Competitive Online Approximation of the Optimal Search Ratio[*]

Rudolf Fleischer[1][**], Tom Kamphans[2], Rolf Klein[2], Elmar Langetepe[2], and Gerhard Trippen[3][**]

[1] `fleischer@acm.org`.
[2] University of Bonn, Institute of Computer Science I, D-53117 Bonn, Germany.
`[kamphans,rolf.klein,langetep]@informatik.uni-bonn.de`.
[3] The Hong Kong University of Science and Technology, CS Dept., Hong Kong.
`trippen@cs.ust.hk`.

Abstract. How efficiently can we search an unknown environment for a goal in unknown position? How much would it help if the environment were known? We answer these questions for simple polygons and for general graphs, by providing online search strategies that are as good as the best offline search algorithms, up to a constant factor. For other settings we prove that no such online algorithms exist.

1 Introduction

One of the recurring tasks in life is to search one's environment for an object whose location is —at least temporarily—unknown. This problem comes in different variations. The searcher may have vision, or be limited to sensing by touch. The environment may be a simple polygon, e. g., an apartment, or a graph, like a street network. Finally, the environment may be known to the searcher, or be unknown.

Such search problems have attracted a lot of interest in online motion planning, see for example the survey by Berman [4]. Usually the cost of a search is measured by the length of the search path traversed; this in turn is compared against the length of the shortest path from the start position to the point where the goal is reached. If we are searching in an unknown environment, the maximum quotient, over all goal positions and all environments, is the *competitive ratio* of the search algorithm.

Most prominent is the problem of searching two half-lines emanating from a common start point. The "doubling" strategy visits the half-lines alternatingly,

[*] The work described in this paper was partially supported by a grant from the Germany/Hong Kong Joint Research Scheme sponsored by the Research Grants Council of Hong Kong and the German Academic Exchange Service (Project No. G-HK024/02).

[**] The authors were partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. HKUST6010/01E) and by the RGC Direct Allocation Grant DAG03/04.EG05.

each time doubling the depth of exploration. This way, the goal point is reached after traversing a path at most 9 times as long as its distance from the start, and the competitive ratio of 9 is optimal for this problem; see Baeza-Yates et al. [3] and Alpern and Gal [2]. This doubling approach frequently appears as a subroutine in more complex navigation strategies.

In searching $m > 2$ half-lines, a constant ratio with respect to the distance from the start can no longer be achieved. Indeed: Even if the half lines were replaced by segments of the same finite length, the goal could be placed at the end of the segment visited last, causing the ratio to be at least $2m - 1$. Exponentially increasing the exploration depth by $m/m - 1$ is known to lead to an optimal competitive ratio of $C(m) = 1 + 2m\left(\frac{m}{m-1}\right)^{m-1} \leq 1 + 2me$.

Much less is known about more realistic settings. Suppose a searcher with vision wants to search an unknown simple polygon for a goal in unknown position. He could employ the $m-$way technique from above: By exploring the shortest paths from the start to the m reflex vertices of the polygon—ignoring their tree structure—a competitive ratio of $C(m)$ can easily be achieved [13]. Schuierer [15] has refined this method and obtained a ratio of $C(2k)$, where k denotes the smallest number of convex and concave chains into which the polygon's boundary can be decomposed.

But these results do not completely settle the problem. For one, it is not clear why the numbers m or k should measure the difficulty of searching a polygon. Also, human searchers easily outperform $m-$way search, because they make educated guesses about the shape of those parts of the polygon not yet visited.

In this paper we take the following approach: Let π be a *search path* for the fixed polygon P, i. e., a path from the start point, s, through P from which each point p inside P will eventually be visible. Let $\text{rise}_\pi(p)$ be the first point on π where this happens. The cost of getting to p via π equals the length of π from s to $\text{rise}_\pi(p)$, plus the Euclidean distance from $\text{rise}_\pi(p)$ to p. We divide this value by the length of the shortest $s-$to$-p$ path in P. The maximum of these ratios, over all $p \in P$, is the *search ratio* of π. The lowest search ratio possible, over all search paths, is the *optimum search ratio* of P; it measures the "searchability" of P.

Koutsoupias et al. [14] studied graphs with unit length edges where the goal can only be located at vertices, and they only studied the offline case, i. e., with full a priori knowledge of the graph. They showed that computing the optimal search ratio offline is an NP-complete problem, and gave a polynomial time 8-approximation algorithm based on the doubling heuristic.

