, we examine algorithms for the related problems of exploration and search. We present a 5/4-competitive randomized algorithm for exploring a rectilinear polygon; the only previous work here is the deterministic 2-competitive algorithm claimed in Deng, Kameda, and Papadimitriou. For the problem of searching for a distinguished point in a polygon, we give a $\sqrt{3}$-competitive algorithm for traversing a street, which improves on a result of Klein by more than a factor of 3. Finally, the techniques we use in exploration and the construction of search patterns are combined to give an algorithm for searching an arbitrary and unknown rectilinear polygon; here, no constant competitive ratio can be achieved, but our algorithm is within a constant factor of optimal in the worst case.

---

*This work was supported in part by the Sloan Fellowship of Éva Tardos.
1. Introduction

A recurring theme in robotics and other branches of computer science is the need to design an autonomous algorithm that can interact with its environment. Very often, a major goal in such problems is to write an algorithm with as little "hard-wired" knowledge as possible; it should be able to deal with a wide range of environments as it encounters them. We cannot say that the topic of navigation in an unknown geometric scene ("on-line navigation") has grown out of this theme, in that some of the questions it considers — maze-solving, for example — reach back a hundred years or more [Ore]. But problems in on-line navigation provide us with a clear illustration of the issues faced in the design of such autonomous algorithms.

Navigation problems had been addressed by Blum and Kozen in the context of automata theory [BK], and by Lumelsky, Stepanov, and others in robotics [LS]. The incorporation of robot navigation into the framework of on-line algorithms (cf. [ST, MMS]), however, was first proposed 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. Navigation here is viewed as a geometric optimization problem — the input (the set of obstacles in the environment) is revealed only gradually, as the robot sees (or touches) it, and the cost incurred by the robot is the distance it travels. An algorithm whose competitive ratio is bounded by \(c\) can be termed \(c\)-competitive for the given problem. Thus in the problem originally considered in [PY], that of finding a short path between given points \(A\) and \(B\) amidst an unknown set of obstacles, the competitive ratio of an algorithm is the worst-case ratio of the length of the \(A-B\) path it generates to the length of the shortest obstacle-avoiding \(A-B\) path. Naturally, the best competitive ratio achievable depends on the class of obstacles considered; variants of this problem have been considered in [BRS, BBFY].

Here, we investigate two other kinds of navigation problems: exploration and search in a simple polygon. The model in these problems is as follows. A robot with vision is placed in a simple polygon \(P\). From its current position \(s\) it can see any point \(x\) inside or on the boundary of \(P\) such that the line segment from \(s\) to \(x\) is completely contained in \(P\). We assume here that it has unlimited memory; as it moves, it can build a "map" of \(P\), consisting of all the points on the boundary of \(P\) that it has seen. The full map of \(P\) is simply the circular list of the coordinates of its vertices.

Deng, Kameda, and Papadimitriou introduced the problem of exploring a simple polygon in [DKP]. A robot must traverse a closed path inside \(P\), beginning and ending at a given point \(s\), such that it sees all points on the boundary of \(P\); this is an on-line version of the "Shortest Watchman Route" problem of Chin and Ntafos [CN]. They claim a 2-competitive deterministic algorithm for exploring a simple rectilinear polygon (distances measured in the \(L_1\) metric). We give a \(5/4\)-competitive randomized algorithm for this problem, and show that no deterministic algorithm can achieve a ratio better than \(5/4\).

It is less clear how to define the problem of searching a simple polygon \(P\). The idea is for a robot with vision to begin at some point \(s\) in \(P\) and travel to a point \(t\) in \(P\). The location of \(t\) is unknown, but the robot will recognize it when it first sees it. There are two flavors of this problem, one of
them more “on-line” than the other, depending on whether the map of $P$ is known or not. In the style of search problem pioneered by Baeza-Yates, Culberson, and Rawlins [BCR], a robot is faced with a known geometric environment and must find a distinguished goal point that is “hidden” in the environment; its competitive ratio is the worst-case ratio of the distance it travels to the length of the shortest path from start to goal. Thus, the problem becomes to construct a search pattern in the environment that sees all points at each given distance as quickly as possible (in what follows, we make no distinction between the search pattern and the corresponding algorithm to traverse it). The techniques of [BCR] were developed for fairly simple environments such as a set of $m$ rays diverging from a common point; they have proven, however, to have application in a wide variety of on-line navigation problems. (See also [KRT], in which a randomized algorithm for the problem of searching $m$ concurrent rays is presented.)

