Competitive Online Approximation of the Optimal Search Ratio[*]

Rudolf Fleischer[†] Tom Kamphans[‡] Rolf Klein[‡] Elmar Langetepe[‡]
Gerhard Trippen[§]

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 undirected 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.

Keywords: Online motion planning, competitive ratio, searching, exploration.

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, for example, 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 [5]. 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. The maximum quotient of these values, taken over all environments and all goal positions within an environment, 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, 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. [4] 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

[*]A preliminary version of this paper appeared at ESA 2004 [13].

[†]Fudan University, Shanghai Key Laboratory of Intelligent Information Processing, Department of Computer Science and Engineering, Shanghai, China. The work described in this paper was partially supported by a grant from the National Natural Science Fund China (grant no. 60573025).

[‡]University of Bonn, Institute of Computer Science I, D-53117 Bonn, Germany.

[§]The Hong Kong University of Science and Technology, Hong Kong, Department of Computer Science.
The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. HKUST6010/01E) and 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).

be at least $2m - 1$. Exponentially increasing the exploration depth by $m/m - 1$ is known [4, 2] 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 [20]. Schuierer [24] 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 can often 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 a 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 p_π be the first point on π where this happens for a point p. The cost of getting to p via π is equal to the length of π from s to p_π, plus the Euclidean distance from p_π to p. We divide this value by the length of the shortest $s-$to$-p$ path in P. The supremum 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.

Apparently, this definition was first given by Koutsoupias et al. [21]. They studied graphs with unit length edges where the goal can be located only at vertices, and they studied only the offline case where the graph is completely known a priori. 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 undirected 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.)

Remark that this way of measuring performance is one step beyond competitivity. Although the definitions of the search ratio and the competitive factor are quite similar, the concepts are different. In the competitive framework, we simply compare the online path from the start to the goal to the shortest s–to–t path. For an approximation of the optimal search ratio, we compare the online path to the best possible offline path, which—in turn—may already have a bad competitive ratio.

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. [19] for simple polygons, and the tethered graph exploration strategy by Duncan et al. [11].

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.[1] In exploration, we are up against the shortest

[1]But, as opposed to searching a data structure, we do not have pointers that allow us to jump to a different location for free.

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.[2] 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 both 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 the Sections 4 and 5. In Section 6 we give a construction scheme for lower bounds on the search ratio. Finally, in Section 7, we conclude with a summary of our results.

2 Definitions

We want to find a good search path in some given environment $\mathcal{E}$. This may be a tree, a (planar) graph, or a (rectangular) polygon with or without holes. In a graph environment, the edges may have either 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 only restriction to the type of environments required by our framework in Section 3 is that there is a shortest path from every point, p, in $\mathcal{E}$ back to the start point, s, that is of the same length as a shortest path from s to p;[3] that is, for all p in $\mathcal{E}$ holds $|\operatorname{sp}(s,p)| = |\operatorname{sp}(p,s)|$.

In most cases, we want to search the whole environment, but there are special kinds of search problems where we know that the goal may be hidden only in some parts of the scene. Let the *goal set* $\mathcal{G} \subseteq \mathcal{E}$ denote the part of the environment where the stationary goal may be located. For example, if we search in a graph, $G = (V, E)$, the goal may be located anywhere along the edges of the graph; we call this setting *geometric search* and set $\mathcal{G} := V \cup E$. On the other hand, we may consider a *vertex search*, where the goal is restricted to be hidden in a vertex; in this case, we set $\mathcal{G} := V$. Now, *exploring* $\mathcal{E}$ means to move around in $\mathcal{E}$ until all potential goal positions $\mathcal{G}$ have been seen, whereas *searching* in $\mathcal{E}$ means to follow some exploration path in $\mathcal{E}$, until the goal has been seen. We require—as usual—that the distance to the goal is at least 1; otherwise, no search strategy is able to achieve a bounded competitive factor. Further, we assume that agents have perfect localization abilities; that is, they always know a map of the already explored part of $\mathcal{E}$, and they can always recognize when they visit some point for the second time (the robot localization problem is actually a difficult problem by itself, see for example [14]).

The searcher can either be *blind* (i.e., it can sense only its very close neighborhood) or it can have *vision*; that is, it can see objects far away from its current position if the line of sight is not blocked by an obstacle. The model of visibility depends on the type of environment; see Sections 4 and 5 for more details.

We now introduce a few notations: Given a start point, $s \in \mathcal{E}$, let π be a path in the environment $\mathcal{E}$ starting in s. For a given point $q \in \pi$ let $\pi(q)$ denote the part of π from s to q. For an arbitrary point $p \in \mathcal{E}$ let $\operatorname{sp}(p)$ denote a shortest path from s to p in the given environment, and let $p_\pi \in \pi$ denote the point from which a searcher following π sees p for the first time, see Figure 1. We denote the length of a path segment $\pi(p)$ by $|\pi(p)|$. Paths computed

[2]Observe, however, that neither plain BFS nor DFS would work! BFS is lacking the doubling element, and DFS, in a simple polygon, would tend to follow a long convex chain even though a small step to the side could be sufficient to see its endpoint.

[3]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.

by some algorithm $\mathcal{A}$ will be named $\mathcal{A}$, too. In particular, we write $|\mathcal{A}|$ for the length of the tour computed by $\mathcal{A}$. The main concept in our paper is the following:

Definition 1 Let $\mathcal{E}$ be an environment, $\mathcal{G} \subseteq \mathcal{E}$ a goal set and $s \in \mathcal{E}$ a point in the environment. A *search path*, π, with start point s is a path which starts in s and allows a searcher following π to see every goal position in $\mathcal{G}$ from at least one point on π. The *search ratio*, $\mathrm{sr}(\pi)$, is defined as

$$\mathrm{sr}(\pi) := \sup_{p \in \mathcal{G}} \frac{|\pi(p_\pi)| + |p_\pi p|}{|\,\mathrm{sp}(p)|}\,.$$

In other words, we compare the path walked by a searcher to the shortest path, and take the worst ratio among all possible targets as the search ratio of our search path.

An *optimal search path*, π_{opt}, is a search path with a minimum search ratio among all possible paths in the environment. We denote the *optimal search ratio* by $\mathrm{sr}_{\mathrm{opt}}$; that is, $\mathrm{sr}_{\mathrm{opt}} := \mathrm{sr}(\pi_{\mathrm{opt}})$. For blind agents, $p = p_\pi$ holds for every $p \in \mathcal{E}$; therefore, the search ratio can be computed as $\mathrm{sr}(\pi) := \sup_{p \in \mathcal{G}} \frac{|\pi(p)|}{|\,\mathrm{sp}(p)|}$.