The crucial question we are considering in this paper is the following: Is it possible to design an *online* search strategy whose search ratio stays within a constant factor of the optimum search ratio, for arbitrary instances of the environment? Surprisingly, the answer is positive for simple polygons as well as for general graphs. (However, for polygons with holes, and for graphs with unit edge length, where the goal positions are restricted to the vertices, no such online strategy exists.)

Note that *search ratio* and *competitive ratio* have very similar definitions, but they are actually rather different concepts. For the competitive ratio, an online search algorithm has no idea how the environment looks like and has to learn it while searching for the goal. In contrast, the search ratio is defined for a given fixed environment. Since the optimal search ratio path minimizes the quotient of search distance over shortest distance to the goal, the optimal search ratio is actually a lower bound for the competitive search ratio of any online search algorithm. Computing online a c-approximation of the optimal search ratio path means we compute online a search path whose competitive ratio is at most a factor of c worse than the optimal competitive ratio of any online search algorithm, but that does not tell us anything about the competitive ratio itself which could be arbitrarily bad. In some special cases, we can search with a constant competitive ratio, for example on the line optimally 9-competitive (the Lost-Cow problem [3]) and in street polygons optimally $\sqrt{2}$-competitive [12,16].

The *search* strategies we will present use, as building blocks, modified versions of constant-competitive strategies for online *exploration*, namely the exploration strategy by Hoffmann et al. [11] for simple polygons, and the tethered graph exploration strategy by Duncan et al. [9].

At first glance it seems quite natural to employ an exploration strategy in searching—after all, either task involves looking at each point of the environment. But there is a serious difference in performance evaluation! In searching an environment, we compete against shortest start-to-goal paths, so we have to proceed in a BFS manner. In exploration, we are up against the shortest round trip from which each point is visible; this means, once we have entered some remote part of the environment we should finish it, in a DFS manner, before moving on. However, we can fit these exploration strategies to our search problem by restricting them to a bounded part of the environment. This will be shown in Section 3 where we present our general framework, which turns out to be quite elegant despite the complex definitions. The framework can be applied to online and offline search ratio approximations. In Section 2 we review basic definitions and notations. Our framework will then be applied to searching in various environments like trees, (planar) graphs, and (rectilinear) polygonal environments with and without holes in Sections 4 and 5. Finally, in Section 6, we conclude with a summary of our results (see also Table 1).

2 Definitions

We want to find a good search path in some given *environment* $\mathcal{E}$. This might be a tree, a (planar) graph, or a (rectangular) polygon with or without holes. In a graph environment, the edges may either have unit length or arbitrary length. Edge lengths do not necessarily represent Euclidean distances, not even in embedded planar graphs. In particular, we do not assume that the triangle inequality holds.

The *goal set*, $\mathcal{G}$, is the set of locations in the environment where the (stationary!) goal might be hidden. If $\mathcal{E}$ is a graph $G = (V, E)$, then the goal may be located on some edge (*geometric search*), i. e., $\mathcal{G} = V \cup E$, or its position may be restricted to the vertices (*vertex search*), i. e., $\mathcal{G} = V$. To *explore* $\mathcal{E}$ means to

Table 1. Summary of our approximation results, where $\alpha > 0$ is an arbitrary constant. The entry marked with * had earlier been proven by Koutsoupias et al. [14]. They had also shown that computing the optimal search path is NP-complete for (planar) graphs. It is also NP-complete for polygons, whereas it is not known to be NP-complete for trees.

Environment	Edge length	Goal	Polytime approximation ratio	
			Online	Offline
Tree	unit, arbitrary	vertex, geometric	4	4
Planar graph	arbitrary	vertex	no search-competitive alg.	8
Planar graph	unit	vertex	$104 + 40\alpha + \frac{64}{\alpha}$	4
General graph	unit	vertex	no search-competitive alg.	8^*
General graph	arbitrary	geometric	$48 + 16\alpha + \frac{32}{\alpha}$	4
Simple polygon			212	8
Rect. simple polygon			$8\sqrt{2}$	8
Polygon with rect. holes			no search-competitive alg.	?

move around in $\mathcal{E}$ until all potential goal positions $\mathcal{G}$ have been seen. To *search* $\mathcal{E}$ means to follow some exploration path in $\mathcal{E}$, the *search path*, until the goal has been seen. We assume that all search and exploration paths return to the start point, s, and we make the usual assumption that goals must be at least a distance 1 away from the start point.[1]

For $d \geq 1$, let $\mathcal{E}(d)$ denote the part of $\mathcal{E}$ in distance at most d from s. A *depth-d restricted exploration* explores all potential goal positions in $\mathcal{G}(d)$. The exploration path may move outside $\mathcal{E}(d)$ as long as it stays within $\mathcal{E}$. Depth-d restricted search is defined accordingly.

