

RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT BONN
INSTITUT FÜR INFORMATIK I



Daniel Herrmann¹ Tom Kamphans²
Elmar Langetepe¹

Exploring Simple Triangular and Hexagonal Grid Polygons Online

Technical Report 007
December 2007

¹University of Bonn, Institute of Computer Science, Dept. I, Römerstr. 164,
53117 Bonn, Germany.

²Braunschweig University of Technology, Computer Science, Algorithms Group,
Mühlenpfordtstraße 23, 38106 Braunschweig, Germany

Abstract

We investigate the online exploration problem (aka covering) of a short-sighted mobile robot moving in an unknown cellular environment with hexagons and triangles as types of cells. To explore a cell, the robot must enter it. Once inside, the robot knows which of the 3 or 6 adjacent cells exist and which are boundary edges. The robot's task is to visit every cell in the given environment and to return to the start. Our interest is in a short exploration tour; that is, in keeping the number of multiple cell visits small. For arbitrary environments containing no obstacles we provide a strategy producing tours of length $S \leq C + \frac{1}{4}E - 2.5$ for hexagonal grids, and $S \leq C + E - 4$ for triangular grids. C denotes the number of cells—the area—, E denotes the number of boundary edges—the perimeter—of the given environment. Further, we show that our strategy is $\frac{4}{3}$ -competitive in both types of grids, and we provide lower bounds of $\frac{13}{13}$ for hexagonal grids and $\frac{7}{6}$ for triangular grids.

The strategies were implemented in a Java applet [20] that can be found in

<http://www.geometrylab.de/Gridrobot/>

Key words: Robot motion planning, exploration, covering, online algorithms, competitive analysis, grid graphs

1 Introduction

Exploring an unknown environment is one of the basic tasks of autonomous mobile robots and has received a lot of attention in computational geometry and in robotics; see, for example, [14, 21, 30, 31, 35, 13, 8, 7, 15]—just to mention a few of these works.

For some applications, it is convenient to subdivide the given environment by a regular grid into basic blocks (so-called *cells*). For example, the agent’s vision may be limited and a cell is used as to approach the visibility range. Or the agent has to visit every part of the environment for cleaning or lawn mowing, and a cell is an approximation of the robot’s tool (sometimes, this task is called *covering*). The robot’s position is always given by the cell currently occupied by the robot. From its current position, the robot can enter one of the neighboring *free* cells (i.e., cells that are not blocked by an obstacle). The whole environment is not known in advance—so we are dealing with *online* strategies—, but once inside a cell, the robot knows which neighboring cell is blocked and which one is free. The robot’s task is to visit every free cell inside the given environment and to return to the start. There are only three possible regular tilings of the plane: square, hexagonal, or triangular subdivisions [6]. We call a subdivision of the given environment into squares (hexagons, triangles) a square polygon (hexagonal polygon, triangular polygon; respectively). Hexagonal cells are a matter of particular interest for robots that are equipped with a circular tool such as lawn mowers, because hexagonal grids provide a better approximation for the tool than square grids [3].

In a square polygon with obstacles, the offline problem (i.e., finding a minimum length tour that visits every cell) is known to be NP-hard, by work of Itai et al. [26]. By modeling the environment as a *grid graph* with one vertex for every cell and edges between neighboring cells, we can use $1 + \varepsilon$ approximation schemes for Euclidean TSP by Grigni et al. [19], Arora [5], and Mitchell [29]. For square polygons there is a $\frac{53}{40}$ approximation by Arkin et al. [3].

In a square polygon without obstacles, the complexity of constructing offline a minimum length tour is still open. Ntafos [33] and Arkin et al. [3] have shown how to approximate the minimum length tour with factors of $\frac{4}{3}$ and $\frac{6}{5}$, respectively. Umans and Lenhart [36] 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 (i.e., different start and end positions), Everett [17] has given a polynomial algorithm for certain grid graphs. Cho and Zelikovsky [22] studied *spanning closed trails*. Hamiltonian cycles on triangular and hexagonal grids were studied by Polishuk et al. [34, 4], and Islam et al. [25], see also [2].

In this paper, our interest is in the online version of the cell exploration

problem for *hexagonal* and *triangular* polygons without holes. The task of exploring square polygons with holes was independently considered by Gabriely and Rimon [18] and Icking et al. [24], see also Kamphans [27]. Our exploration strategy is based on the strategy *SmartDFS* by Icking et al. [23] for simple polygons. This strategy is $\frac{4}{3}$ -competitive¹ and the number of steps from cell to cell is bounded by $C + \frac{1}{2}E - 3$, where C denotes the number of cells (i.e., the polygon’s area) and E the number of edges (the polygon’s perimeter). Further, there is a lower bound of $\frac{7}{6}$ on the competitive factor for this problem.

Another online task is the *piecemeal exploration*, where the robot has to interrupt the exploration every now and then so as to return to the start point, for example, to refuel. Piecemeal exploration of grid graphs was studied by Betke et al. [9] and Albers et al. [1]. Note that their objective is to visit every node *and* every edge, whereas we require a complete coverage of only the cells. Subdividing the robot’s environment into grid cells is used also in the robotics community, see, for example, Moravec and Elfes [32], Elfes [16], Bruckstein et al. [10, 11], and Koenig and Liu [28]. See also the survey by Choset [12].

Our paper is organized as follows: In Section 2, we give more detailed description of our explorer and the environment. We give lower bounds on the competitive factor in Section 3. In Section 4, we present an exploration strategy for simple polygons. We analyze the performance of this strategy in hexagonal polygons in Section 4.1 and for triangular polygons in Section 4.2.

