# Exploring Simple Grid Polygons

Christian Icking[1], Tom Kamphans[2], Rolf Klein[2], and Elmar Langetepe[2]

[1] University of Hagen, Praktische Informatik VI, 58084 Hagen, Germany.
[2] University of Bonn, Computer Science I, Römerstraße 164, 53117 Bonn, Germany.

**Abstract.** We investigate the online exploration problem of a short-sighted mobile robot moving in an unknown cellular room without obstacles. The robot has a very limited sensor; it can determine only which of the four cells adjacent to its current position are free and which are blocked, i. e., unaccessible for the robot. Therefore, the robot must enter a cell in order to explore it. The robot has to visit each cell and to return to the start. Our interest is in a short exploration tour, i. e., in keeping the number of multiple cell visits small. For abitrary environments without holes we provide a strategy producing tours of length $S \leq C + \frac{1}{2}E - 3$, where $C$ denotes the number of cells—the area—, and $E$ denotes the number of boundary edges—the perimeter—of the given environment. Further, we show that our strategy is competitive with a factor of $\frac{4}{3}$, and give a lower bound of $\frac{7}{6}$ for our problem. This leaves a gap of only $\frac{1}{6}$ between the lower and the upper bound.

**Key words:** Robot navigation, exploration, covering, online algorithms, competitive analysis, lower bounds, grid polygons

## 1 Introduction

Exploring an unknown environment and searching for a target in unknown position are among the basic tasks of autonomous mobile robots. Both problems have received a lot of attention in computational geometry and in robotics; see e. g. [3, 5, 9, 12, 13, 17].

We use a simple model for the robot and its environment: the robot is short-sighted, and the surrounding is subdivided by a rectangular integer grid, similar to a chessboard. Essentially, there are two motivations for using this model instead of a robot with a full vision system: First, even a laser scanner has a reliable range of only a few meters. Hence, the robot has to move towards more distant areas in order to explore them. Second, service robots like lawn mowers or cleaners need to get close to their work areas. The robot's sensors provide the information, which of the four neighbors of the currently occupied cell do not belong to the polygon and which ones do. The robot can enter the latter cells. The robot's task is to visit every cell inside the polygon and to return to the start cell. Sometimes, this task in also called *covering*.

Even though our robot does not know its environment in advance it is interesting to ask how short a tour can be in the offline situation, i. e., when the environment is already known. This amounts to constructing a shortest traveling salesperson tour on the free cells.

If the polygonal environment contains obstacles, the problem of finding such a minimum length tour is known to be NP-hard, see Itai et al. [14]. There are $1+\varepsilon$ approximation schemes by Grigni et al. [7], Arora [2], and Mitchell [16], and a $\frac{53}{40}$ approximation by Arkin et al. [1].

In a polygon without obstacles, the complexity of constructing offline a minimum length tour seems to be open. Ntafos [18] and Arkin et al. [1] have shown how to approximate the minimum length tour with factors of $\frac{4}{3}$ and $\frac{6}{5}$, respectively. Umans and Lenhart [19] have provided an $O(C^4)$ algorithm for deciding if there exists a Hamiltonian cycle, i. e., a tour that visits each of the $C$ cells of a polygon *exactly* once. For the related problem of Hamiltonian paths, Everett [4] has given a polynomial algorithm for certain grid graphs.

In this paper our interest is in the online version of the cell exploration problem. Exploring a grid polygon *with* holes was considered by Icking et al. [10, 11] and independently by Gabriely and Rimon [6]. Icking et al. showed a lower bound of 2 for this problem and introduced an exploration strategy that needs no more than $C+\frac{1}{2}E+3H+W-2$ steps,[3] see [15], where $C$ denotes the number of cells, $E$ the number of boundary edges, $H$ the number of holes and $W$ is a measure for the windings of the polygon. Gabriely and Rimon showed an upper bound of $C+B$, where $B$ denotes the number of boundary cells.

We consider the exploration of polygons *without* holes. Although both problems seem to be closely related there is an important difference: We have a lower bound of 2 for polygons with holes, but it turns out that we can do much better in simple polygons.