It remains to define what it means that the searcher "sees" the goal. In graph searching, agents are usually assumed to be *blind*, i. e., standing at a vertex of a directed graph the agent sees the set of outgoing edges, but neither their lengths nor the position of the other vertices are known. Incoming edges cannot be sensed; see [7]. Blind agents must eventually visit all points in the goal set.

Polygon searchers are usually assumed to have *vision*, that is, they always know the current visibility polygon. Such agents need not visit a goal position if they can see it from somewhere else. Searching a polygon means to visit all visibility cuts, i. e., all rays extending the edges of the reflex vertices. Actually, there is always a subset of the cuts, the *essential cuts*, whose visit guarantees that all other cuts are also visited on the way.

In case of online algorithms, we assume that agents have perfect memory. They always know a map of the part of $\mathcal{E}$ already explored, and they can always recognize when they visit some point for the second time, i. e., they have perfect localization (the robot localization problem is actually a difficult problem by itself, see for example [10]).

[1] If goals could be arbitrarily close to s, no algorithm could be competitive.

We now introduce a few notations. Let π be a path in the environment $\mathcal{E}$. For a point $p \in \pi$ let $\pi(p)$ denote the part of π between s and p, and $\mathrm{sp}(p)$ the shortest path from s to p in $\mathcal{E}$. We denote the length of a path segment $\pi(p)$ by $|\pi(p)|$. Paths computed by some algorithm $\mathcal{A}$ will be named $\mathcal{A}$, too. For a point $p \in \mathcal{E}$ let $\mathrm{rise}_\pi(p)$ denote the point $q \in \pi$ from which p is seen for the first time when moving along π starting at s, see Fig. 1.

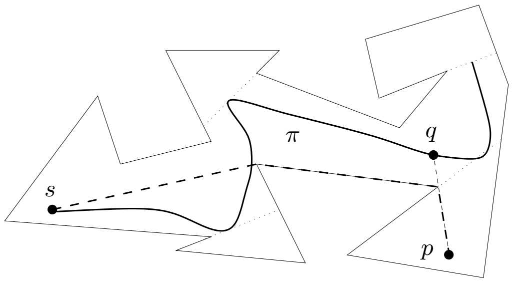

Fig. 1. A search path π in a polygon, visiting all essential cuts (the dotted lines). The dashed path is the shortest path $sp(p)$ from s to the goal p. Moving along π, p can first be seen from $q = \mathrm{rise}_\pi(p)$.

The quality of a search path is measured by its *search ratio* $\mathrm{sr}(\pi)$, defined as $\mathrm{sr}(\pi) := \max_{p \in \mathcal{G}} \frac{|\pi(q)|+|qp|}{|\,\mathrm{sp}(p)|}$, where $q = \mathrm{rise}_\pi(p)$. Note that $q = p$ for blind agents. An *optimal search path*, π_{opt}, is a search path with a minimum search ratio $\mathrm{sr}_{\mathrm{opt}} = \mathrm{sr}(\pi_{\mathrm{opt}})$.

Since the optimal search path seems hard to compute [14], we are interested in finding good approximations of the optimal search path, in offline and online scenarios. We say a search path π is *C-search-competitive* (with respect to the optimal search path π_{opt}) if $\mathrm{sr}(\pi) \le C \cdot \mathrm{sr}(\pi_{\mathrm{opt}})$. Note that π is then a $C \cdot \mathrm{sr}(\pi_{\mathrm{opt}})$-competitive search algorithm (in the usual competitive sense).

3 A General Approximation Framework

In this section we will show how to transform an exploration algorithm, offline or online, into a search algorithm, without losing too much on the approximation factor.

Let $\mathcal{E}$ be the given environment and π_{opt} an optimal search path. We assume that, for any point p, we can reach s from p on a path of length at most $\mathrm{sp}(p)$.[2]

For $d \ge 1$, let *Expl*(d) be a family of depth-d restricted exploration algorithms for $\mathcal{E}$, either online or offline. Let OPT and OPT(d) denote the corresponding optimal offline depth-d restricted exploration algorithms.