2 Definitions

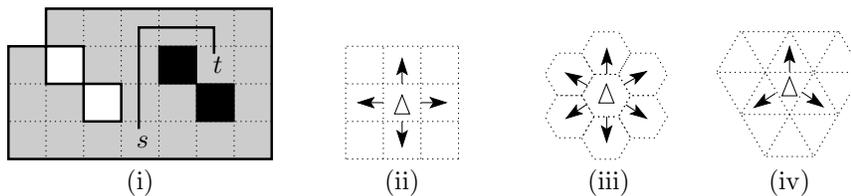


Figure 1: (i) Polygon with 23 cells, 38 edges and one(!) hole (black cells), a path from s to t of length 6 (ii)-(iv) neighboring and touching cells; the agent can determine which of the neighboring cells (marked by an arrow) are free, and enter an adjacent free cell.

Definition 1 We consider polygons that are subdivided by a regular grid. A *cell* is a basic block in our environment. A cell is either *free* and can be

¹That is, the path produced by this online strategy is never longer than $\frac{4}{3}$ times the optimal offline path.

visited by the robot, or *blocked* (i.e., unaccessible for the robot).² We call two cells *adjacent* or *neighboring* if they share a common edge, and *touching* if they share only a common corner.

A *path*, Π , from a cell s to a cell t is a sequence of free cells $s = c_1, \dots, c_n = t$ where c_i and c_{i+1} are adjacent for $i = 1, \dots, n - 1$. Let $|\Pi|$ denote the length of Π . 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 that the robot walks.

A *grid polygon*, P , is a path-connected set of free cells; that is, for every $c_1, c_2 \in P$ exists a path from c_1 to c_2 that lies completely in P . We denote a grid polygon subdivided into square, hexagonal, or triangular cells by P_{\square} , P_{\circ} , or P_{\triangle} , respectively.

We call a set of touching blocked cells that are completely surrounded by free cells an *obstacle* or *hole*; see Figure 1. Polygons without holes are called *simple polygons*.

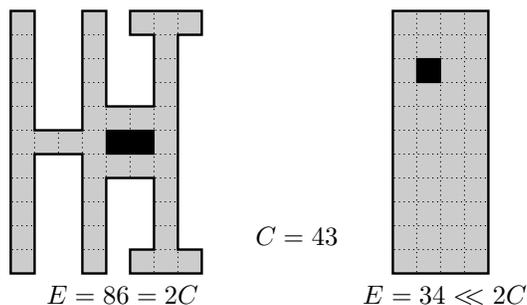


Figure 2: The perimeter, E , is used to distinguish between *thin* and *thick* environments.

We analyze the performance of an exploration strategy using some parameters of the grid polygon. In addition to the area, C , of a polygon we use the *perimeter*, E . The parameter C is the number of free cells and E is the total number of edges that appear between a free cell and a blocked cell; see, for example, Figure 1 or Figure 2. We use the perimeter, E , to distinguish between thin environments that have many corridors of width 1, and thick environments that have wider areas. In the following sections we present strategies that explore grid polygons using no more than roughly $C + \frac{1}{4}E$ steps (hexagons) and $C + E$ (triangles). Since all cells in the environment have to be visited, C is a lower bound on the number of steps that are needed to explore the whole polygon and to return to s . Thus, the number of edges (or a fraction of them) is an upper bound for the number of additional cell visits. For thick environments, the value of E is in $O(\sqrt{C})$, so that the number of additional cell visits is substantially smaller than the

²In the following, we sometimes use the terms *free cells* and *cells* synonymously.

number of free cells. Only for polygons that do not contain any 2×2 square of free cells, E achieves its maximum value of $2(C + 1)$. But in those cases, no online strategy can do better; even the optimal path has this length.

We will see that our strategy SmartDFS explores the polygon in layers: Beginning with the cells along the boundary, the agent proceeds towards the interior of P . Thus, we number the single layers:

Definition 2 Let P be a (simple) grid polygon (of either type). 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 ℓ th layer and the ℓ -offset of P are defined successively; see Figure 3.

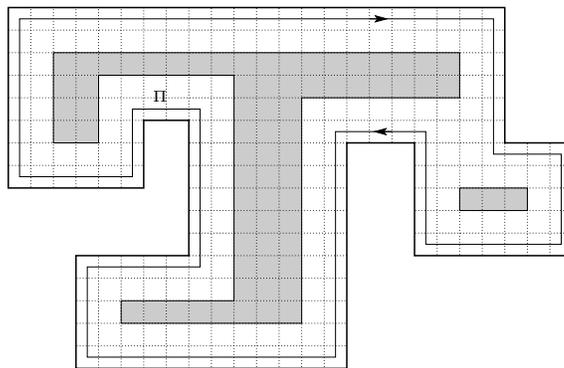


Figure 3: The 2-offset (shaded) of a grid polygon P .

Note that the ℓ -offset of a polygon P is not necessarily connected.

3 Lower Bounds

In an online setting, the agent does not know the environment in advance. So we are interested in the best competitive factor we can expect for a strategy that visits every cell at least once and returns to the start cell. In an environment *with holes*, we have the following theorem:

Theorem 3 *The competitive complexity of exploring an unknown grid polygon with obstacles and hexagonal or triangular cells is 2.*

Proof. We can simply adapt the lower bound construction for square cells by Icking et al. [24], yielding a lower bound of 2. On the other hand, we can apply a simple depth-first search, resulting in a tour with $2C - 2$ steps. The shortest tour needs at least C steps to visit all cells and to return to s , so DFS is competitive with a factor of 2. \square

Surprisingly, we cannot trim the lower bound construction by Icking et al. [24] for simple polygons with hexagonal or triangular cells. The lower bound construction for polygons with holes uses only corridors of width 1, so the type of cells does not matter. In contrast, the construction for simple polygons uses wider areas, where the number of neighboring cells plays a major role. However, the lower bounds for squares and triangles are identical:

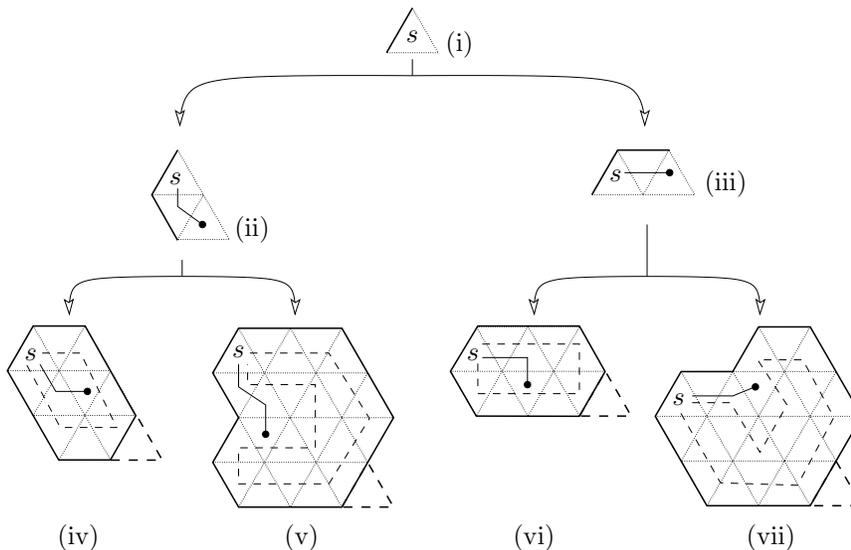


Figure 4: A lower bound on the exploration of simple triangular polygons. The thin dashed lines show the optimal solution, the bold dashed triangles denotes the start cell of the next block.

Theorem 4 *There is no online strategy for the exploration of simple triangular grid polygons with a competitive factor better than $\frac{7}{6}$.*

Proof. Let the agent start in a cell with two neighbors, one to the south and one to the northwest, see Figure 4(i). If it walks to the south, we add a cell such that the only possible step is to the southwest; see Figure 4(ii). If it walks from the start to the east, we force it to move another step to the east; see Figure 4(iii). In both cases, the agent has the choice to leave the polygon’s boundary (Figure 4(iv) and (vi)) or to follow the polygon’s boundary (Figure 4(v) and (vii)). In either case, we fix the polygon after this step. If the agent leaves the boundary, it needs at least 12 steps while the optimal path has 10 steps. In the other case, the agent needs at least 26 steps; the optimal path in this polygon has a length of 22 steps.

To construct arbitrarily large polygons, we use more of these blocks and glue them together using the cell that is shown with bold dashed lines in the figure and denotes the start cell of the next block. Unfortunately, both

the online strategy and the optimal path need two additional steps for the transition between two blocks. Let n denote the number of blocks, then we have in the best case a ratio of $\frac{26+28(n-1)}{22+24(n-1)}$, which converges to $\frac{7}{6}$ if n goes to infinity. \square

The lower bound construction for hexagonal polygons is simpler, but yields a smaller value.

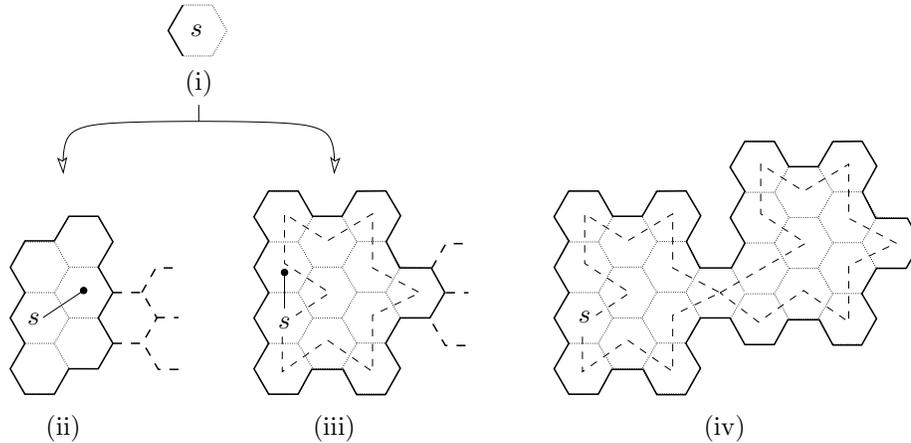


Figure 5: A lower bound on the exploration of simple polygons. The dashed lines show the optimal solution.

Theorem 5 *There is no online strategy for the exploration of simple hexagonal grid polygons with a competitive factor better than $\frac{14}{13}$.*

Proof. We start in a cell with four neighboring cells, see Figure 5(i). The agent may leave the polygon's boundary by walking northwest or southwest, or follow the boundary by walking north or south. In the first case, we close the block as shown in Figure 5(ii), in the second case as shown in Figure 5(iii), yielding a ratio of $\frac{7}{6}$ or $\frac{13}{12}$, respectively.

As in the preceding proof, we construct polygons of arbitrary size by concatenating the blocks from Figure 5(ii) and Figure 5(iii). A subsequent block attaches using the cell(s) shown with bold, dashed lines. Again, we need one or two additional steps for the transition, yielding a best-case ratio of $\frac{13+14(n-1)}{12+13(n-1)}$, where n denotes the number of blocks. This ratio converges to $\frac{14}{13} \approx 1.076$. \square

4 Exploring Simple Polygons

In this section, we briefly describe the strategy $SmartDFS_{\Delta, \circ}$. Our strategy is based on the same ideas as $SmartDFS_{\square}$ [24], but it is generalized for triangular and hexagonal grids.