An upper bound for our exploration strategy is given in terms of the polygon's *area*, $C$, and the *perimeter*, $E$. While $C$ is the number of free cells, $E$ is the number of edges between a free cell and a blocked cell, see for example Fig. 1. We use $E$ to distinguish between skinny and thick environments. For thick environments, $E \in O(\sqrt{C})$ holds; thus, the number of additional cell visits is substantially smaller than $C$. Only in polygons that do not contain any $2 \times 2$-square of free cells, $E$ achieves its maximum value of $2(C+1)$, and our upper bound is equal to $2C-2$, but in this case one cannot do better, since even the optimal offline strategy needs that number of steps.

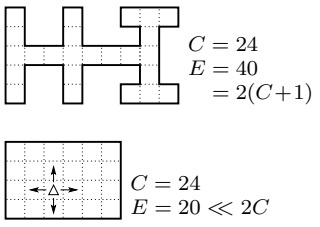

**Fig. 1.** The perimeter, $E$, to distinguish between 'thin' and 'thick' environments.

Our paper is organized as follows: in Sect. 2 we give more detailled description of our robot and the environment. We give a lower bound for our problem in Sect. 3. In Sect. 4 we present an exploration strategy, SmartDFS. The analysis shows in Sect. 5 that this strategy uses no more than $C+\frac{1}{2}E-3$ steps and is in fact competitive with a factor of $\frac{4}{3}$.
SmartDFS was implemented in a Java-Applet available in the internet, see [8].

[3] We assume that the cells have unit size, so the length of the path is equal to the number of steps from cell to cell.

## 2 Definitions

We consider a simple model for the environment of the robot: the robot moves in a surrounding with a grid structure. More precisely, a *cell* is a basic block in our environment, defined by a pair $(x, y) \in \mathbb{N}^2$. A cell is either *free* and can be visited by the robot, or *blocked*, i. e., unaccessible for the robot. We call two cells *adjacent*, if they share a common edge, and *touching*, if they share a common edge or corner. A *grid polygon*, $P$, is a connected set of free cells. A polygon without blocked cells inside its boundary is called *simple*. From its current position, the robot can find out which of the adjacent cells are free and which are blocked, and it can move in one *step* to an adjacent free cell, see Fig. 1. The robot has enough memory to store a map of known cells.

## 3 A Lower Bound

**Theorem 1.** *Every strategy for the exploration of a simple grid polygon with $C$ cells needs at least $\frac{7}{6}C$ steps.*

*Proof.* We assume that the robots starts in a corner of the polygon, see Fig. 2(i). W. l. o. g. we assume that the strategy decides to walk one step to the east. For the second step, the strategy has two possibilities: either it leaves the wall with a step to the south, see Fig. 2(ii), or it continues to follow the wall with a further step to the east, see Fig. 2(iii). In the first case, we close the polygon as shown in Fig. 2(iv). The robot needs at least 8 steps to explore this polygon, but the optimal strategy needs only 6 steps yielding a factor of $\frac{8}{6}$. In the second case we proceed as follows. If the robot leaves the boundary, we close the polygon as shown in Fig. 2(v) and (vi). The robot needs 12 step, but 10 steps are sufficient. In the most interesting case, the robot still follows the wall, see Fig. 2(vii). In this case, the robot needs at least 28 steps to explore this polygon, whereas an optimal strategy needs only 24 steps. Thus, we achieve a factor of $\frac{7}{6}$.

We can easily extend this pattern to polygons of arbitrary size by repeating the construction using the 'entry' and 'exit' cells denoted by the arrows in Fig. 2(iv)–(vii). This construction cannot lead to overlapping polygons or polygons with holes, since the polygon always extends to the same direction. □

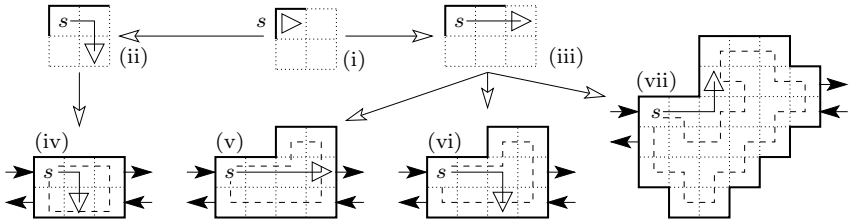


**Fig. 2.** A lower bound for the exploration of simple polygons. The dashed lines show the optimal solution, △ denotes the robot's position.