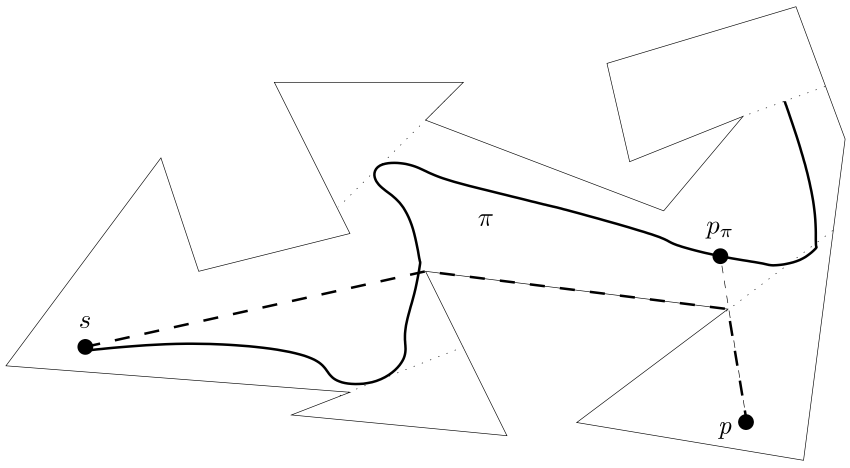


Figure 1: A search path π in a polygon, visiting all essential cuts (the dotted lines). The dashed path is the shortest path $sp(p)$ from s to the goal p. Moving along π, p can first be seen from p_π.

As the optimal search path seems hard to compute [21], we are interested in finding good approximations of the optimal search path, in offline and online scenarios. We say a search algorithm $\mathcal{A}$ is *search-competitive* with factor C—or C–search-competitive for short—if there are constants $C \geq 1$ and $B \geq 0$, so that $\mathrm{sr}(\pi_{\mathcal{A}}) \leq C \cdot \mathrm{sr}_{\mathrm{opt}} + B$ holds for every path $\pi_{\mathcal{A}}$ computed by $\mathcal{A}$. Note that $\pi_{\mathcal{A}}$ is then a $C \cdot \mathrm{sr}(\pi_{\mathrm{opt}})$-competitive search path in the usual competitive sense. We use the term C–search-competitive also for the approximation factor of offline approximation algorithms. If there is no C–search-competitive algorithm for any constant C, we call this type of environments *hard searchable.*

In the following, we want to use existing exploration algorithms to approximate the optimal search path. We assume that every exploration algorithm returns to the start point when the whole environment is explored.

Definition 2 For a given environment $\mathcal{E}$ and $d \geq 1$, let $\mathcal{E}(d)$ denote the part of $\mathcal{E}$ in distance at most d from s, and $\mathrm{OPT}(d)$ the optimal exploration for $\mathcal{E}(d)$. Further, let *Expl* be an—online or offline—algorithm for the exploration of environments of the given type. *Expl* is called *depth-restrictable* if for every $d \geq 1$ it is possible to modify the algorithm *Expl* to an algorithm *Expl*(d) that explores $\mathcal{E}(d)$ (i.e., the algorithm sees at least all potential goal positions of distance at most d—and maybe some more—before it returns to the start point), and there are constants $\beta > 0$ and $C_\beta \geq 1$, so that

$$|\mathit{Expl}(d)| \leq C_\beta \cdot |\mathrm{OPT}(\beta \cdot d)| \tag{1}$$

holds for every environment of the given type (i.e., *Expl*(d) is C_β-competitive with respect to $\mathrm{OPT}(\beta \cdot d)$).

For example, the DFS traversal for trees is depth restrictable with $\beta = 1$ and $C_\beta = 1$: In every step we know exactly the distance to the tree's root. Thus, we can decide whether we can explore the children of the current node, or cannot explore the tree more deeply, because we have reached depth d. Obviously, DFS is optimal also for depth-restricted exploration.

In the usual 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 depth-restrictable 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 Sections 4 and 5 we will derive depth-restricted exploration algorithms for graphs and polygons by carefully adapting existing exploration algorithms for the entire environment.

3 A General Approximation Framework

In this section, we show how to transform a depth-restrictable 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. Remember that we assume that, for any point p, we can reach s from p on a path of length at most $\mathrm{sp}(p)$.

Let *Expl* be an exploration algorithm for $\mathcal{E}$, and for $d \geq 1$, let *Expl*(d) be the corresponding depth-restricted exploration algorithm for $\mathcal{E}(d)$. Let OPT and $\mathrm{OPT}(d)$ denote the corresponding optimal offline depth-restricted exploration algorithms. We assume that the exploration strategies will always return to the start.

To obtain a search algorithm for $\mathcal{E}$, we use the well-known *doubling paradigm*, and successively apply our given exploration strategy with increasing exploration depth; that is, we successively run *Expl*(2^i), each iteration starting and ending in the start point, s.

Theorem 3 *The doubling strategy based on a depth-restrictable exploration algorithm with factors C_β and β is a $4\beta C_\beta$–search-competitive search algorithm for blind agents, and a $8\beta C_\beta$–search-competitive search algorithm for agents with vision.*

Proof. Consider one iteration of the doubling strategy with search radius $d \geq 1$. The optimal search path π_{opt} for $\mathcal{E}$ must in particular explore all possible goal positions in distance at most d from s. Let last_d be the point on π_{opt} from which we see the last point in distance at most d from s when moving along π_{opt}. Returning from last_d to s closes an exploration tour of $\mathcal{E}(d)$; therefore,

$$|\mathrm{OPT}(d)| \leq |\pi_{\mathrm{opt}}(\mathit{last}_d)| + |\,\mathrm{sp}(\mathit{last}_d)|\,.$$

In contrast to blind searchers, last_d may be located outside $\mathcal{E}(d)$ for agents with vision; thus, we distinguish between blind agents and agents with vision to bound $|\,\mathrm{sp}(\mathit{last}_d)|$. A blind agent can detect last_d only by visiting it, so we have $\mathrm{sp}(\mathit{last}_d) \leq d$. Thus,

$$\mathrm{sr}_{\mathrm{opt}} \geq \frac{|\pi_{\mathrm{opt}}(last_d)|}{d} \geq \frac{|\mathrm{OPT}(d)| - d}{d} \iff |\mathrm{OPT}(d)| \leq d \cdot (\mathrm{sr}_{\mathrm{opt}} + 1)\,. \tag{2}$$