[2] Note that this is not the case for directed graphs, but it holds for undirected graphs and polygonal environments. We will see later that there is no constant-competitive online search algorithm for directed graphs, anyway.

Definition 1. The family *Expl*(d) is *DREP (depth restricted exploration property)* if there are constants $\beta > 0$ and $C_\beta \geq 1$ such that, for any $d \geq 1$, *Expl*(d) is C_β-competitive against the optimal algorithm $\mathrm{OPT}(\beta d)$, i. e., $|\mathit{Expl}(d)| \leq C_\beta \cdot |\mathrm{OPT}(\beta d)|$. □

In the normal competitive framework we would compare *Expl*(d) to the optimal algorithm $\mathrm{OPT}(d)$, i. e., $\beta = 1$. As we will see later, our more general definition makes it sometimes easier to find DREP exploration algorithms. Usually, we cannot just take an exploration algorithm *Expl* for $\mathcal{E}$ and restrict it to points in distance at most d from s. This way, we might miss useful shortcuts outside of $\mathcal{E}(d)$. Even worse, it may not be possible to determine in an online setting which parts of the environment belong to $\mathcal{E}(d)$, making it difficult to explore the right part of $\mathcal{E}$. In the next two sections we will derive DREP exploration algorithms for graphs and polygons by carefully adapting existing exploration algorithms for the entire environment.

To obtain a search algorithm for $\mathcal{E}$ we use the *doubling strategy.* For $i = 1, 2, 3, \ldots$, we successively run the exploration algorithm *Expl*(2^i), each time starting at s.

Theorem 1. *The doubling strategy based on a DREP exploration strategy is a $4\beta C_\beta$-search-competitive (plus an additive constant) search algorithm for blind agents, and a $8\beta C_\beta$-search-competitive (plus an additive constant) search algorithm for agents with vision.*

Proof. Consider one iteration of the doubling strategy with search radius $d \geq 1$. Let $\mathrm{last}(d)$ be the point on the optimal search path π_{opt} for $\mathcal{E}$ from which we see the last point in distance at most d from s when moving along π_{opt}. If the agent has vision, $\mathrm{last}(d)$ could lie outside of $\mathcal{E}(d)$. Note that $|\mathrm{OPT}(d)| \leq |\pi_{\mathrm{opt}}(\mathrm{last}(d))| + |\,\mathrm{sp}(\mathrm{last}(d))|$. For a blind agent we have $\mathrm{sp}(\mathrm{last}(d)) \leq d$. Thus, $\mathrm{sr}_{\mathrm{opt}} \geq \frac{\pi_{\mathrm{opt}}(\mathrm{last}(d))}{d} \geq \frac{|\mathrm{OPT}(d)|-d}{d}$, or $|\mathrm{OPT}(d)| \leq d \cdot (\mathrm{sr}_{\mathrm{opt}} + 1)$. If the goal is in distance $2^j + \epsilon$ for some small $\epsilon > 0$, then the search ratio of the doubling strategy is bounded by $\frac{\sum_{i=1}^{j+1} |\mathit{Expl}(2^i)|}{2^j} \leq C_\beta \cdot \frac{\sum_{i=1}^{j+1} |\mathrm{OPT}(\beta 2^i)|}{2^j} \leq C_\beta \cdot \frac{\sum_{i=1}^{j+1} \beta 2^i \cdot (\mathrm{sr}_{\mathrm{opt}}+1)}{2^j} \leq 4\beta C_\beta \cdot (\mathrm{sr}_{\mathrm{opt}} + 1)$.

If the agent has vision, we only know that $|\,\mathrm{sp}(\mathrm{last}(d))| \leq |\pi_{\mathrm{opt}}(\mathrm{last}(d))|$. Thus, $\mathrm{sr}_{\mathrm{opt}} \geq \frac{|\pi_{\mathrm{opt}}(\mathrm{last}(d))|}{d} \geq \frac{|\mathrm{OPT}(d)|}{2d}$, or $|\mathrm{OPT}(d)| \leq 2d \cdot \mathrm{sr}_{\mathrm{opt}}$. So in this case we can bound the search ratio by $\frac{2^j + \sum_{i=1}^{j+1} |\mathit{Expl}(2^i)|}{2^j} \leq 1 + 8\beta C_\beta \cdot \mathrm{sr}_{\mathrm{opt}}$. □

The only problem is now to find good DREP exploration algorithms for various environments.