## 4 An Exploration Strategy

As a first approach, we can apply a simple depth-first search algorithm (DFS): The polygon is explored following the left-hand rule, i. e., for every entered cell the robot tries to continue its path to an adjacent and unexplored cell, preferring a step to the left over a straight step over a step to the right. This results in a complete exploration, but takes $2C - 2$ steps. Since the shortest tour needs at least $C$ steps, DFS turns out to be 2-competitive. However, there is no reason to visit *each* cell twice just because this is required in some special situations like dead ends of width 1. In the following, we introduce two improvements to DFS.

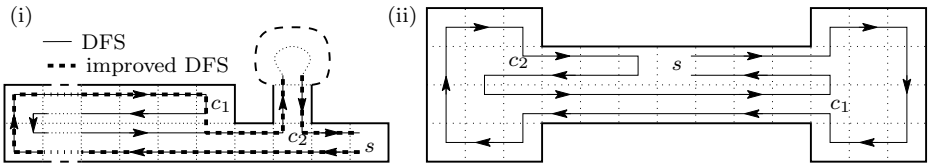


**Fig. 3.** Improvement to DFS: (i) optimize return path, (ii) detect polygon splits.

The first improvement is to return directly to those cells that have unexplored neighbors. See e. g. Fig. 3(i): DFS walks from $c_1$ to $c_2$ through the completely explored corridor. A more efficient strategy walks on a shortest path—on cells that are already known—from $c_1$ to $c_2$.

Now, observe the polygon shown in Fig. 3(ii). With DFS, the robot walks four times through the narrow corridor. A more clever solution explores the right part immediately after the first visit of $c_1$, and continues with the left part, resulting in only two visits. The cell $c_1$ has the property that the graph of unvisited cells splits into two components after $c_1$ is explored. We call cells like this *split cells*. The second improvement is to recognize and handle split cells, see Sect. 5. The following description of our strategy, SmartDFS, resumes both improvements to DFS, see Fig. 5 for an example.

**SmartDFS(**$P$**,** *start***):**
    Choose direction *dir*, such that
      reverse(*dir*) is a blocked cell;
    ExploreCell(*dir*);
    Walk on the shortest path to *start*;

**ExploreStep(**$base$**,** $dir$**):**
    **if** unexplored($base$, $dir$) **then**
        Walk on shortest path to $base$;
        move($dir$);
        ExploreCell($dir$);
    **end if**

**ExploreCell(**$dir$**):**
    $base$ := current position;
    **if** not isSplitCell($base$) **then**
        ExploreStep($base$, ccw($dir$));
        ExploreStep($base$, $dir$);
        ExploreStep($base$, cw($dir$));
    **else**
        Choose different order, see
          Sect. 5.
    **end if**

## 5 The Analysis of SmartDFS

SmartDFS explores the polygon in layers, beginning with the cells along the boundary of $P$ and proceeding towards the interior of $P$.

**Definition 2.** *Let $P$ be a grid polygon. The boundary cells of $P$ uniquely define the* first layer *of $P$. The polygon $P$ without its first layer is called the* 1-offset *of $P$. The $\ell$-th layer and the $\ell$-offset of $P$ are defined successively, see Fig. 4(i).*

**Lemma 3.** *The $\ell$-offset of a simple grid polygon, $P$, has at least $8\ell$ edges less than $P$.*

*Proof.* First, we cut off blind alleys narrower than $2\ell$, since those parts of $P$ do not affect the $\ell$-offset. We walk clockwise around the boundary cells of the remaining polygon, see Fig. 4(i). For every left turn the offset gains at most $2\ell$ edges and for every right turn the offset looses at least $2\ell$ edges. Since, there are four more right turns than left turns, we loose at least $8\ell$ edges. □