The basic idea is to use a simple DFS strategy as shown in Algorithm 4.1.³ From the current position, the explorer tries to visit the adjacent cells in clockwise order, see the procedure *ExploreCell*. If the adjacent cell is still unexplored, the agent enters this cell, proceeds recursively with the exploration, and walks back, see the procedure *ExploreStep*. Altogether, the polygon is explored following the *left-hand rule*: The agent always keeps the polygon’s boundary or the explored cells on its left side.

Algorithm 4.1 DFS

DFS(P , *start*):

Choose direction *dir* such that the cell behind
the explorer is blocked;
ExploreCell(*dir*);

ExploreCell(*dir*):

// Left-Hand Rule:
for all cells c adjacent to the current cell, in clockwise order starting
with the cell opposite to *dir* **do**
 newdir := Direction towards c ;
 ExploreStep(*newdir*);
end for

ExploreStep(*dir*):

if unexplored(*dir*) **then**
 move(*dir*);
 ExploreCell(*dir*);
 move(reverse(*dir*));
end if

Obviously, all cells are visited, because the polygon is connected; and the whole path consists of $2C - 2$ steps, because each cell—except for the start—is entered exactly once by the first *move* statement, and left exactly once by the second *move* statement in the procedure *ExploreStep*.

The first improvement to the simple DFS is to return directly to those cells that have unexplored neighbors. See, for example, Figure 6: After the agent has reached the cell c_1 , DFS walks to c_2 through the completely explored corridor of width 2. A more efficient return path walks on a shortest path from c_1 to c_2 . Note that the agent can use for this shortest path only cells that are already known. With this modification, the agent’s position

³The command *move*(*dir*) executes the actual motion of the agent. The function *unexplored*(*dir*) returns true, if the cell in the given direction seen from the agent’s current position is not yet visited, and false otherwise. Given a direction *dir*, *reverse*(*dir*) returns the direction turned by 180°.

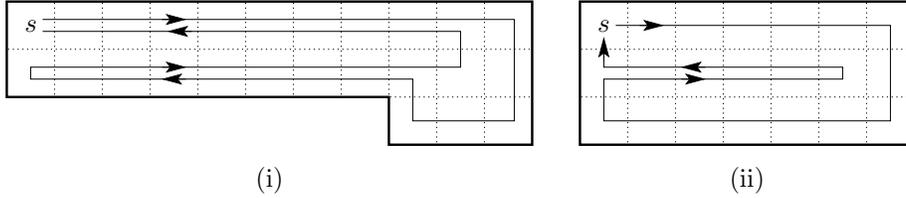


Figure 8: Straightforward strategies are not better than SmartDFS.

lower bound on the performance of this strategy is a corridor of width 3, see Figure 8(ii). Moreover, it is not known whether the offline solution is NP-hard for simple polygons.

4.1 The Analysis of SmartDFS_\circ

In this section, we analyze the performance of our strategy in a hexagonal grid. We start with an important property of the ℓ -offset:

Lemma 6 *The ℓ -offset of a simple, hexagonal grid polygon, P_\circ , has at least 12ℓ edges fewer than P_\circ .⁵*

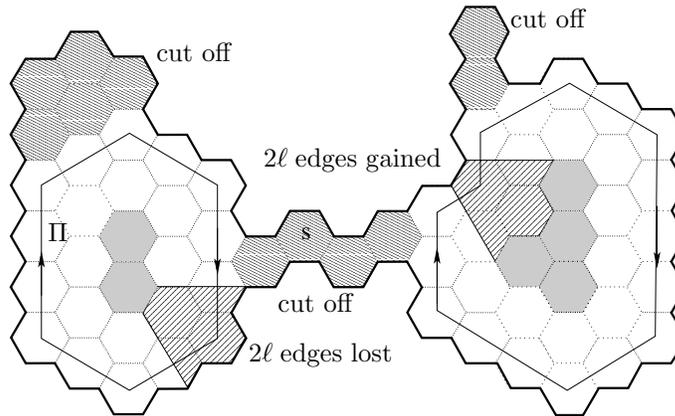


Figure 9: The 2ℓ -offset (shaded) of a grid polygon P_\circ .

Proof. First, we cut the parts of the polygon P_\circ that do not affect the ℓ -offset. Now, we observe the closed path, Π , that connects the midpoints of the boundary cells of the remaining polygon in clockwise order; see Figure 9. For every 60° left turn the offset gains at most 2ℓ edges and for every 60° right turn the offset loses at least 2ℓ edges (cutting off the passages that are narrower than 2ℓ ensures that we have enough edges to lose at this point).

⁵Provided that the ℓ -offset is not empty anyway.

Algorithm 4.2 SmartDFS

SmartDFS(P , $start$):

Choose direction dir such that the cell behind
the explorer is blocked;
ExploreCell(dir);
Walk on the shortest path to the start cell;

ExploreCell(dir):

Mark the current cell with the number of the current layer;
 $base :=$ current position;
if not isSplitCell($base$) **then**
 // Left-Hand Rule:
 for all cells c adjacent to the current cell, in clockwise order
 starting with the cell opposite to dir **do**
 $newdir :=$ Direction towards c ;
 ExploreStep($base$, $newdir$);
 end for
else
 // choose different order, see page 12 ff
 Determine the types of the components using the layer numbers
 of the surrounding cells;
 if No component of type III exists **then**
 Use the left-hand rule, but omit the first possible step.
 else
 Visit the component of type III at last.
 end if
end if

ExploreStep($base$, dir):

if unexplored($base$, dir) **then**
 Walk on shortest path using known cells to $base$;
 move(dir);
 ExploreCell(dir);
end if

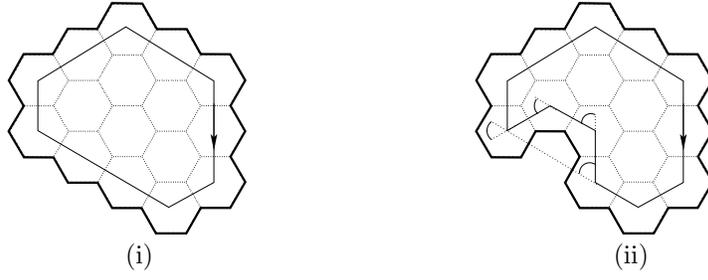


Figure 10: (i) A convex hexagonal grid polygon with six 60° right turns in the first layer, (ii) adding two 60° left turns forces adding to 60° right turns.