In this paper, we consider the question of search patterns in an arbitrary simple rectilinear polygon. We identify the notion of an essential cut introduced in [CN] as the feature of a simple polygon that makes search difficult; certain points lying beyond an essential cut cannot be seen unless the cut is crossed. Indeed, a simple construction gives a class of rectilinear polygons $P_1, P_2, \ldots$, such that $P_m$ has $m$ essential cuts and no algorithm has competitive ratio better than $\Omega(m)$ for the problem of searching for a distinguished point in $P_m$. We then show that for any simple rectilinear polygon $P$ with $m$ essential cuts, an $O(m)$-competitive search pattern can be constructed for the problem of searching for a goal inside $P$.

Very little work has been done on the problem of search when both the environment and the location of the goal are unknown, and it seems difficult to give situations in which a constant competitive ratio is achievable. One example is the algorithm of Klein [Kl] for traversing a special kind of polygon known as a street. We present an algorithm for this problem which improves on the competitive ratio given in [Kl] by more than a factor of 3. Finally, we show how to combine the notions used in exploration and the construction of search patterns to give an algorithm for a robot searching for a point in an unknown rectilinear polygon. That is, the robot must again construct a search pattern, but here it must truly do so in an on-line fashion — it does not have access to the map of $P$. If $P$ has $m$ essential cuts, the competitive ratio of the algorithm is again $O(m)$; this shows that knowledge of $P$ does not help, in an asymptotic sense, for the problem of searching for a point inside it.

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 with a competitive ratio of $c$ in $L_1$ has a competitive ratio at most $c\sqrt{2}$ 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 search algorithms are $O(m)$-competitive in both.

The paper is organized as follows. The algorithm for traversing a street is presented in Section 2. Section 3 develops a number of useful facts about rectilinear polygons and uses them to prove the existence of an $O(m)$-competitive search pattern in any simple rectilinear polygon. Section 4 presents our results on exploration, and Section 5 combines these earlier ideas to present the $O(m)$-competitive algorithm for searching an unknown rectilinear polygon. Finally, Section 6 concludes
with open problems and possible extensions of this work.

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.

![Diagram](image)

Figure 1: Streets and their views

Figure 1(a) can be completed to form a 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 $XY$, 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 $1 + \frac{3}{2}\pi$ ($\sim 5.72$) is presented. Below, we give an algorithm with competitive ratio at most $\sqrt{3}$ ($\sim 1.73$).

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 at point $s$ is moving towards segment $XY$, 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 $XY$ 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 $XY$ 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 $C$. In the neighborhood of a cavemouth $v$, $P$ 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 before presenting the algorithm itself. The first is a standard fact about shortest paths inside a 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 $L$ ($R$), and let $\Psi$ 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 $\Psi$.

*Proof.* Every point on boundary chain $\Psi$ must see 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 $\Psi$. ■

**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, a contradiction. ■

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

If the extended view contains any left cavemouts, we define $c_l$ to be the rightmost one. The point $c_r$ is defined analogously for right cavemouts. 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, we have
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 $\Gamma$, is a chain of line segments joined at reflex vertices of $P$. Assume inductively that the robot is currently sitting on a vertex $x \in \Gamma$; our algorithm shows how it can get to another vertex $x' \in \Gamma$ while traveling at most $\sqrt{3}d(x, x')$. Based on the robot’s extended view, there are four cases to consider. (See Figure 2.)

![Diagram](image)

Figure 2: The algorithm at work

**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, $\Gamma$ passes directly through $c_l$. Thus, the robot moves directly to $c_l$, following $\Gamma$ 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 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 is monotone with respect to the chosen coordinates. Thus, it has traveled no more than $\sqrt{2}$ times the distance from $x$ to $x'$ along $\Gamma$.

**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”;

5
it can no longer follow a path guaranteed to be monotone to either \( c_l \) or \( c_r \). However, since \( P \) is a street, if the robot touches the segment \( c_l c_r \), it will see the entirety of at least one of their respective caves, and thus will be able to return to \( \Gamma \) by moving to the other cavemouth.