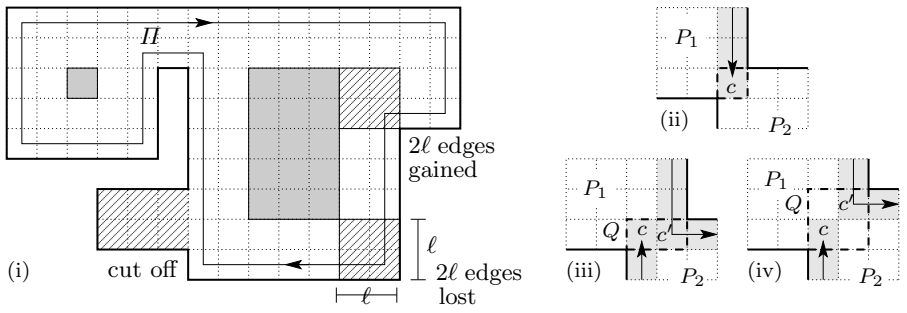


**Fig. 4.** (i) The 2-offset (shaded) of a grid polygon; three examples for split cells, (ii) type (II), (iii) and (iv) type (I).

Definition 2 allows us to specify the handling of a split cell in SmartDFS. Let us consider the situation shown in Fig. 5(i): SmartDFS has just met the first split cell, $c$, in the fourth layer of $P$. $P$ divides into three parts: $P = K_1 \dot\cup K_2 \dot\cup \{\text{ visited cells of } P\}$, where $K_1$ and $K_2$ denote the connected components of the unvisited cells. In this case it is reasonable to explore the component $K_2$ first since the start cell $s$ is closer to $K_1$.

We use the layer numbers to decide which component we have to visit at last. Whenever a split cell occurs in layer $\ell$, every component is one of the following types, see Fig. 4(ii)–(iv): (I) $K_i$ is *completely* surrounded by layer $\ell$,[4] (II) $K_i$ is *not* surrounded by layer $\ell$, or (III) $K_i$ is *partially* surrounded by layer $\ell$.

In any case, it is the best choice to explore the component of type (III) at last. Note that it may occur that three components arise at a split cell, but we can handle this case as two successive splits occuring at the same split cell.

[4] More precisely, the part of layer $\ell$ that surrounds $K_i$ is completely visited. For convenience, we will use slightly sloppy, but shorter form.

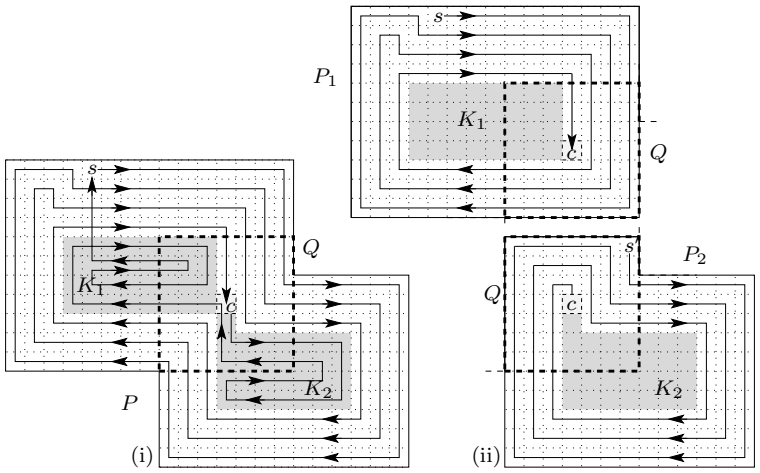


**Fig. 5.** A decomposition of $P$ at the split cell $c$ and its handling in smartDFS.

For the analysis we consider two polygons, $P_1$ and $P_2$, as follows. Let $Q$ be the square of width $2q+1$ around $c$ with $q := \begin{cases} \ell, & \text{if } K_2 \text{ is of type (I)} \\ \ell - 1, & \text{if } K_2 \text{ is of type (II)} \end{cases}$, where $K_2$ denotes the component that is explored first, and $\ell$ denotes the layer in which the split cell was found. We choose $P_2 \subset P \cup Q$, such that $K_2 \cup \{c\}$ is the $q$-offset of $P_2$, and $P_1 := ((P \backslash P_2) \cup Q) \cap P$, see Fig. 5. The intersection with $P$ is necessary, since $Q$ may exceed the boundary of $P$.