It is easy to see that there are six more 60° right turns than left turns (we count 120° turn as two 60° turns): In a convex polygon, the tour along the first layer has six 60° right turns. For every 60° left turn that we add to the polygon, we also add another 60° right turns, see Figure 10. Altogether, we loose at least 12 edges for every layer. \square

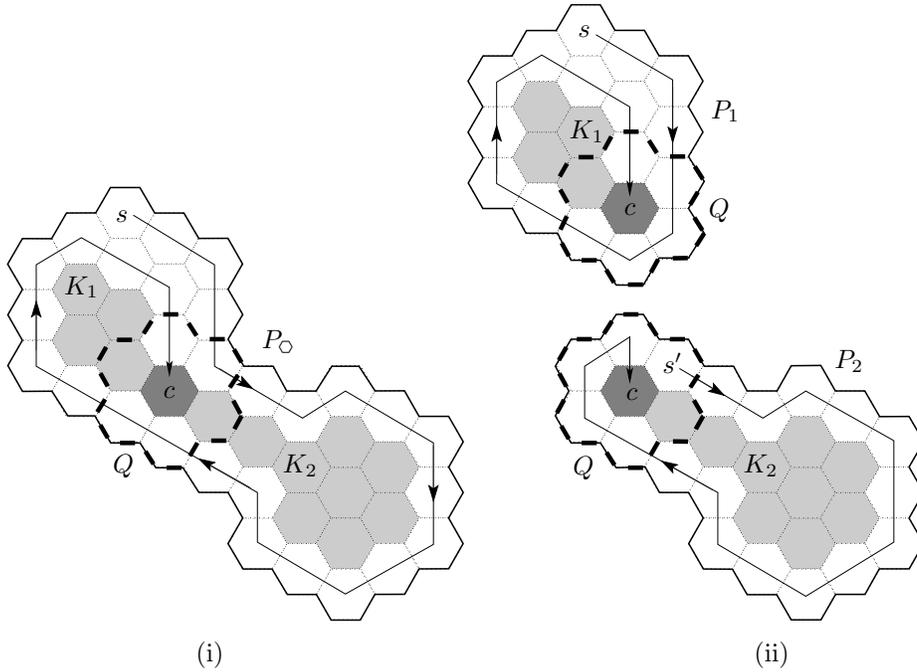


Figure 11: A decomposition of P_O at the split cell c and its handling in $SmartDFS_O$.

Now, let us consider the handling of a split cell. Observe a situation as shown in Figure 11(i): $SmartDFS_O$ has just met the first split cell, c , in the

second layer of P_\circ . P_\circ divides into three parts:

$$P_\circ = K_1 \dot{\cup} K_2 \dot{\cup} \{ \text{visited cells of } P_\circ \},$$

where K_1 and K_2 denote the connected components of the set of unvisited cells. In this case it is reasonable to explore the component K_2 first, because the start cell s is closer to K_1 ; that is, we can extend K_1 with ℓ layers, such that the resulting polygon contains the start cell s .

More generally, we want to divide our polygon P_\circ into two parts, P_1 and P_2 , such that each of them is an extension of the two components. Both polygons overlap in the area around the split cell c . At least one of these polygons contains the start cell. If only one of the polygons contains s , we want our strategy to explore this part at last, expecting that in this part the path from the last visited cell back to s is the shorter than in the other part. Vice versa, if there is a polygon that does *not* contain s , we explore the corresponding component first. In Figure 11, SmartDFS $_\circ$ recursively enters K_2 , returns to the split cell c , and explores the component K_1 next.

In the preceding example, there is only one split cell in P_\circ , but in general there will may be a sequence of split cells, c_1, \dots, c_k . In this case, we apply the handling of split cells in a recursive way; that is, if a split cell c_{i+1} , $1 \leq i < k$, is detected in one of the two components occurring at c_i we proceed the same way as described earlier. If another split cell occurs in K_2 , the role of the start cell is played by the preceding split cell c_i . In the following, the term *start cell* always refers to the start cell of the current component; that is, either to s or to the previously detected split cell. Note that—in contrast to square grid polygons—there is no case where three components arise at a split cell apart from the start cell s .

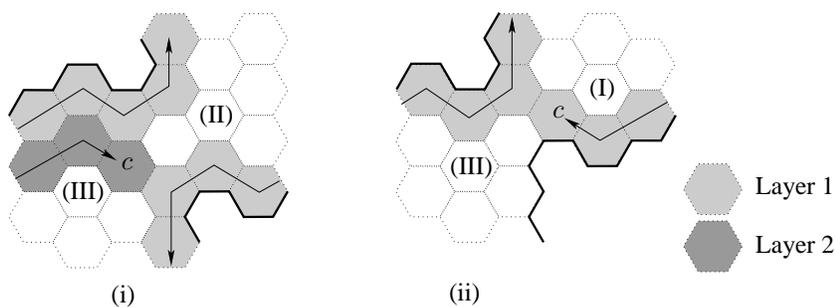


Figure 12: Several types of components.