The worst case for the search ratio of the doubling strategy occurs when we explore the environment up to some distance 2^{j+1} while the goal is in distance $2^j + \epsilon$ for some small $\epsilon > 0$. Thus, the search ratio of the doubling strategy is bounded by

$$\mathrm{sr}(\pi) \leq \frac{\sum_{i=1}^{j+1} |\mathit{Expl}(2^i)|}{2^j + \epsilon}\,.$$

Expl is depth restrictable with factor C_β, so we can apply Equation 1

$$\mathrm{sr}(\pi) \leq \frac{C_\beta}{2^j} \cdot \sum_{i=1}^{j+1} |\mathrm{OPT}(\beta \cdot 2^i)|\,.$$

Finally, with Equation 2 we get

$$\begin{aligned}
\mathrm{sr}(\pi) &\leq \frac{C_\beta}{2^j} \cdot \sum_{i=1}^{j+1} \beta \cdot 2^i \cdot (\mathrm{sr}_{\mathrm{opt}} + 1) \\
&\leq \beta C_\beta \cdot \left(\frac{2^{j+2} - 2}{2^j}\right) \cdot (\mathrm{sr}_{\mathrm{opt}} + 1) \\
&\leq 4\beta\, C_\beta \cdot (\mathrm{sr}_{\mathrm{opt}} + 1)\,.
\end{aligned}$$

If the agent has vision, it may see the last point in distance at most d from somewhere else, so we cannot guarantee $\mathrm{sp}(last_d) \leq d$. We know only $|\,\mathrm{sp}(last_d)| \leq |\pi_{\mathrm{opt}}(last_d)|$ (i.e., the return path to s cannot be longer than the path the agent traveled along π_{opt}). Thus,

$$\mathrm{sr}_{\mathrm{opt}} \geq \frac{|\pi_{\mathrm{opt}}(last_d)|}{d} \geq \frac{|\mathrm{OPT}(d)|}{2d} \iff |\mathrm{OPT}(d)| \leq 2d \cdot \mathrm{sr}_{\mathrm{opt}}\,.$$

Further, we detect the goal by applying the exploration strategy with depth 2^{j+1} and return to the start, then we still have to move to the goal. Similar to the case of blind agents, we obtain for the search ratio an upper bound of

$$\begin{aligned}
\frac{2^j + \sum_{i=1}^{j+1} |\mathit{Expl}(2^i)|}{2^j} &\leq 1 + C_\beta \cdot \frac{\sum_{i=1}^{j+1} |\mathrm{OPT}(\beta 2^i)|}{2^j} \leq 1 + 2C_\beta \cdot \frac{\sum_{i=1}^{j+1} \beta 2^i\, \mathrm{sr}_{\mathrm{opt}}}{2^j} \\
&\leq 1 + 8\beta C_\beta \cdot \mathrm{sr}_{\mathrm{opt}}\,.
\end{aligned}$$

□

In the next two sections we will apply our framework to various types of environments and agents. The difficult part is always to find good depth-restrictable exploration algorithms.

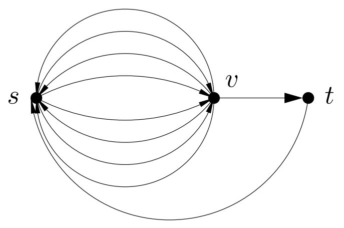
4 Searching Graphs

We distinguish between graphs with unit length and arbitrary length edges, planar and nonplanar graphs, directed and undirected graphs, as well as vertex and geometric search. We consider only blind agents: Located at a vertex of a (directed) graph the agent senses only the number of outgoing edges, but neither their lengths nor the positions of the other vertices are known. Incoming edges cannot be sensed; see Deng and Papadimitriou [9]. Blind agents must eventually visit all points in the goal set. 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 Figure 2(i), 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 can also use undirected edges.

4.1 Hard-searchable Graphs

First, we show that for many graph classes 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 we have the following implications for hard-searchable graphs:

- If planar graphs are hard searchable, then so are nonplanar graphs.
- If graphs with unit length edges are hard searchable, then so are graphs with arbitrary length edges.
- If undirected graphs are hard searchable, then so are directed graphs (we can replace each undirected edge with directed edges in both directions).



(i)

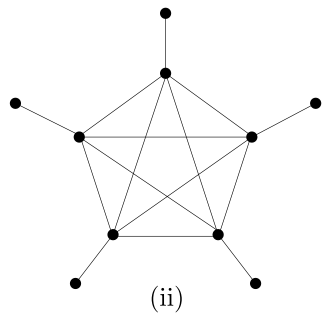

(ii)

Figure 2: There is no constant–search-competitive vertex search algorithm for (i) graphs with parallel edges, (ii) vertex search in nonplanar graphs.

Theorem 4 *For blind agents, there is no constant–search-competitive online algorithm in the following settings:*

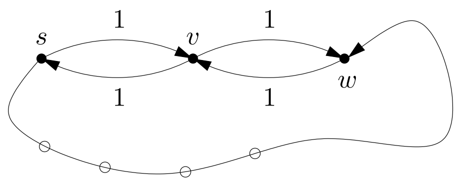
1. *Vertex search in nonplanar graphs*
2. *Vertex search and geometric search in directed planar graphs*
3. *Vertex search in planar graphs with arbitrary edge lengths*

Proof.

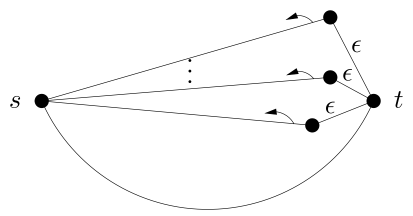
1. Consider the graph in Figure 2(ii) with unit length edges. It is a k-clique, where each vertex has a sibling that has only one connection to the clique.

   The optimal search ratio is $\Theta(k)$: We successively visit every clique vertex and explore its sibling before proceeding to the next clique vertex. This yields a path of length $3k$. On the other hand, any online search algorithm can be forced to travel $\Omega(k^2)$ before reaching the last vertex, so it has search ratio $\Omega(k^2)$. Thus, it is not better than k-competitive.