The choice of $P_1, P_2$ and $Q$ ensures that the robot's path in $P_1 \backslash Q$ and in $P_2 \backslash Q$ do not change compared to the path in $P$. The parts of the robot's path that lead from $P_1$ to $P_2$ and from $P_2$ to $P_1$ are fully contained in the square $Q$. Just the parts inside $Q$ are bended to connect the appropriate paths inside $P_1$ and $P_2$, see Fig. 5.

We want to visit every cell in the polygon and to return to $s$. Every strategy needs at least $C(P)$ steps to fulfill this task. Thus, we can split the overall length of the exploration path $\Pi$ into two parts, $C(P)$ and excess$(P)$, with $|\Pi| = C(P) + \text{excess}(P)$. Since SmartDFS recursively explores $K_2 \cup \{c\}$, we want to apply the upper bound inductively to the component $K_2 \cup \{c\}$. The following lemma gives us the relation between the path lengths in $P$ and the path lengths in the two components.

**Lemma 4.** *Let $P$ be a simple grid polygon. Let the robot visit the first split cell, $c$, which splits the unvisited cells of $P$ into two components $K_1$ and $K_2$, where $K_2$ is of type (I) or (II). With the preceding notions we have* $\text{excess}(P) \leq \text{excess}(P_1) + \text{excess}(K_2 \cup \{c\}) + 1$.

*Proof.* Since $c$ is the first split cell, there is no excess in $P_2 \backslash (K_2 \cup \{c\})$ and it suffices to consider excess$(K_2 \cup \{c\})$ for this part. After $K_2 \cup \{c\}$ is finished, the robot starts at $c$ and explores $K_1$. For this part we take excess$(P_1)$ into account. Finally, we add one single step, because the split cell $c$ is visitited twice:

once, when SmartDFS detects the split and once more after the exploration of excess$(K_2 \cup \{c\})$ is finished. Altogether, the given bound is achieved. □

The following lemma can easily be shown and allows us to charge the number of edges in $P_1$ and $P_2$ against the number of edges in $P$ and $Q$.

**Lemma 5.** *Let $P$ be a simple grid polygon, and let $P_1, P_2$ and $Q$ be defined as above. The number of edges satisfy $E(P_1) + E(P_2) = E(P) + E(Q)$.*

**Lemma 6.** *Let $\Pi$ be the shortest path between two cells in a grid polygon $P$. The length of $\Pi$ is bounded by $|\Pi| \leq \frac{1}{2}E(P) - 2$.*

*Proof.* The maximal distance is achieved between two cells in the first layer, and the shortest path between them is never longer than $\frac{1}{2}\cdot\#$(cells in the first layer). Analogously to Lemma 3, this layer has at most $E(P) - 4$ cells. □

Now, we can give an upper bound for the number of steps used to explore a simple polygon.

**Theorem 7.** *Let $P$ be a simple grid polygon with $C$ cells and $E$ edges. $P$ can be explored with $S \leq C + \frac{1}{2}E - 3$ steps. This bound is tight.*

*Proof.* $C$ is the number of cells and thus a lower bound on the number of steps that are needed to explore the polygon $P$. We will show by induction on the number of components that excess$(P) \leq \frac{1}{2}E(P) - 3$ holds.

For the induction base we consider a polygon without any split cell, i. e., SmartDFS visits all cells and returns on the shortest path to the start cell. Since there is no polygon split, all cells of $P$ can be visited by a path of length $C - 1$. By Lemma 6 the shortest path back to the start cell is not longer than $\frac{1}{2}E - 2$ and excess$(P) \leq \frac{1}{2}E(P) - 3$ holds.