Thus, the problem in Case 4 becomes the following. Assume that the robot is currently at \( y \). It must choose a “landing point” \( z \) on the segment \( c_l c_r \) so as to minimize the relative distance it travels in returning to \( \Gamma \). If we set \( r = d(x, y), s = d(y, z), t = d(z, c_l), t' = d(z, c_r) \), and let \( d_l \) and \( d_r \) denote the shortest-path distances from \( x \) to \( c_l \) and \( c_r \) respectively, then the robot wants to choose \( z \) so as to minimize

\[
\max\left(\frac{r + s + t}{d_l}, \frac{r + s + t'}{d_r}\right).
\]

Based on what it can see from the point \( y \), it has enough information to make this calculation; when it finally arrives at \( c_l \) or \( c_r \), it will once again be sitting on \( \Gamma \).

It is difficult to give the precise worst-case value of this ratio over all \( r, s, t \in \mathbb{R}^+ \), but we can put a fairly tight upper bound on it as follows. Suppose that the robot follows the weaker strategy of always moving along the normal to segment \( c_l c_r \). Then we have \( d_l \geq \sqrt{r^2 + s^2 + t^2} \) and \( d_r \geq \sqrt{(r + s)^2 + t'^2} \), so an easy calculation shows that both of the ratios in the expression above are bounded by \( \sqrt{3} \).

Finally, we should note that it is possible for the robot’s motion in the positive \( y \) direction to be stopped by the boundary of \( P \). By 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 \( \Gamma \); the preceding analysis is not affected. Thus we have

**Theorem 1** For any street \( P \), the above algorithm produces a path from \( s \) to \( t \) that is at most \( \sqrt{3} \) times as long as the shortest \( s \)-\( t \) path in \( P \).

---

**3. Search Patterns in a Rectilinear Polygon**

We now turn to the question of search patterns in rectilinear polygons. As in described in the introduction, our robot is given the map of a simple rectilinear polygon \( P \) and a point \( s \) in \( P \); it must find a distinguished point \( t \) in \( P \). The goal is to construct a search pattern \( T \) in \( P \) so that by traversing \( T \) it will follow an \( s \)-\( t \) path which is as short as possible relative to \( d(s, t) \), regardless of the location of \( t \). In this and the remaining sections, distances will be measured in the \( L_1 \) metric.

There are a number of definitions based on those in [CN, DKP] that will be useful here. Let \( P \) be a simple rectilinear polygon and \( e \) an edge of \( P \). The edge \( e \) is contained in a line \( \ell \); 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 \( e \). Define \( L_{\hat{e}} \) and \( R_{\hat{e}} \) to be the two boundary chains of \( P \) formed by removing the endpoints of \( \hat{e} \). Varying slightly from the terminology of [CN], we say that a horizontal extended edge \( \hat{e} \) is an *essential cut* if \( L_{\hat{e}} \subseteq L_f \) for all horizontal extended edges \( f \), or \( R_{\hat{e}} \subseteq R_f \) for all horizontal \( f \). We make the analogous definition for vertical extended edges.
Assume that a starting point \( s \) inside \( P \) is given. Then we will say that an extended edge \( \hat{e} \) induced by \( e \) is a horizon if there is no path from \( s \) to \( e \) that does not cross \( \hat{e} \). See Figure 3. We can define a partial order on horizons as follows: if \( h \) and \( h' \) are horizons, then \( h \leq h' \) (\( h \) dominates \( h' \)) if any path from \( s \) to \( h' \) must cross \( h \). Because \( P \) is a simple polygon, we have the following useful lemma.

**Lemma 5** 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 \( \leq \) (\( h \leq h' \) or \( h' \leq h \)).

![Figure 3: Some simple rectilinear polygons](image)

Based on the above lemma, we can represent the partial order \( \leq \) 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 \( T_v \).