4 Searching Graphs

We distinguish between graphs with unit length vs. arbitrary length edges, planar vs. non-planar graphs, directed vs. undirected graphs, and vertex vs. geometric search. We assume agents are blind, i. e., they can at any vertex only sense the number of outgoing edges but they cannot sense the incoming edges and the endpoints of the outgoing edges. In the vertex search problem, we assume w.l.o.g. that graphs do not have parallel edges. Otherwise, there can be no constant-search-competitive vertex search algorithm. In Fig. 2(iv), the optimal search path $s \to v \to t \to s$ has length 3, whereas any online search path can be forced to cycle often between s and v before traversing the edge $v \to t$. Note that we also could use undirected edges.

4.1 Non-competitiveness Results

We first show that for many variants there is no constant-search-competitive online search algorithm. Incidentally, there is also no constant-competitive online exploration algorithm for these graph classes. Note that non-search-competitiveness for planar graphs implies non-search-competitiveness for general graphs, non-search-competitiveness for unit length edges implies non-search-competitiveness for arbitrary length edges, and non-search-competitiveness for undirected graphs implies non-search-competitiveness for directed graphs (we could replace each undirected edge with directed edges in both directions).

Theorem 2. *For blind agents, there is no constant-search-competitive online vertex search algorithm for (i) non-planar graphs, (ii) directed planar graphs, (iii) planar graphs with arbitrary edge lengths. Further, there is no constant-search-competitive online geometric search algorithm for directed graphs with unit length edges.*

Proof. It is not difficult to verify the claims on the graphs in Fig. 2(i)-(iii). □

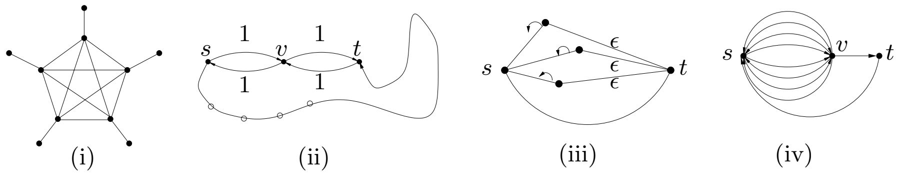

Fig. 2. Lower bound constructions of Theorem 2.

4.2 Competitive Search in Graphs

In this subsection we will present search-competitive online and offline search algorithms for the remaining graph classes. We assume in this subsection that graphs are always undirected.

Trees. On trees, DFS is a 1-competitive online exploration algorithm for vertex and geometric search that is DREP; it is still 1-competitive when restricted to search depth d, for any $d \geq 1$. Thus, the doubling strategy gives a polynomial time 4-search-competitive search algorithm for trees, online and offline. On the other hand, it is an open problem whether the computation of an optimal vertex or geometric search path in trees with unit length edges is NP-complete [14].

Competitive Graph Search Algorithms. We will now give competitive search algorithms for planar graphs with unit length edges (vertex search) and for general graphs with arbitrary length edges (geometric search). Both algorithms are based on an online algorithm for online tethered graph exploration.

In the *tethered exploration* problem the agent is fixed to the start point by a restricted length rope. An optimal solution to this problem was given by Duncan et al. [9]. Their algorithm can explore an unknown graph with unit length edges in $2|E| + (4 + \frac{16}{\alpha})|V|$ edge traversals, using a rope length of $(1 + \alpha)d$, where d is the distance of the point farthest away from the start point and $\alpha > 0$ is some parameter. As they pointed out, the algorithm can also be used for depth restricted exploration. To explore all edges in $G((1 + \alpha)d)$, for $d \geq 1$, their algorithm uses at most $2|E((1 + \alpha)d)| + (4 + \frac{16}{\alpha})|V((1 + \alpha)d)|$ edge traversals using a rope of length $(1+\alpha)d$. Let us call this algorithm *Expl*(d). The algorithm explores the graph in a mixture of bounded-depth DFS on G, DFS on some spanning tree of G, and recursive calls to explore certain large subgraphs. The bound for the number of edge traversals can intuitively be explained as follows: bounded-depth DFS visits each edge at most twice, thus the term $2|E((1+\alpha)d)|$; DFS on the spanning tree visits every node at most twice, but nodes can be in two overlapping spanning trees, thus the term $4|V((1+\alpha)d)|$; relocating between recursive calls does not happen too often (because of the size of the subgraphs), giving the term $\frac{16}{\alpha}|V((1 + \alpha)d)|$.