Now, we assume that there is more than one component during the application of SmartDFS. Let $c$ be the first split cell detected in $P$. When SmartDFS reaches $c$, two new components, $K_1$ and $K_2$, occur. We consider the two polygons $P_1$ and $P_2$ defined as above using the square $Q$ around $c$.

W. l. o. g. we assume that $K_2$ is recursively explored first. After $K_2$ is completely explored, SmartDFS proceeds with the remaining polygon. As shown in Lemma 4 we have excess$(P) \leq$ excess$(P_1)+$excess$(K_2 \cup \{c\})+1$ . Now, we apply the induction hypothesis to $P_1$ and $K_2 \cup \{c\}$ and get

$$\text{excess}(P) \leq \tfrac{1}{2}E(P_1) - 3 + \tfrac{1}{2}E(K_2 \cup \{c\}) - 3 + 1\,.$$

By applying Lemma 3 to the $q$-offset $K_2 \cup \{c\}$ of $P_2$ we achieve
excess$(P) \leq \frac{1}{2}E(P_1) - 3 + \frac{1}{2}(E(P_2) - 8q) - 3 + 1 = \frac{1}{2}(E(P_1) + E(P_2)) - 4q - 5$ .
From Lemma 5 we conclude $E(P_1) + E(P_2) \leq E(P) + 4(2q + 1)$. Thus, we get excess$(P) \leq \frac{1}{2}E(P) - 3$. This bound is achieved exactly in polygons that do not contain any $2 \times 2$-square of free cells. □

So far we have shown an upper bound for the number of steps needed to explore a polygon that depends on the number of cells and edges in the polygon. Now we want to analyze SmartDFS in the competitive framework.

Corridors of width 1 or 2 play a crucial role in the following, so we refer to them as *narrow passages*.[5] It is easy to see that narrow passages are explored optimally. In passages of width 1 both SmartDFS and the optimal strategy visit every cell twice, and in the other case both strategies visit every cell exactly once. We need two lemmata to show a competitive factor for SmartDFS. The first one gives us a relation between the number of cells and the number of edges for a special class of polygons.

**Lemma 8.** *For a simple grid polygon, $P$, without any narrow passage or split cells in the first layer, $E(P) \leq \frac{2}{3} C(P) + 6$ holds.*

*Proof.* Consider such a polygon, $P$, see Fig. 6(i). We successively remove an outer row or column of at least three boundary cells, maintaining our assumptions on $P$. These assumptions ensure that we can always find such a row or column. Thus, we remove at least three cells and at most two edges. This decomposition ends with a $3 \times 3$ block of cells that fulfills $E = \frac{2}{3}C(P) + 6$. Now, we reverse our decomposition, i. e., we successively add all rows and columns until we end up with $P$. In every step, we add at least three cells and at most two edges. Thus, $E \leq \frac{2}{3}C(P) + 6$ is fulfilled in every step. □

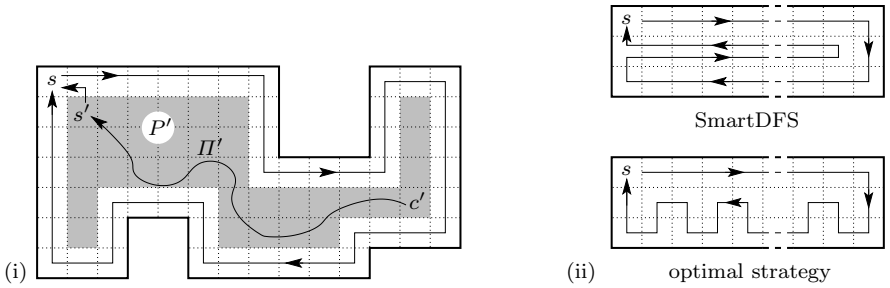


**Fig. 6.** (i) For polygons without narrow passages or split cells in the first layer, $E(P) \leq \frac{2}{3} C(P) + 6$ holds, and the last explored cell, $c'$, lies in the 1-offset, $P'$ (shaded), (ii) In a corridor of width 3 and even length, $S(P) = \frac{4}{3} S_{\mathrm{Opt}}(P) - 2$ holds.