Assume that the polygon \( P \) has \( m \) essential cuts. We will say that \( s \) lies “beyond” an essential cut if it is on the side of the cut that is dominated by all other horizons of the same orientation. Clearly \( s \) can be beyond at most one horizontal and one vertical essential cut; the rest of the essential cuts appear as leaves of \( T_h \) and \( T_v \). Conversely, a leaf of \( T_h \) corresponds to a horizontal essential cut, or a horizon \( h \) that dominates some vertical horizon \( h' \). In the latter case, no other leaf of \( T_h \) can dominate \( h' \), by Lemma 5, so \( h \) can be uniquely charged to \( h' \). The same analysis holds for \( T_v \), giving us

**Lemma 6** The total number of leaves of \( T_h \) and \( T_v \) is between \( m - 2 \) and \( 2m \).

If \( e \) is an edge of \( P \) and \( \hat{e} \) is a horizon, a point on the interior of \( e \) can only be seen once the robot touches \( \hat{e} \). As a result, we have the straightforward example of Figure 3(b), in which polygon \( P_m \) contains \( m \) long “arms,” each ending in an essential segment. For any given search pattern, we can place the goal beyond the essential cut it crosses last. A similar argument holds for randomized search algorithms, so we have
Proposition 1  No algorithm (deterministic or randomized) for searching $P_m$ can be better than $\Omega(m)$-competitive.

We note that the definition of a street can be applied here to give an equivalent measure of the complexity of searching $P$. Say that a street decomposition of $P$ is a representation of $P$ as $\bigcup_{i=1}^{k} Q_i$, where each $Q_i$ is a rectilinear street all of whose vertices lie on the boundary of $P$. Let $k(P)$ denote the minimal value of $k$, taken over all possible street decompositions of $P$. We can show that if $P$ has $m$ essential cuts, then $k(P)$ is $\Omega(m)$, and clearly the polygons $P_m$ have decompositions of size $\Theta(m)$. Thus, our upper and lower bounds on searching can be phrased equivalently in terms of $k(P)$.

The following three lemmas provide a basis for constructing a search pattern that tries to cross each horizon in $P$ as quickly as possible. The first — a key property of the $L_1$ metric — and a special case of the second are given in [DKP].

**Lemma 7**  Let $v$ be a point in $P$ and $\alpha$ a horizontal or vertical line segment in $P$. Assume that a shortest path from $v$ to $\alpha$ ends at the point $u \in \alpha$. Then for any other $u' \in \alpha$, $d(v, u') = d(v, u) + d(u, u')$.

**Lemma 8**  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 $\alpha_i$ in order is generated by the greedy algorithm, which, from segment $\alpha_j$, always chooses the shortest path to $\alpha_{j+1}$.

**Proof.** Let $S$ denote the greedy algorithm, and $A$ any other algorithm. Each algorithm defines a “hitting point” $S(i), A(i)$ on segment $\alpha_i$, where it first touches this segment. Between these points, it clearly should follow an $L_1$ shortest path. Set $S(0) = A(0) = v$, and $s_i = d(S(i), S(i+1))$, $a_i = d(A(i), A(i+1))$, $d_i = d(S(i), A(i))$. Then by Lemma 7 and the definition of $S$, $s_i + a_i + d_{i+1} \leq s_i + a_i$, so $s_i \leq a_i + (d_i - d_{i+1})$. The total distance traveled by $S$ is $\sum_i s_i$, which is less than or equal to $\sum_i (a_i + d_i - d_{i+1}) = -d_n + \sum_i a_i$. \(\square\)

**Lemma 9**  Let $v$ be a point in $P$ such that $v$ 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 of the previous section, consider the “view” of $P$ from point $v$. Define a cave, as above, 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 $vu$ and seen $v$. \(\square\)

We now inductively construct a tree $T'_h$ consisting of polygonal chains in $P$ as follows. The root is the point $s$. When $T'_h$ crosses a horizon $h \in T_h$ with out-edges $(h, h_1), \ldots, (h, h_k)$, we extend $T'_h$ via shortest paths to the horizons $h_1, \ldots, h_k$. An analogous construction yields a tree $T'_v$. These trees have the same root; we view them as a single tree $T$ of polygonal chains.
Lemma 10 \( T \) contains at most \( 2m \) leaves, and a shortest path from \( s \) to each horizon in \( P \).