   Note that in a k-clique without siblings a BFS traversal achieves the optimal search ratio.



(i)



(ii)

Figure 3: There is no constant–search-competitive vertex search algorithm for (i) vertex search and geometric search in directed graphs (the nodes ∘ occur only in the case of unit-length edges), (ii) vertex search in planar graphs with arbitrary edge length (the arrows denote the points on the edges where the searcher decides to return to s).

2. For vertex search, consider the planar graph in Figure 3(i) with a very long edge from s to w. The optimal search path, $s \to v \to w$, has search ratio 1. However, any online algorithm can be forced to first explore the long edge from s to w, resulting in a very high search ratio.

   If we are restricted to unit length edges, we can add more vertices along the long edge (marked with ∘ in Figure 3(i)). Then the optimal search path explores this long path after exploring the short cycle $s \to w \to s$. Because in the worst case the goal is hidden on the first vertex of the long path, this path achieves a search ratio of 5.

   For a geometric search—with unit length edges or arbitrary edges—, we can use the same planar graph. Again, we force the online algorithm to explore the long edge at first. In contrast, the optimal strategy visits the cycle $s \to w \to s$ before exploring the long edge. The optimal strategy achieves its worst case if the goal is hidden in distance 1 on the long edge.

3. For any given online search algorithm, we construct a planar graph as in Figure 3(ii) with k outgoing edges at the start vertex, s. An online algorithm visits one of the k edges at first. If the algorithm does not return to s while exploring this edge, the currently visited edge is the only long edge in the graph and all other edges are very short. The optimal strategy visits only short edges; thus, this algorithm can be arbitrarily bad.

   Hence, to achieve a good approximation factor, an online algorithm must at some point decide to stop exploring the first edge and to return to s (the small arrows in the figure indicate this point). We place the endpoint of the currently visited edge immediately behind the last visited point, and continue similarly with the next $k-2$ edges. The last edge ends in vertex t and its length is the minimum length among the first $k-1$ edges. Then we connect the endpoints of the first $k-1$ edges to t by an edge of length ϵ, where $\epsilon > 0$ is some very small number. The optimal search path in this graph first travels the last edge and then visits every other vertex quickly from t. Thus, its search ratio is close to 1. On the other hand, the online algorithm has search ratio at least k. Note that we introduced all the edge endpoints (instead of having the edges ending in t) because we assumed that there are no parallel edges.

□

Note that there is a $O(D^8)$-competitive exploration for directed graphs by Fleischer and Trippen [15]. D denotes the *deficiency* of the given graph, that is the minimum number of edges that must be added to get an Euler graph. This example shows that there are settings where there is no constant–search-competitive online algorithm, although there is $O(D^8)$–competitive exploration algorithm. The problem is that this algorithm is not depth restrictable. Besides, directed graphs do not fulfill $|\operatorname{sp}(s,p)| = |\operatorname{sp}(p,s)|$ for all p in $\mathcal{E}$, so we cannot apply our framework to directed graphs, anyway.

4.2 Competitive Search in Graphs

In this subsection, we present search-competitive online and offline search algorithms for the remaining graph classes. Remark that we consider only undirected graphs.

4.2.1 Trees

On trees, DFS is a 1-competitive online exploration algorithm for vertex and geometric search that is depth restrictable; 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—offline as well as online. 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 [21].

4.2.2 Vertex Search in Graphs with Unit Length Edges

Now, we give competitive search algorithms for vertex search in planar graphs with unit length edges and—in the next section—for geometric search in undirected graphs with arbitrary length edges. Both algorithms are based on an online algorithm for tethered graph exploration.

In the *tethered exploration* problem the agent is fixed to the start point by a rope of restricted length. An optimal solution to this problem was given by Duncan et al. [11]. Their algorithm, CFS, explores 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. Note that CFS explores the whole graph in spite of the *tethered* restriction.

CFS explores the graph in a mixture of depth-bounded DFS on G, DFS on spanning trees of parts of G, and recursive calls to explore certain large subgraphs. The basic idea of CFS is to maintain a set, $\mathcal{T}$, of edge-disjoint, incompletely[4] explored subtrees—more precisely, spanning trees of uncompleted graph parts—that fulfill certain conditions on minimum size and maximal depth. Initially, $\mathcal{T}$ consists of one tree containing only the start node, s. The strategy successively selects the subtree nearest to s and walks to its root. Then, CFS traverses the subtree using DFS. For each encountered, incompletely explored vertex, CFS starts a depth-bounded DFS. In this process, new vertices are discovered and new subtrees are added to $\mathcal{T}$, possibly split into several smaller subtrees if they do not fulfill the conditions.

The costs for applying CFS sum up from relocation from s to the roots of the subtrees and back to s, the DFS traversals, and the costs for the depth-bounded DFS starting in unexplored vertices. The latter traverses only unexplored edges; thus, we have costs $2|E|$ for this part. For a subtree, T, with $|T|$ vertices, we have costs $2|T|$ for the DFS traversal. The size restrictions for the subtrees ensure that we can bound the relocation costs by $\frac{8}{\alpha}|T|$. As the subtrees may overlap, we can bounded the sum of all vertices in the subtrees by $2|V|$. Altogether, we get

$$2|E| + \left(2 + \frac{8}{\alpha}\right)\sum_{T}|T| \leq 2|E| + \left(2 + \frac{8}{\alpha}\right)\cdot 2|V| = 2|E| + \left(4 + \frac{16}{\alpha}\right)|V| \,.$$

[4]That is, there are vertices that are already discovered, but still have unvisited incident edges.