For the same class of polygons, we can show that SmartDFS behaves slightly better than the bound in Theorem 7.

**Lemma 9.** *A polygon of the same type as in Lemma 8 can be explored using no more than $S(P) \leq C(P) + \frac{1}{2}E(P) - 5$ steps.*

*Proof.* We have shown $S(P) \leq C(P) + \frac{1}{2}E(P) - 3$ in Theorem 7. In the proof, we used Lemma 6 to bound the return path, but this lemma bounds the path between two cells in the first layer. By our assumptions on $P$, we can completely explore the first layer of $P$ before visiting another layer, and the return path,

[5] More precisely, a cell, $c$, belongs to a narrow passage, if $c$ can be removed without changing the layer number of any other cell.

$\Pi$, starts in a cell, $c'$, in the 1-offset, $P'$, see Fig. 6(i). Let $s'$ denote the first visited cell in $P'$. Remark that $s$ and $s'$ are at least touching each other. Now, $\Pi$ is bounded by a shortest path, $\Pi'$, from $c'$ to $s'$ in $P'$ and a shortest path from $s'$ to $s$, i. e., $|\Pi| \leq |\Pi|' + 2$. $\Pi'$, in turn, is bounded using Lemma 6 by $|\Pi|' \leq \frac{1}{2}E(P') - 2$. With Lemma 3, $E(P') \leq E(P) - 8$ holds, and altogether we get $|\Pi| \leq \frac{1}{2}E(P) - 4$, which is two steps shorter than stated in Lemma 6. □

**Theorem 10.** *The strategy* SmartDFS *is* $\frac{4}{3}$*-competitive.*

*Proof.* Let $P$ be a simple grid polygon. First, we remove all narrow passages from $P$ and get a sequence of (sub-)polygons $P_i$, $i = 1, \ldots, k$, without narrow passages. For every $P_i$, $i = 1, \ldots, k-1$, the optimal strategy in $P$ explores the part of $P$ that corresponds to $P_i$ up to the narrow passage that connects $P_i$ with $P_{i+1}$, enters $P_{i+1}$, and fully explores every $P_j$ with $j \geq i$. Then it returns to $P_i$ and continues with the exploration of $P_i$. Further, we already know that narrow passages are explored optimally. This allows us to consider every $P_i$ separately without changing the competitive factor of $P$.

Now, we observe a (sub-)polygon $P_i$. We show by induction on the number of split cells in the first layer that $S(P_i) \leq \frac{4}{3}C(P_i) - 2$ holds. Note that this is exactly achieved in polygons of size $3 \times m$ with $m$ even, see Fig. 6(ii).

If $P_i$ has no split cell in the first layer, we can apply Lemma 9 and Lemma 8:
$S(P_i) \leq C(P_i) + \frac{1}{2}E(P_i) - 5 \leq C(P_i) + \frac{1}{2}\left(\frac{2}{3}C(P_i) + 6\right) - 5 = \frac{4}{3}C(P_i) - 2\,.$

Two cases occur if we meet a split cell, $c$, in the first layer, see Fig. 4(ii)–(iv). In the first case, the new component was never visited before (type (II)). Here, we define $Q := \{c\}$. The second case occurs, because the robot meets a cell, $c'$, that is in the first layer and touches the current cell, $c$, see for example Fig. 4(iii) and (iv). Let $Q$ be the smallest rectangle that contains both $c$ and $c'$.