Visiting Order

We use the layer numbers to decide which component we have to visit at last. Whenever a split cell occurs in layer ℓ , every component is one of the following types, see Figure 12:

- I. K_i is *completely* surrounded by layer ℓ ⁶
- II. K_i is *not* surrounded by layer ℓ
- III. K_i is *partially* surrounded by layer ℓ

It is reasonable to explore the component of type III at last: There are two cases in which SmartDFS_\circ switches from a layer $\ell - 1$ to layer ℓ . Either it reaches the first cell of layer $\ell - 1$ in the current component and thus passes the start cell of the current component, or it hits another cell of layer $\ell - 1$ but no polygon split occurs. In the second case, the considered start cell must be located in a corridor that is completely explored; otherwise, the strategy would be able to reach the first cell of layer $\ell - 1$ as in the first case. In both cases the part of P_\circ surrounding a component of type III contains the first cell of the current layer ℓ as well as the start cell. Thus, we explore a component of type III at first, provided that such a component exists.

Unfortunately, there are two cases in which no component of type III exists:

1. The part of the polygon that contains the preceding start cell is explored completely, see for example Figure 13(i). In this case the order of the components makes no difference.
2. Both components are completely surrounded by a layer, because the polygon split and the switch from one layer to the next occurs within the same cell, see Figure 13(ii). A step that follows the left-hand rule will move towards the start cell, so we just omit the first possible.

We proceed with the rule in case 2 whenever there is no component of type III, because the order in case 1 does not make a difference.

⁶More precisely, the part of layer ℓ that surrounds K_i is completely visited. For convenience, we use the slightly sloppy, but shorter form.

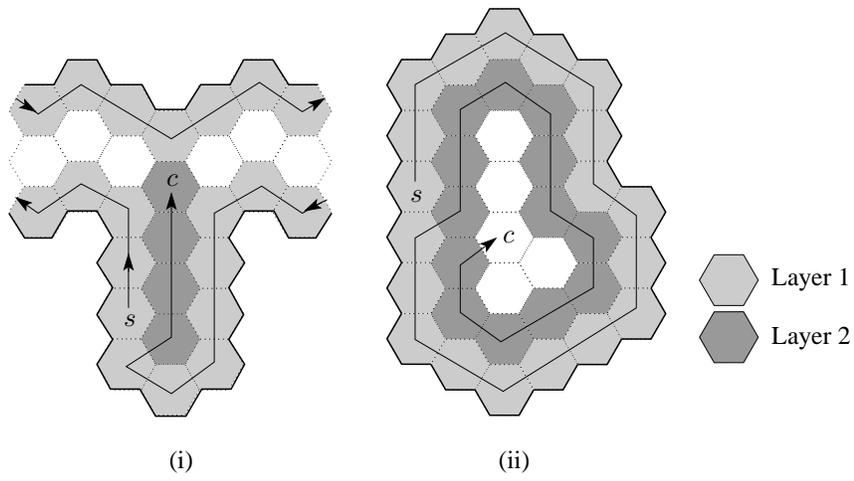


Figure 13: No component of type III exists.

4.1.1 An Upper Bound on the Number of Steps

For the analysis of our strategy we consider two polygons, P_1 and P_2 , as follows. Let Q be the polygon that is made of c and extended by q layers, where

$$q := \begin{cases} \ell, & \text{if } K_2 \text{ is of type I} \\ \ell - 1, & \text{if } K_2 \text{ is of type II} \end{cases} .$$

K_2 denotes the component that is explored first, and ℓ denotes the layer in which the split cell was found. We choose $P_2 \subset P_{\circ} \cup Q$ such that $K_2 \cup \{c\}$ is the q -offset of P_2 , and $P_1 := ((P_{\circ} \setminus P_2) \cup Q) \cap P_{\circ}$, see Figure 11. The intersection with P_{\circ} is necessary, because Q may exceed the boundary of P_{\circ} , see Figure 14. Note that at least P_1 contains the preceding start cell. There is an arbitrary number of polygons P_2 , such that $K_2 \cup \{c\}$ is the q -offset of P_2 , because 'dead ends' of P_2 that are not wider than $2q$ do not affect the q -offset. To ensure a unique choice of P_1 and P_2 , we require that both P_1 and P_2 are connected, and both $P_{\circ} \cup Q = P_1 \cup P_2$ and $P_1 \cap P_2 \subseteq Q$ are satisfied.

The choice of P_1, P_2 and Q ensures that the agent's path in $P_1 \setminus Q$ and in $P_2 \setminus Q$ do not change compared to the path in P_{\circ} . The parts of the agent's path that lead from P_1 to P_2 and from P_2 to P_1 are fully contained in Q . Just the parts inside Q are bended to connect the appropriate paths inside P_1 and P_2 ; see Figure 11 and Figure 14.

In Figure 11, K_1 is of type III and K_2 is of type II. A component of type I occurs, if we detect a split cell as shown in Figure 14. Note that Q may exceed P_{\circ} , but P_1 and P_2 are still well-defined.

We want to visit every cell in the polygon and to return to s . Every strategy needs at least $C(P_{\circ})$ steps to fulfill this task, where $C(P_{\circ})$ denotes the number of cells in P_{\circ} . Thus, we can split the overall length of the exploration path, Π , into two parts, $C(P_{\circ})$ and $\text{excess}(P_{\circ})$, with $|\Pi| = C(P_{\circ}) + \text{excess}(P_{\circ})$. In this context, $C(P_{\circ})$ is a lower bound on the number of steps that are needed for the exploration task, whereas $\text{excess}(P_{\circ})$ is the number of additional cell visits.