We note that *Expl*(d) can be modified to run on graphs with arbitrary length edges. Then we do not bound the number of edge traversals, but the total length of all traversed edges. Let *length*(E) denote the total length of all edges in E. It is possible to adapt the proofs in [9] to prove the following lemma.

Lemma 1. *In graphs with arbitrary length edges, Expl*(d) *explores all edges and vertices in* $G(d)$ *using a rope of length* $(1 + \alpha)d$ *at a cost of at most* $(4 + \frac{8}{\alpha}) \cdot \mathit{length}(E((1 + \alpha)d))$.

Proof. (Sketch) Intuitively, DFS and bounded-depth DFS traverse each edge at most twice, thus the term $4 \cdot \mathit{length}(E((1+\alpha)d))$; relocating between recursive calls does not happen too often and subgraphs do not overlap (at least not their edges), thus the term $\frac{8}{\alpha} \cdot \mathit{length}(E((1 + \alpha)d))$. □

Lemma 2. *In planar graphs with unit length edges, Expl*(d) *is a DREP online vertex exploration algorithm with* $\beta = 1 + \alpha$ *and* $C_\beta = 10 + \frac{16}{\alpha}$.

Proof. $E((1 + \alpha)d) \leq 3V((1 + \alpha)d) - 6$ by Euler's formula. Thus, the number of edge traversals is at most $(10+\frac{16}{\alpha})V((1+\alpha)d)$. On the other hand, $\mathrm{OPT}((1+\alpha)d)$ must visit each vertex in $V((1 + \alpha)d)$ at least once. □

Theorem 3. *The doubling strategy based on Expl(d) is a $(104 + 40\alpha + \frac{64}{\alpha})$-search-competitive online vertex search algorithm for blind agents in planar graphs with unit length edges. The competitive ratio is minimal for $\alpha = \sqrt{\frac{16}{10}}$.* □

Lemma 3. *In general graphs with arbitrary length edges, Expl(d) is a DREP online geometric exploration algorithm with $\beta = 1 + \alpha$ and $C_\beta = 4 + \frac{8}{\alpha}$.*

Proof. The total cost of *Expl*(d) is at most $(4+\frac{8}{\alpha}) \cdot length(E((1+\alpha)d))$ by Lemma 1. On the other hand, OPT$((1+\alpha)d)$ must traverse each edge in $E((1+\alpha)d)$ at least once. □

Theorem 4. *The doubling strategy based on Expl(d) is a $(48 + 16\alpha + \frac{32}{\alpha})$-search-competitiveonline geometric search algorithm for blind agents in general graphs with arbitrary length edges.* □

5 Searching Polygons

5.1 Simple Polygons

A simple polygon P is given by a closed non-intersecting polygonal chain. We assume that agents have vision. To apply our framework we need a DREP online exploration algorithm *Expl*$_{\text{onl}}(d)$.

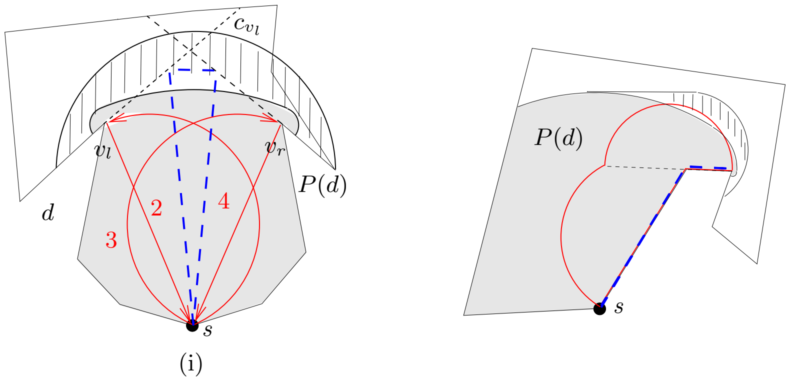

Fig. 3. (i) $PE(d)$ explores the left reflex vertex v_l along a circular arc (1), returns to the start (2) and explores the right reflex vertex v_r likewise (3)+(4). On the other hand, the shortest exploration path for $P(d)$ in P, the dashed line, leaves $P(d)$. But we can extend $P(d)$ by the circular arc. (ii) $PE(d)$ leaves $P(d)$ whereas the shortest exploration path for $P(d)$ lies inside $P(d)$.