As Duncan et al. pointed out, the algorithm can be used even if the necessary rope length, d, is not known: They explore the whole graph by successively applying CFS and doubling d in every step. The important part is that the analysis still holds in this case. Particularly, we can apply CFS for a depth-restricted exploration (i.e., explore only a subgraph of G). For $d \geq 1$, let $G(d)$ denote the subgraph[5] of $G = (V, E)$ where all points $p \in V \cup E$ have distance at most $(1+\alpha)d$ from s. For convenience, let $G^* = (V^*, E^*) := G((1+\alpha)d)$. To explore all vertices in $G(d)$—and maybe some additional vertices from $G((1+\alpha)d)$—using a rope of length $(1+\alpha)d$, the number of edge traversals is bound by

$$2|E^*| + \left(4 + \frac{16}{\alpha}\right)|V^*|\,.$$

Let us call this algorithm CFS(d). We have

Lemma 5 *In planar graphs with unit length edges, CFS is a depth-restrictable algorithm for online vertex exploration with $\beta = 1+\alpha$ and $C_\beta = 10 + \frac{16}{\alpha}$.*

Proof. As G^* is planar, we have $|E^*| \leq 3|V^*| - 6$ by Euler's formula. Thus, the number of edge traversals of CFS(d) is most

$$2\,|E^*| + \left(4 + \frac{16}{\alpha}\right)|V^*| \leq 6\,|V^*| + \left(4 + \frac{16}{\alpha}\right)|V^*| = \left(10 + \frac{16}{\alpha}\right)|V^*|\,.$$

On the other hand, we have $\mathrm{OPT}((1+\alpha)d) \leq |V^*|$, because the optimal algorithm must visit each vertex in V^* at least once. $\square$

Now, we can apply our framework with CFS:

Theorem 6 *The doubling strategy based on CFS(d) is a better than 206–search-competitive online vertex search algorithm for blind agents in planar graphs with unit length edges.*

Proof. By Lemma 5, CFS(d) is depth restrictable with $\beta = 1+\alpha$ and $C_\beta = 10 + \frac{16}{\alpha}$. By Theorem 3, the doubling strategy based on CFS is $4\beta C_\beta$-competitive. Altogether, we have:

$$4 \cdot \beta \cdot C_\beta = 4 \cdot (1+\alpha) \cdot \left(10 + \frac{16}{\alpha}\right) = 104 + 40\alpha + \frac{64}{\alpha}\,.$$

By simple analysis, we get the minimal competitive ratio of 205.192... for $\alpha = \sqrt{\frac{8}{5}}$. $\square$

4.2.3 Geometric Search in Graphs with Arbitrary Length Edges

We note that CFS(d) can be modified to work on graphs with arbitrary length edges: Instead of counting the number of edge traversals, the algorithm has to track the length of the traversed edges. Now, it may happen that the maximal rope length is reached on an edge somewhere between two vertices. In this case, we interrupt the edge traversal, add an auxiliary vertex and split the edge in two parts. Note that the added vertex is incompletely explored, so CFS will return to this vertex in a successive stage. Let $\ell(E)$ denote the total length of all edges in E. It is possible to adapt the proofs by Duncan et al. [11] to prove the following lemma.

Lemma 7 *In graphs with arbitrary length edges, CFS(d) explores all edges and vertices in $G(d)$ using a rope of length $(1+\alpha)d$ at a cost of at most $(4 + \frac{8}{\alpha}) \cdot \ell(E^*)$.*

[5]In the case of unit-length edges, we omit all edges with length < 1 in the subgraph $G(d)$. Such edges occur if d is not an integer value.

Proof. *(Sketch)* The proof is similar to the unit-length case, but we can no longer use the number of visited vertices to bound the number of traversed edges. Instead, we bound the costs for the depth-bounded DFS by $2\,\ell(E^*)$. As the subtrees are edge disjoint, we can bound the costs for the DFS traversals by $2\,\ell(E^*)$, too. The size restriction on the subtrees still ensures that the relocation costs are bound by $\frac{8}{\alpha}\,\ell(E^*)$. Altogether, we get $(4+\frac{8}{\alpha})\cdot\ell(E^*)$. Note that we have no additional costs for the auxiliary vertices, because we count only edge lengths and by inserting an auxiliary vertices we split one edge into two smaller edges whose total length is the same as the original edge. $\square$

Lemma 8 *In undirected graphs with arbitrary length edges, CFS is a depth-restrictable online geometric exploration algorithm with $\beta = 1+\alpha$ and $C_\beta = 4+\frac{8}{\alpha}$.*

Proof. The total cost of CFS(d) is at most $(4+\frac{8}{\alpha})\cdot\ell(E^*)$ by Lemma 7. On the other hand, OPT($(1+\alpha)d$) must traverse each edge in E^* at least once. $\square$

Theorem 9 *The doubling strategy based on CFS(d) is a better than 94–search-competitive online geometric search algorithm for blind agents in undirected graphs with arbitrary length edges.*

Proof. By Lemma 8, CFS(d) is depth restrictable with $\beta = 1+\alpha$ and $C_\beta = 4+\frac{8}{\alpha}$. Thus, by Theorem 3 the doubling strategy is $4\beta C_\beta$-competitive, and we have:

$$4\cdot\beta\cdot C_\beta = 4\cdot(1+\alpha)\cdot\left(4+\frac{8}{\alpha}\right) = 48+16\alpha+\frac{32}{\alpha}\,.$$

Simple analysis shows that this factor is minimal for $\alpha=\sqrt{2}$, yielding a factor of $93.254\ldots$. $\square$

4.2.4 Offline Searching

In the offline setting, the searcher knows the graph, but it still does not know the location of the target. Thus, we want to compute (or approximate) an optimal search path in a known graph.

Computing an optimal search path in a known graph is NP-hard [21], but we can use our framework to give an approximation. Given a graph $G=(V,E)$ and an exploration depth $d \geq 1$ we can compute the subgraph $G(d)$. Now, we have to find an appropriate exploration strategy for $G(d)$. In the case of a vertex search, exploring $G(d)$ amounts to finding a Traveling Salesperson Tour on $G(d)$. This problem is NP-hard, too, but we can approximate a TSP-Tour within factor $C_\beta = 2$ using the mimimum–spanning-tree heuristic,[6] or we can use one of the $1+\epsilon$ approximations by Grigni et al. [16], Arora [3], or Mitchell [22].

The problem of finding a minimum-length tour that visits every edge of a given graph at least once is known as the Chinese Postman Problem, and can be solved in polynomial time for graphs that are either directed or undirected [12, 23]. In this case, we have $C_\beta = \beta = 1$. Altogether we have

Theorem 10 *There is a 4–search-competitive strategy for offline geometric search and a 8–search-competitive strategy for offline vertex search.*

[6]Note that we cannot apply the Christofides heuristic [7], because the triangle-inequality is not fulfilled in arbitrary graphs.

5 Searching Polygons

5.1 Simple Polygons

A simple polygon, P, is given by a closed, nonintersecting polygonal chain. Our searcher is equipped with ideal, unlimited vision; that is, it is provided with the full *visibility polygon* with respect to the searchers current position.