Because SmartDFS_{\circ} recursively explores $K_2 \cup \{c\}$, we want to apply the upper bound inductively to the component $K_2 \cup \{c\}$. If we explore P_1 with SmartDFS_{\circ} until c is met, the set of unvisited cells in P_1 is equal to K_1 , because the path outside Q do not change. Thus, we can apply our bound inductively to P_1 , too. The following lemma gives us the relation between the path lengths in P_{\circ} and the path lengths in the two components.

Lemma 7 *Let P_{\circ} be a simple hexagonal grid polygon. Let the explorer visit the first split cell, c , which splits the unvisited cells of P_{\circ} into two components K_1 and K_2 , where K_2 is of type I or II. With the preceding notations we have*

$$\text{excess}(P_{\circ}) \leq \text{excess}(P_1) + \text{excess}(K_2 \cup \{c\}) + 1 .$$

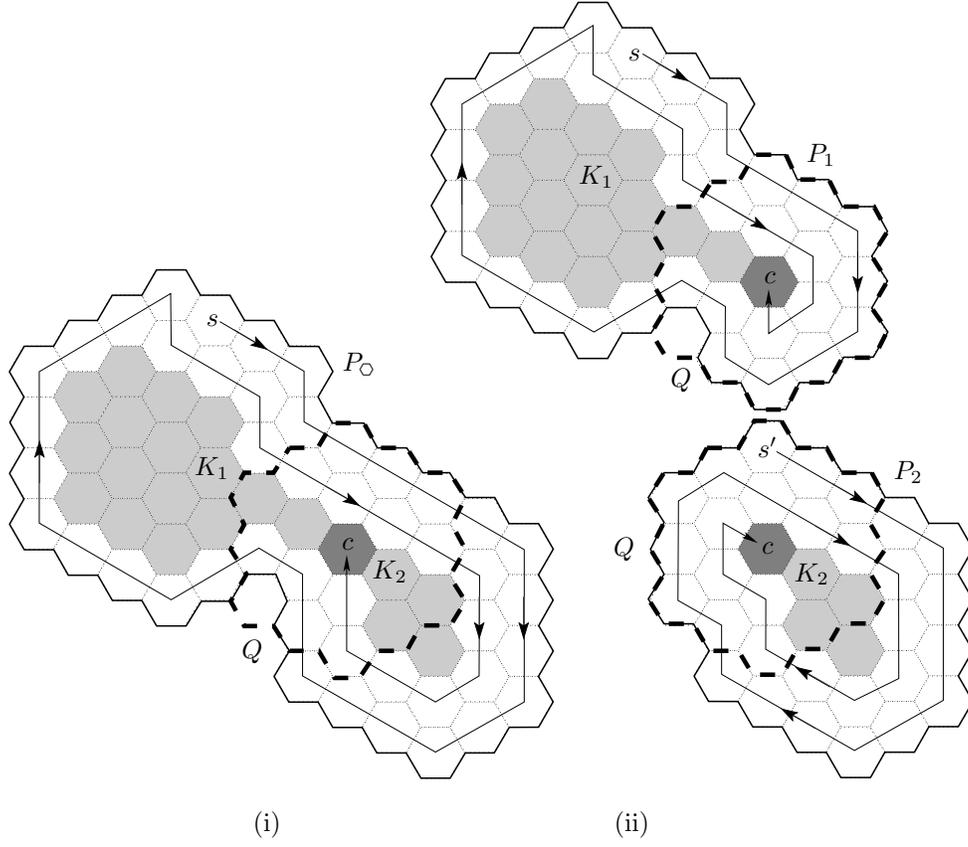


Figure 14: The component K_2 is of type I. Q may exceed P_O .

Proof. The strategy SmartDFS_O has reached the split cell c and explores $K_2 \cup \{c\}$ with start cell c first. Because c is the first split cell, there is no excess in $P_2 \setminus (K_2 \cup \{c\})$ and it suffices to consider $\text{excess}(K_2 \cup \{c\})$ for this part of the polygon. After $K_2 \cup \{c\}$ is finished, the agent returns to c and explores K_1 . For this part we take $\text{excess}(P_1)$ into account. Finally, we add one single step, because the split cell c is visited twice: once, when SmartDFS_O detects the split and once more after the exploration of $K_2 \cup \{c\}$ is finished. Altogether, the given bound is achieved. \square

c is the first split cell in P_O , so $K_2 \cup \{c\}$ is the q -offset of P_2 and we can apply Lemma 6 to bound the number of boundary edges of $K_2 \cup \{c\}$ by the number of boundary edges of P_2 . The following lemma allows us to charge the number of edges in P_1 and P_2 against the number of edges in P_O and Q .

Lemma 8 *Let P_O be a simple hexagonal grid polygon, and let P_1, P_2 and Q be defined as earlier. The number of edges satisfy the equation*

$$E(P_1) + E(P_2) = E(P_O) + E(Q) .$$

Proof. Obviously, two arbitrary polygons P_1 and P_2 always satisfy⁷

$$E(P_1) + E(P_2) = E(P_1 \cup P_2) + E(P_1 \cap P_2) .$$

With $P_1 \cap P_2 = P_\circ \cap Q$ and $P_1 \cup P_2 = P_\circ \cup Q$ we have

$$\begin{aligned} E(P_1) + E(P_2) &= E(P_1 \cap P_2) + E(P_1 \cup P_2) \\ &= E(P_\circ \cap Q) + E(P_\circ \cup Q) \\ &= E(P_\circ) + E(Q) \quad \square \end{aligned}$$

Finally, we need an upper bound for the length of a path inside a grid polygon.