Proof. The first part of the claim follows by the fact that the leaves of \( T \) correspond to leaves of \( T_h \) and \( T_v \). Consider the path in \( T_h' \) (\( T_v' \)) to a horizontal (vertical) horizon \( h \). It crosses the horizons separating \( s \) from \( h \) in a greedy manner; since any path must cross these horizons, it is a shortest path by Lemma 8.

The search algorithm now traverses \( T \) in a style similar to the algorithms of [BCR]. Initially, the robot can see all points in \( P \) that lie within some fixed distance of \( s \); we assume without loss of generality that this distance is 1. The algorithm operates in phases. In phase \( j \), the robot follows all polygonal chains of \( T \) in a depth-first manner out to a distance of \( 2^j \). If at any point it sees the goal \( t \), it moves directly to it.

Theorem 2 The above algorithm is \( O(m) \)-competitive for the problem of searching \( P \).

Proof. Assume that the robot first sees \( t \) in phase \( j > 1 \) (the case \( j = 1 \) is similar), and let \( h \) be a horizon as in the statement of Lemma 9. Let \( \delta \) denote the distance from \( t \) to the point at which the shortest \( s \)-\( h \) path in \( T \) crosses \( h \). Then since \( t \) was not seen in phase \( j - 1 \), \( d(s, h) \geq 2^{j-1} \), and so \( d(s, t) \geq 2^{j-1} + \delta \), by Lemma 8. The robot travels a distance at most \( 2m(2^j) \) in phase \( i \), so it travels at most \( \sum_{i=1}^{j} 2m(2^i) \leq 2^{j+2}m \) to reach \( h \), then \( \delta \) to reach \( t \). Thus, the total distance traveled is at most \( \delta + 2^{j+2}m \leq (8m)d(s, t) \).

4. Exploring a Rectilinear Polygon

In this section, we consider the problem of a robot which must explore a simple rectilinear polygon \( P \), starting and ending at some point \( s \) in \( P \). That is, it must traverse a closed path beginning and ending at \( s \) such that every point on the boundary of \( P \) is visible from some point on the path. As noted in [CN, DKP] (see also the discussion in the preceding section), a closed path through \( s \) has this property if and only if it touches all essential cuts in \( P \). For simplicity of presentation, we assume that \( s \) does not lie beyond any essential cut. In [CN, DKP], it is observed that since any exploration route can be traversed (off-line) without self-crossings, the shortest exploration route will touch the essential cuts in clockwise order. Moreover (see also Lemma 8), 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] will traverse the greedy path that 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 2 No deterministic algorithm for exploring a simple rectilinear polygon can be better than \( 5/4 \)-competitive.
Figure 4: Lower bound constructions

Proof. Consider Figure 4(a). All the long edges of the polygon $P$ have length 2, the short edges have some length $\varepsilon$ much less than 2, and $s$ is at the center. A robot exploring $P$ must cross 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 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.

When $s$ is an interior point of $P$, it is suggested in [DKP] to “take any one of the four directions along the coordinates from $s$ as the initial direction of motion until the boundary is hit,” and then follow an algorithm similar to the one for $s$ on the boundary. However, this algorithm does not have any constant competitive ratio; consider Figure 4(b), in which the optimal exploration route has length 2 and the boundary is as far away as we want.

In the remainder of this section, we present a deterministic 2-competitive algorithm, and adapt this to give a 5/4-competitive randomized algorithm. 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 $d_i = L(T_i) - L(E)$. 
Lemma 11 $\sum_{i=1}^{4} d_i \leq L(E)$.

Proof. 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 $\tilde{T}$. Then we have $L(\tilde{T}) \leq L(E)$ and $L(\tilde{T}) = \sum_{i=1}^{4} d_i$. Since $d_i \leq d_i'$ for each $i$,

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

Thus, if the robot chooses any needle, the exploration route $T_i$ it generates will have length at most $2L(E)$. 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 $5/4L(E)$. Thus we have

Theorem 3 The given deterministic and randomized exploration algorithms have competitive ratios of 2 and $5/4$ respectively.