The only known algorithm, PE, for the online exploration of a simple polygon by Hoffmann et al. [11] achieves a competitive ratio of 26.5. Now we must adapt this algorithm to depth restricted exploration. The undetected parts of P always lie behind cuts c_v emanating from reflex vertices v. These reflex vertices are called *unexplored* as long as we have not visited the corresponding cut c_v. We modify *PE* so that it explores P only up to distance d from s. The algorithm always maintains a list of unexplored reflex vertices and successively visits the corresponding cuts. While exploring a reflex vertex (along a sequence of line segments and circular arcs), more unexplored reflex vertices may be detected or unexplored reflex vertices may become explored. These vertices are inserted into or deleted from the list, respectively. In $PE(d)$, unexplored reflex vertices in a distance greater than d from the start will be ignored, i. e., although they may be detected they will not be inserted into the list. Let $\mathrm{OPT}(P, d)$ be the shortest path that sees all points in $P(d)$.

Note that $\mathrm{OPT}(P, d)$ and $PE(d)$ may leave $P(d)$, see Fig. 3. Nevertheless, it is possible to adapt the analysis of Hoffmann et al. There are actually two ways to do this. Either we enlarge $P(d)$ without creating new reflex vertices such that the larger polygon contains $\mathrm{OPT}(P, d)$ and $PE(d)$. Or we redo the analysis of Hoffmann et al. in P restricted to $\mathrm{OPT}(P, d)$ and $PE(d)$, which is possible. In the former case we can use some sort of extended boundary around the boundary of $P(d)$ such that the enlarged polygon contains $PE(d)$ and $\mathrm{OPT}(P, d)$, see the convex enlargements in Fig. 3. It may happen that these new extensions overlap for different parts of $P(d)$. However, this does not affect the analysis of Hoffmann et al.

Lemma 4. *In a simple polygon, $PE(d)$ is a DREP online exploration algorithm with $\beta = 1$ and $C_\beta = 26.5$.* □

Theorem 5. *The doubling strategy based on $PE(d)$ is a 212-search-competitive online search algorithm for an agent with vision in a simple polygon. There is also a polynomial time 8-search-competitive offline search algorithm.*

Proof. The online search-competitiveness follows from Lemma 4 and Theorem 1.

If we know the polygon, we can compute $\mathrm{OPT}(P, d)$ in polynomial time by adapting a corresponding algorithm for P. Every known polynomial time offline exploration algorithm builds a sequence of the essential cuts, see for example [5, 18,17,8]. Any of these algorithms could be used in our framework. Since an optimal algorithm has approximation factor $C = 1$, our framework yields an approximation of the optimal search ratio within a factor of 8.

If we skip a step with distance 2^i if there is no reflex vertex within a distance between 2^{i-1} and 2^i, the total running time is bounded by a polynomial in the number of the vertices of P. □

Now the question arises whether there is a polynomial time algorithm that computes the optimal search path in a simple polygon. All essential cuts need to be visited, so we can try to visit them in any possible order. However, we do not know where exactly we should visit a cut. We are not sure whether there are only a few possibilities (similar to the shortest watchman route problem), i. e., whether this subproblem is discrete. So the problem of efficiently computing an optimal search path in a polygon is still open.

For rectilinear simple polygons we can find better online algorithms based on a $\sqrt{2}$-competitive online exploration algorithm by Papadimitriou et al. [6] which can be made DREP by ignoring reflex vertices farther away than d. Again, no polynomial time algorithm for the optimal search path is known.

Theorem 6. *For an agent with vision in a simple rectilinear polygon there is a* $8\sqrt{2}$*-search-competitive online search algorithm. There is also a polynomial time* 8*-search-competitive offline search algorithm.* □

5.2 Polygons with Holes

We will show that there is no constant-search-competitive online search algorithm for polygons with rectangular holes. It was shown by Albers et al. [1] that there is no constant-competitive online exploration algorithm for polygons with rectangular holes. For $k \geq 2$, they filled a rectangle of height k and width $2k$ with $O(k^2)$ rectangular holes such that the optimal exploration tour has length $O(k)$, whereas any online exploration algorithm needs to travel a distance of $\Omega(k^2)$. The details of the construction are not important here. We just note that it has the property that any point p is at most at distance $3k$ from the start point, which is in the lower left corner of the bounding rectangle.