Similar to the proof of Theorem 7, we split the polygon $P_i$ into two parts, both including $Q$. Let $P''$ denote the part that includes the component of type (II) or (III), $P'$ the other part. For $|Q| = 1$, see Fig. 4(ii), we conclude $S(P_i) = S(P') + S(P'')$ and $C(P_i) = C(P') + C(P'') - 1$. Applying the induction hypothesis to $P'$ and $P''$ yields $S(P_i) = S(P') + S(P'') \leq \frac{4}{3}C(P_i) + \frac{4}{3} - 4 \;<\; \frac{4}{3}C(P_i) - 2\,.$

For $|Q| \in \{2, 4\}$ we gain some steps by merging the polygons. If we consider $P'$ and $P''$ separately, we count the steps from $c'$ to $c$—or vice versa—in both polygons, but in $P_i$ the path from $c'$ to $c$ is replaced by the exploration path in $P''$. Thus, we have $S(P_i) = S(P') + S(P'') - |Q|$ and $C(P_i) = C(P') + C(P'') + |Q|$. This yields $S(P_i) = S(P') + S(P'') - |Q| = \frac{4}{3}C(P_i) + \frac{1}{3}(|Q| - 6) - 2 < \frac{4}{3}C(P_i) - 2\,.$

An optimal strategy needs $\geq C$ steps, which, altogether, yields a competitive factor of $\frac{4}{3}$. □

## 6 Summary

It turned out that the exploration of simple polygons is easier than the exploration of polygons with holes in terms of competitivity. In contrary to the lower bound of 2 for polygons with holes, we have shown a lower bound of $\frac{7}{6}$ and an upper bound of $\frac{4}{3}$ for simple polygons, leaving a gap of only $\frac{1}{6}$. Additionally, we can also bound the length of an exploration path by $C + \frac{1}{2}E - 3$ which is tight.

## References


[1] E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell. Approximation algorithms for lawn mowing and milling. Technical report, Mathematisches Institut, Universität zu Köln, 1997.

[2] S. Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In *Proc. 37th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 2–11, 1996.

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

[4] H. Everett. Hamiltonian paths in non-rectangular grid graphs. Report 86-1, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, 1986.

[5] A. Fiat and G. Woeginger, editors. *On-line Algorithms: The State of the Art*, volume 1442 of *Lecture Notes Comput. Sci.* Springer-Verlag, 1998.

[6] Y. Gabriely and E. Rimon. Competitive on-line coverage of grid environments by a mobile robot. *Comput. Geom. Theory Appl.*, 24:197–224, 2003.

[7] 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.

[8] U. Handel, C. Icking, T. Kamphans, E. Langetepe, and W. Meiswinkel. Gridrobot—an environment for simulating exploration strategies in unknown cellular areas. Java Applet, 2000. http://www.geometrylab.de/Gridrobot/.

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

[10] C. Icking, T. Kamphans, R. Klein, and E. Langetepe. Exploring an unknown cellular environment. In *Abstracts 16th European Workshop Comput. Geom.*, pages 140–143. Ben-Gurion University of the Negev, 2000.

[11] C. Icking, T. Kamphans, R. Klein, and E. Langetepe. On the competitive complexity of navigation tasks. In H. Bunke, H. I. Christensen, G. D. Hager, and R. Klein, editors, *Sensor Based Intelligent Robots*, volume 2238 of *Lecture Notes Comput. Sci.*, pages 245–258, Berlin, 2002. Springer.

[12] C. Icking, R. Klein, and E. Langetepe. Searching for the kernel of a polygon: A competitive strategy using self-approaching curves. Technical Report 211, Department of Computer Science, FernUniversität Hagen, Germany, 1997.

[13] C. Icking, R. Klein, E. Langetepe, S. Schuierer, and I. Semrau. An optimal competitive strategy for walking in streets. *SIAM J. Comput.*, 33:462–486, 2004.

[14] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM J. Comput.*, 11:676–686, 1982.

[15] T. Kamphans. *Models and Algorithms for Online Exploration and Search*. PhD thesis, University of Bonn, to appear.

[16] 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.

[17] J. S. B. Mitchell. Geometric shortest paths and network optimization. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 633–701. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[18] S. Ntafos. Watchman routes under limited visibility. *Comput. Geom. Theory Appl.*, 1(3):149–170, 1992.

[19] C. Umans and W. Lenhart. Hamiltonian cycles in solid grid graphs. In *Proc. 38th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 496–507, 1997.