5. Searching an Unknown Rectilinear Polygon

Finally, we combine ideas from the preceding two sections to give an $O(m)$-competitive algorithm for searching a simple rectilinear polygon with $m$ essential cuts, even when the map of the polygon is not known in advance. This generalizes the result on search patterns from Section 3; note that the $\Omega(m)$ lower bound on competitive ratio clearly holds here as well. Specifically, the problem is
for a robot, starting at a point $s$ in a simple rectilinear polygon $P$ for which it does not have the map, to find a point $t$ while traveling as little as possible.

There are two lemmas that will be useful in what follows; the first is an interesting fact in its own right. As before, let $d(u,v)$ denote the length of the shortest $u$-$v$ path in $P$.

**Lemma 12** 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). We define the robot’s extended view of $P$ by analogy with Section 2. 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 $L(T') \leq L(T)$.

**Lemma 13** Let $P$ be a simple rectilinear polygon with a distinguished point $s$ in its interior, fix some subset of the horizons with respect to $s$, and let $P'$ be the truncated polygon formed by removing the parts of $P$ beyond these horizons. Then the number of essential cuts of $P'$ is at most the number of essential cuts of $P$.

**Proof.** In effect, we are pruning the trees $T_h$ and $T_v$. That is, any essential cut in $P'$ dominates some essential cut $c$ of $P$ with the same orientation, and by Lemma 5, no other essential cut of $P'$ with this orientation can dominate $c$.

As before, imagination is our robot’s most powerful tool. The idea is to successively form polygons $P^6$, which is the polygon $P$ truncated at horizons more than $\delta$ away from $s$. The robot imagines that it is in $P^6$, and it explores this polygon. If it fails to find $t$, it doubles $\delta$ and tries again. All the points of $P$ within some distance of $s$ are visible from $s$; let us assume that this distance is 1. Hence, when $\delta = 1$, the robot explores $P^6$ by standing still. Otherwise, it follows a modified version of the exploration algorithm of [DKP]. It successively crosses horizons whose endpoints it encounters in clockwise order, treating horizons at a distance of more than $\delta$ as walls. (By Lemma 12, it can correctly gauge distances in $P$, and Lemma 7 implies that the distance to a
horizon can be determined, once it is completely visible, by the distance to its endpoints.) It is not
difficult to show that combining this technique with the algorithm of the preceding section gives a
2-competitive algorithm for exploring $P^6$. Hence,

**Theorem 4** If $P$ is an unknown rectilinear polygon with $m$ essential cuts, the above algorithm is
$O(m)$-competitive for the problem of searching for a point $t$ in $P$.

*Proof.** By Lemma 13, the number of essential cuts in each $P^6$ is at most $m$. One way to explore
$P^6$ would be to travel separately to each essential cut and back to $s$. Since each essential cut is
at most $\delta$ away from $s$, this exploration route has length at most $2m\delta$. The robot is following a
2-competitive algorithm for exploring $P^6$, so it travels no more than $4m\delta$.

Since $\delta$ is doubled at each iteration, an argument very similar to that in the proof of Theorem 2
shows that the robot travels no more than $O(m)$ times $d(s, t)$ to reach $t$. ■

We note that the polygons $P_m$ can be modified to provide a lower bound of $\Omega(m)$ for the problem
of finding a shortest path between $s$ and $t$ when the coordinates of $s$ and $t$ are both known but the
map of $P$ is not. Our algorithm is therefore asymptotically optimal for this problem as well.

6. Conclusion and Open Problems

We have considered a number of questions related to exploration and search in a simple polygon.
For general search problems in rectilinear polygons, the worst-case competitive ratio is bounded
in terms of the number of essential cuts in the polygon; this is realized by simple lower-bound
examples. In this sense, the definition of a “street” is worth noting, in that it gives a fairly
general class of polygons in which a constant competitive ratio for search is achievable. It would be
interesting to find other classes of polygons which can be searched efficiently.

There are a number of other open questions that could be worth investigating. First of all, there
is still a gap between the lower bound of $5/4$ and the upper bound of 2 on the best competitive
ratio achievable for exploring a rectilinear polygon deterministically. Also, any improvement on the
randomized exploration algorithm given here would beat the deterministic lower bound. Finally,
exploration and search become quite difficult when the robot is not in a simply connected region; it
is interesting to consider which restricted cases of this problem allow for algorithms with a constant
competitive ratio, independent of the number of “holes.”

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.
References