Theorem 7. *For an agent with vision in a polygon with rectangular holes there is no constant-search-competitive online search algorithm.*

Proof. We extend the construction of Albers et al. by making the bounding rectangle larger and placing a new hole of height k and width $2k$ just below all the holes of the previous construction. The new start point s is again in the lower left corner of the bounding rectangle. Any point that is not immediately visible from s has at least distance k from s. □

Since the offline exploration problem is NP-complete (by straightforward reduction from planar TSP) we cannot use our framework to obtain a polynomial time approximation algorithm of the optimal search path. However, there is an exponential time 8-approximation algorithm. We can list the essential cuts of $\mathrm{OPT}(P, d)$ in any order to find the best one. Application of our framework then gives an approximation factor of 8 for the optimal search ratio.

The results of Koutsoupias et al. [14] imply that the offline problem of computing an optimal search path in a known polygon with (rectangular) holes is NP-complete.

6 Conclusion and Open Problems

We have introduced a framework for computing online and offline approximations of the optimal search path graph and polygonal environments. We have obtained fairly simple proofs of the existence of approximation strategies and their approximation factors, although the factors are quite high. We have also shown that some environments do not have constant-search-competitive online search strategies. Our framework would also work for randomized algorithms, but there are not many randomized exploration algorithms around. For some of the settings it remains open whether the offline optimization problem is NP-hard.

References

1. Susanne Albers, Klaus Kursawe, and Sven Schuierer. Exploring unknown environments with obstacles. In *Proc. 10th SODA*, pp. 842–843, 1999.
2. Steve Alpern and Shmuel Gal. *The Theory of Search Games and Rendezvous.* Kluwer Academic Publications, 2002.
3. R. Baeza-Yates, J. Culberson, and G. Rawlins. Searching in the plane. *Inform. Comput.*, 106:234–252, 1993.
4. Piotr Berman. On-line searching and navigation. In A. Fiat and G. Woeginger, eds., *Competitive Analysis of Algorithms.* Springer, 1998.
5. W.-P. Chin and S. Ntafos. Shortest watchman routes in simple polygons. *Discrete Comput. Geom.*, 6(1):9–31, 1991.
6. Xiaotie Deng, Tiko Kameda, and Christos Papadimitriou. How to learn an unknown environment I: The rectilinear case. *Journal of the ACM*, 45(2):215–245, 1998.
7. Xiaotie Deng and Christos H. Papadimitriou. Exploring an unknown graph. *Journal of Graph Theory*, 32:265–297, 1999.
8. M. Dror, A. Efrat, A. Lubiw, and J. S. B. Mitchell. Touring a sequence of polygons. In *Proc. 35th STOC*, pp. 473–482, 2003.
9. Christian A. Duncan, Stephen G. Kobourov, and V. S. Anil Kumar. Optimal constrained graph exploration. In *Proc. 12th SODA*, pp. 307–314, 2001.
10. Rudolf Fleischer, Kathleen Romanik, Sven Schuierer, and Gerhard Trippen. Optimal robot localization in trees. *Information and Computation*, 171:224–247, 2001.
11. Frank Hoffmann, Christian Icking, Rolf Klein, and Klaus Kriegel. The polygon exploration problem. *SIAM Journal on Computing*, 31:577–600, 2001.
12. Christian Icking, Rolf Klein, and Elmar Langetepe. An optimal competitive strategy for walking in streets. In *Proc. 16th STACS*, LNCS 1563, pp. 110–120. Springer, 1999.
13. R. Klein. *Algorithmische Geometrie.* Addison-Wesley Longman, 1997.
14. Elias Koutsoupias, Christos H. Papadimitriou, and Mihalis Yannakakis. Searching a fixed graph. In *Proc. 23th ICALP*, LNCS 1099, pp. 280–289. Springer, 1996.
15. S. Schuierer. On-line searching in simple polygons. In H. Christensen, H. Bunke, and H. Noltemeier, eds., *Sensor Based Intelligent Robots*, LNAI 1724, pp. 220–239. Springer, 1997.
16. Sven Schuierer and Ines Semrau. An optimal strategy for searching in unknown streets. In *Proc. 16th STACS*, LNCS 1563, pp. 121–131. Springer, 1999.
17. X. Tan, T. Hirata, and Y. Inagaki. Corrigendum to "an incremental algorithm for constructing shortest watchman routes". *Internat. J. Comput. Geom. Appl.*, 9(3):319–323, 1999.
18. X. H. Tan, T. Hirata, and Y. Inagaki. An incremental algorithm for constructing shortest watchman routes. *Internat. J. Comput. Geom. Appl.*, 3(4):351–365, 1993.