To apply our framework, we need a depth-restrictable online exploration algorithm. The best known algorithm, PolyExplore, for the online exploration of a simple polygon by Hoffmann et al. [19] achieves a competitive ratio of 26.5.

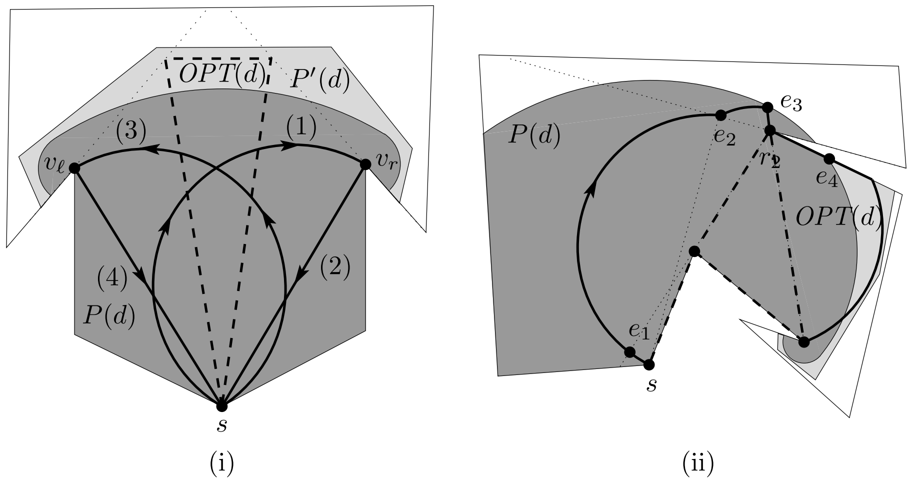


Figure 4: (i) PolyExplore(d) explores the right reflex vertex v_r along a circular arc (1), returns to the start (2) and explores the left reflex vertex v_ℓ likewise (3)+(4). OPT(d) (dashed line) leaves $P(d)$.
(ii) PolyExplore(d) leaves $P(d)$ whereas the shortest exploration path for $P(d)$ lies inside $P(d)$. In both cases, we can extend $P(d)$ (dark gray) to $P'(d)$ (light gray) containing both PolyExplore(d) and OPT(d).

Let $P(d) \subseteq P$ denote the part of the polygon P where all points have a distance at most d from the start. We can modify PolyExplore to explore $P(d)$: During the exploration, an unseen part of P always lies behind a *cut* c_v emanating from a reflex vertex, v. These reflex vertices are called *unexplored* as long as we have not visited the corresponding cut c_v. The algorithm 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 PolyExplore(d), unexplored reflex vertices in a distance greater than d from the start are simply ignored; that is, although they may be detected they will not be inserted into the list. Let OPT(d) be the shortest path that sees all points in $P(d)$.

Note that both PolyExplore(d) and OPT(d) may exceed $P(d)$ as shown in Figure 4. In (i) PolyExplore(d) explores successively the vertices v_r and v_ℓ, but OPT(d) visits the cuts outside $P(d)$. In (ii) PolyExplore(d) leaves $P(d)$ in e_4. However, we can enlarge $P(d)$ to $P'(d)$ by a convex region, so that the resulting polygon contains PolyExplore(d) as well as OPT(d), see Figure 4. Because we add no reflex vertices, we do not change the paths of OPT(d) and PolyExplore(d), even if the extensions for different parts of $P(d)$ overlap. Thus, the analysis of PolyExplore by Hoffmann et al. still holds in $P'(d)$ and we have

Lemma 11 *In a simple polygon, PolyExplore is a depth-restrictable online exploration algorithm with $\beta = 1$ and $C_\beta = 26.5$.* □

Theorem 12 *The doubling strategy based on PolyExplore(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 11 and Theorem 3.

If we know the polygon, we can compute OPT(d) in polynomial time by adapting a corresponding algorithm for P. Every known polynomial time offline exploration algorithm visits the essential cuts in a certain sequence, see for example [6, 26, 25, 10]. Any of these algorithms can be used in our framework. As an optimal algorithm has approximation factor $C = 1$, our framework yields an approximation of the optimal search path with a factor of 8.

The overall running time of the algorithm seems to depend on the distance to the farthest reflex vertex of the polygon. However, we skip a step with distance 2^i if there is no reflex vertex within a distance between 2^{i-1} and 2^i. Thus, we always explore at least one new vertex in every iteration of the doubling strategy. Altogether, the total running time is bounded by a polynomial in the number of the vertices of P. □

Note that there is a considerable gap between the upper bound given by Hoffmann el al. [19] and best known lower bound of 1.2825 [18]. The authors conject that the actual performance of PolyExplore is below 10 [19]; the worst case known to so far is 5 [17]. Under this assumption, the search-competitivity of a doubling strategy based on PolyExplore can be expected to be below 80.

Now the question arises whether there is a polynomial time algorithm that computes the optimal search path in a simple polygon. We have to visit every essential cut, so we can try to visit them in any possible order. Anyway, we do not know exactly which point on the cut we should visit. We are not sure whether there are only a few possibilities as in the shortest watchman route problem. In other words, it is unknown whether this subproblem is discrete at all. So the problem of computing an optimal search path in a polygon is still open.

Even for rectilinear simple polygons no polynomial time algorithm for the optimal search path is known, but we can find better online algorithms:

Theorem 13 *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.*

Proof. For a rectilinear simple polygon, P, Papadimitriou et al. [8] introduced a simple $\sqrt{2}$-competitive online exploration algorithm. This algorithm is depth restrictable—we simply ignore reflex vertices farther away than d—while remaining $\sqrt{2}$-competitive compared to the restricted optimal path OPT(d). The optimal path never leaves $P(d)$, because we have only 90° reflex vertices. Our framework gives a $8\sqrt{2}$–search-competitive online search algorithm.

In the offline setting, we can obtain a polynomial time 8–search-competitive search algorithm based on an optimal depth-restricted exploration algorithm similar to Theorem 12. □

5.2 Polygons with Holes

In this section, we show that there is no constant–search-competitive online search algorithm for polygons with (rectangular) holes. Albers et al. [1] showed that there is no constant-competitive online exploration algorithm for polygons with holes. They filled a rectangle of height k and

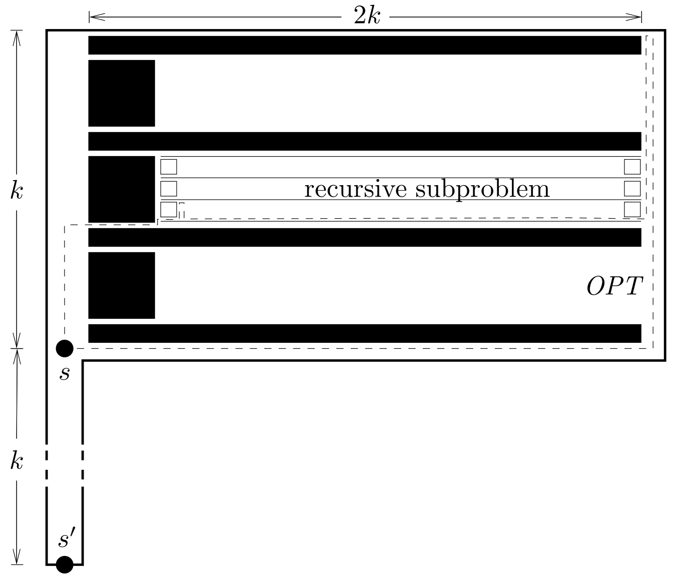


Figure 5: Lower bound construction for approximating the optimal search path in polygons with holes (start point s'). The upper half shows the lower bound for the exploration task by Albers et al. (start point s).

width $2k$, $k \geq 2$, with $O(k^2)$ rectangular holes such that the optimal exploration tour has a length in $O(k)$, whereas any online exploration algorithm needs to travel a distance in $\Omega(k^2)$. The details of the construction are not important here; just note that every point $p \in \mathcal{E}$ has at most the distance $3k$ from the start point, s, see Figure 5.

Theorem 14 *For an agent with vision in a polygon with holes there is no constant–search-competitive online search algorithm.*

Proof. Unfortunately, in the lower bound construction of Albers et al. the optimal exploration paths yields already a bad search ratio. Thus, we enlarge the setting by a thin corridor of length k that leads to the former start point, s. Our new start point, s', is located at the end of the new corridor, see Figure 5. Now, every point that is not visible from s' is at least k steps away from s'; that is, $\text{sp}(s', p) \geq k$ holds for such a point p. The optimal exploration path is still never longer than $C \cdot k$ for a constant C; thus, the optimal exploration path is a C-approximation of the optimal search path. In the new scene, every online exploration algorithm is forced to walk a path of length in $\Omega(k^2)$. Because every online approximation of the optimal search path is also an online exploration algorithm, there are points that are discovered after walking a path length in $\Omega(k^2)$, although their distance to s' is in $O(k)$. Thus, no online approximation is able to achieve a constant approximation factor. □

Since the offline exploration problem is NP-complete (by straightforward reduction from planar TSP) we cannot use our framework to get 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 $\text{OPT}(d)$ in any order to find the best one. Applying our framework gives an approximation factor of 8 for the optimal search ratio.

The results of Koutsoupias et al. [21] imply that the offline problem of computing an optimal search path in a known polygon with holes is NP-complete.

6 A General Lower Bound

We have seen that for certain types of environments there exists an approximation for the optimal search path if there exists a depth-restrictable, competitive exploration strategy. Further, we have seen that polygons with holes are hard searchable. Now, we want to generalize the latter result; that is, we want to show that—under a certain condition—there is no approximation up to a constant factor if there is no competitive exploration strategy for environments of the given type.

Usually, the nonexistence of competitive exploration strategies is shown by giving a lower bound—a scenario, in which every exploration strategy is forced to walk a path whose length exceeds the length of the optimal exploration path by more than a constant factor. To transfer such a result to search path approximations, we require that the scenario can be extended around the start point, such that the start point moves further away from the original scenario. We used this technique in Section 5.2.

Definition 15 Let $\mathcal{E}$ be an environment of arbitrary type, and s be a start point in $\mathcal{E}$. We call $\mathcal{E}$ *s-extendable* if we can enlarge $\mathcal{E}$ locally around the start point; that is, it is possible to choose a new start point, s', outside $\mathcal{E}$, and enlarge $\mathcal{E}$ to $\mathcal{E}'$, such that s' is contained in $\mathcal{E}'$ and every path from s' to a point in $\mathcal{E}$ passes s.

Theorem 16 *If there is no constant-competitive online exploration algorithm for environments of a given type, and the corresponding lower bound is s-extendable, then there is no competitive online approximation of the optimal search path.*

Proof. Let $\mathcal{S}$, $|\mathcal{S}| = n$, be the lower bound construction, OPT denote the optimal exploration algorithm, and $f(n)$ be a function, so that $|\text{OPT}| \in O(f(n))$ holds. There is no competitive online exploration, so $|\mathcal{A}| \in \omega(f(n))$ holds for every online algorithm $\mathcal{A}$.

Because any online approximation of the optimal search path is also an online exploration strategy, any online approximation strategy in $\mathcal{S}$ can be forced to detect the last point, p, after traversing a path with a length in $\omega(f(n))$.

We construct a lower bound, $\mathcal{S}'$, for the online approximation of the search path by placing a new start point, s', *outside* $\mathcal{S}$ with distance $f(n)$ and connecting it to the former start point s. In $\mathcal{S}'$ any online search strategy can also be forced to detect the last point, p, after moving a path in $\omega(f(n))$, whereas the distance from p to s' is still in $O(f(n))$. Therefore, the search ratio of any online search path in $\mathcal{S}'$ is in $\Omega(n)$. On the other hand, the optimal exploration path in $\mathcal{S}'$ is still in $O(f(n))$, yielding—of course—a constant optimal search ratio. Altogether, no constant-competitive online approximation exists. $\square$

7 Conclusion and Open Problems

There are environments where no online search strategy can achieve a constant competitive factor. Therefore, we used the *search ratio* as a parameter of a given environment that gives a measure for the environment's searchability. A search strategy is considered "good" if it achieves a good approximation of the optimal search ratio; that is, the search ratio of an online strategy is at most a constant factor worse than the optimal search ratio.

We showed that we can use depth-restrictable exploration strategies—exploration strategies that can be modified to explore the environment only up to a certain depth while they are still competitive—to approximate the optimal search path by successively applying the exploration with exponentially increasing exploration depths. For blind agents we showed that there are $4\beta C_\beta$-approximations, for searchers with vision $8\beta C_\beta$-approximations, where β and

C_β are parameters that depend on the modifications to turn an exploration algorithm into a depth-restricted exploration. We applied our results to various types of graphs and polygons, see Table 7.

Further, we showed that there is no constant–search-competitive strategy for polygons with holes. The main idea for this proof—enlarging the environment close to the start point—can be generalized for environments that fulfill a certain condition we called s-extendable. We also showed that some graph settings—including directed graphs—are hard searchable.

Altogether, we showed a close relation between searching and exploring: For environments fulfilling $|\,\mathrm{sp}(s,p)| = |\,\mathrm{sp}(p,s)|$ for all p in $\mathcal{E}$ there is an equivalence between constant-competitive exploring and searching if the exploration strategy is depth restrictable and the lower bounds are s-extendable. Naturally, these results lead to the question if there is a stronger connection. More precisely, can we omit the prerequisites 'depth restrictable' and 's-extendable', and show the following conjecture?

Conjecture 17 *For a given type of environments that fulfills $\forall p \in \mathcal{E} : |\,\mathrm{sp}(s,p)| = |\,\mathrm{sp}(p,s)|$, there is a constant–search-competitive strategy* ***if and only if*** *there exists a constant-competitive online exploration for environments of this type.*

Proving this conjecture would show a closer relation between exploration and searching: We are able to approximate the optimal search path—with other words, we can find a good search strategy—if there is a constant-competitive exploration strategy. And, vice versa, we have no chance to find a good search strategy if no constant-competitive exploration is possible. Note that the sp-condition is necessary, anyway. Not only because our approximation framework relies on it, it also seems to be hard to find depth-restrictable exploration strategies for such environments. For example, there is a competitive exploration, yet not depth restrictable, for directed graphs.

Table 1: Summary of our approximation results. The entry marked with * had earlier been proven by Koutsoupias et al. [21]. They had also shown that computing the optimal search path is NP-complete for (planar) graphs. It is also NP-complete for polygons with holes, whereas it is not known to be NP-complete for trees and polygons without holes.

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	205.192 . . .	8
Undirected graph	unit, arbitrary	vertex	no search-competitive alg.	8*
Undirected graph	arbitrary	geometric	93.254 . . .	4
Simple polygon			212	8
Rect. simple polygon			$8\sqrt{2}$	8
Polygon with holes			no search-competitive alg.	?

References

[1] S. Albers, K. Kursawe, and S. Schuierer. Exploring unknown environments with obstacles. In *Proc. 10th ACM-SIAM Sympos. Discrete Algorithms*, pages 842–843, 1999.

[2] S. Alpern and S. Gal. *The Theory of Search Games and Rendezvous*. Kluwer Academic Publications, 2003.

[3] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998.

[4] R. Baeza-Yates, J. Culberson, and G. Rawlins. Searching in the plane. *Inform. Comput.*, 106:234–252, 1993.

[5] P. Berman. On-line searching and navigation. In A. Fiat and G. Woeginger, editors, *Competitive Analysis of Algorithms*. Springer-Verlag, 1998.

[6] W.-P. Chin and S. Ntafos. Shortest watchman routes in simple polygons. *Discrete Comput. Geom.*, 6(1):9–31, 1991.

[7] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. In J. F. Traub, editor, *Sympos. on New Directions and Recent Results in Algorithms and Complexity*, page 441, New York, NY, 1976. Academic Press.

[8] X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment I: The rectilinear case. *J. ACM*, 45(2):215–245, 1998.

[9] X. Deng and C. H. Papadimitriou. Exploring an unknown graph. *Journal of Graph Theory*, 32:265–297, 1999.

[10] M. Dror, A. Efrat, A. Lubiw, and J. S. B. Mitchell. Touring a sequence of polygons. In *Proc. 35th Annu. ACM Sympos. Theory Comput.*, pages 473–482, 2003.

[11] C. A. Duncan, S. G. Kobourov, and V. S. A. Kumar. Optimal constrained graph exploration. In *Proc. 12th ACM-SIAM Symp. Discr. Algo.*, pages 307–314, 2001.

[12] J. Edmonds and E. L. Johnson. Matching, euler tours and the chinese postman. *Math. Prog.*, 5:88–124, 1973.

[13] R. Fleischer, T. Kamphans, R. Klein, E. Langetepe, and G. Trippen. Competitive online approximation of the optimal search ratio. In *Proc. 12th Annu. European Sympos. Algorithms*, volume 3221 of *Lecture Notes Comput. Sci.*, pages 335–346. Springer-Verlag, 2004.

[14] R. Fleischer, K. Romanik, S. Schuierer, and G. Trippen. Optimal robot localization in trees. *Information and Computation*, 171:224–247, 2001.

[15] R. Fleischer and G. Trippen. Exploring an unknown graph efficiently. In *Proc. 13th Annu. European Sympos. Algorithms*, volume 3669 of *Lecture Notes Comput. Sci.*, pages 11–22. Springer-Verlag, 2005.

[16] M. Grigni, E. Koutsoupias, and C. H. Papadimitriou. An approximation scheme for planar graph TSP. In *Proc. 36th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 640–645, 1995.

[17] R. Hagius. Untere Schranken für das Online-Explorationsproblem. Diplomarbeit, FernUniversität Hagen, Fachbereich Informatik, Mai 2002.

[18] R. Hagius, C. Icking, and E. Langetepe. Lower bounds for the polygon exploration problem. In *Abstracts 20th European Workshop Comput. Geom.*, pages 135–138. Universidad de Sevilla, 2004.

[19] F. Hoffmann, C. Icking, R. Klein, and K. Kriegel. The polygon exploration problem. *SIAM J. Comput.*, 31:577–600, 2001.

[20] R. Klein. *Algorithmische Geometrie*. Springer, Heidelberg, 2nd edition, 2005.

[21] E. Koutsoupias, C. H. Papadimitriou, and M. Yannakakis. Searching a fixed graph. In *Proc. 23th Internat. Colloq. Automata Lang. Program.*, volume 1099 of *Lecture Notes Comput. Sci.*, pages 280–289. Springer, 1996.

[22] J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM J. Comput.*, 28:1298–1309, 1999.

[23] C. H. Papadimitriou. On the complexity of edge traversing. *J. ACM*, 23:544–554, 1976.

[24] S. Schuierer. On-line searching in simple polygons. In H. Christensen, H. Bunke, and H. Noltemeier, editors, *Sensor Based Intelligent Robots*, volume 1724 of *LNAI*, pages 220–239. Springer Verlag, 1997.

[25] 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.

[26] 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.