Lemma 9 *Let Π be the shortest path between two cells in a hexagonal grid polygon P_\circ . The length of Π is bounded by*

$$|\Pi| \leq \frac{1}{4}E(P_\circ) - \frac{3}{2} .$$

Proof. W.l.o.g. we can assume that the start cell, s , and the target cell, t , of Π belong to the first layer of P , because we are searching for an upper bound for the shortest path between two arbitrary cells.

Let Π_1 be the closed path in the 1-offset of P_\circ . For every forward step we have 2 edges on the boundary of P_\circ , for every 60° right turn 3 edges, for every 120° right turn 4 edges, and for every left turn 1 edge. For every left turn we can charge one right turn, so we have in average 2 edges for every step. Further, we have 6 more right turns (120° turns count twice) than left turns; that is, $|\Pi_1| \leq \frac{E(P_\circ)-6}{2}$. Now, observe the path Π_L from s to t in the first layer that follows the boundary of P clockwise and the path Π_R that follows the boundary counterclockwise. In the worst case, both Π_L and Π_R have the same length, so $|\Pi| = |\Pi_L| = |\Pi_R|$ holds. Thus, we have

$$2 \cdot |\Pi| \leq |\Pi_1| \leq \frac{E(P_\circ) - 6}{2} \implies |\Pi| \leq \frac{1}{4}E(P_\circ) - \frac{3}{2} .$$

□

Now, we are able to show our main theorem:

Theorem 10 *Let P_\circ be a simple grid polygon with $C(P_\circ)$ cells and $E(P_\circ)$ edges. P_\circ can be explored with*

$$S(P_\circ) \leq C(P_\circ) + \frac{1}{4}E(P_\circ) - \frac{5}{2}$$

steps. This bound is tight.

⁷Note that $P_1 \cap P_2$ may have thin parts if common edges of P_1 and P_2 do not occur in $P_1 \cup P_2$. We count these these edges twice in $P_1 \cap P_2$.

Proof. We show by an induction on the number of components that $\text{excess}(P_\circ) \leq \frac{1}{4}E(P_\circ) - \frac{5}{2}$ holds. For the induction base we consider a polygon without any split cell: SmartDFS_\circ visits each cell and returns on the shortest path to the start cell. Because there is no polygon split, all cells of P_\circ can be visited by a path of length $C(P_\circ) - 1$. By Lemma 9 the shortest path back to the start cell is not longer than $\frac{1}{4}E(P_\circ) - \frac{3}{2}$; thus, $\text{excess}(P_\circ) \leq \frac{1}{4}E(P_\circ) - \frac{5}{2}$ holds.

Now, we assume that there is more than one component during the application of SmartDFS_\circ . Let c be the first split cell detected in P_\circ . When SmartDFS_\circ reaches c , two new components, K_1 and K_2 , occur. We consider the two polygons P_1 and P_2 defined as earlier, using the polygon Q around c ; K_2 is recursively explored first. As shown in Lemma 7 we have

$$\text{excess}(P_\circ) \leq \text{excess}(P_1) + \text{excess}(K_2 \cup \{c\}) + 1 .$$

Now, we apply the induction hypothesis to P_1 and $K_2 \cup \{c\}$ and get

$$\text{excess}(P_\circ) \leq \frac{1}{4}E(P_1) - \frac{5}{2} + \frac{1}{4}E(K_2 \cup \{c\}) - \frac{5}{2} + 1 .$$

Applying Lemma 6 to the q -offset $K_2 \cup \{c\}$ of P_2 yields $E(K_2 \cup \{c\}) \leq E(P_2) - 12q$. Thus, we achieve

$$\text{excess}(P_\circ) \leq \frac{1}{4}E(P_1) + \frac{1}{4}E(P_2) - 3q - 4 .$$

With Lemma 8 we have $E(P_1) + E(P_2) = E(P_\circ) + E(Q)$; and from Lemma 6 we conclude $E(Q) = 12q + 6$ (a hexagon has 6 edges and Q gains 12 edges per layer). Altogether, we have

$$\begin{aligned} \text{excess}(P_\circ) &\leq \frac{1}{4} \left(E(P_1) + E(P_2) \right) - 3q - 4 \\ &\leq \frac{1}{4} \left(E(P_\circ) + 12q + 6 \right) - 3q - 4 \\ &= \frac{1}{4}E(P_\circ) - \frac{5}{2} . \end{aligned}$$

It is easy to see that this bound is exactly achieved in corridors of width 1. The exploration of such a corridor needs $2(C(P_\circ) - 1)$ steps. On the other hand, the number of edges is $E(P_\circ) = 4C(P_\circ) + 2$ (a corridor with one cell has 6 edges, and for every additional cell we get 4 additional edges). \square

4.1.2 Competitive Factor

So far we have shown an upper bound on 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_\circ 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*. More precisely, a cell, c , belongs to a narrow passage, if c can be removed without changing the layer number of any other cell.

It is easy to see that narrow passages are explored optimally: In corridors of width 1 both SmartDFS_\circ 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_\circ . The first one gives us a relation between the number of cells and the number of edges for a special class of polygons.

Lemma 11 *For a simple grid polygon, P_\circ , with $C(P_\circ)$ cells and $E(P_\circ)$ edges, and without any narrow passage or split cells in the first layer, we have*

$$E(P_\circ) \leq \frac{4}{3}C(P_\circ) + \frac{26}{3}.$$

Proof. Consider a simple polygon, P_\circ . We successively remove at least three connected boundary cells that either form a straight line or two lines with a 60° angle. We remove a set of cells only if the resulting polygon still fulfills our assumption that the polygon has no narrow passages or split cells in the first layer. These assumptions ensure that we can always find such a row or column (i.e., if we cannot find such a row or column, the polygon has a narrow passage or a split cell in the first layer). Thus, we remove at least three cells and at most *four* edges. This decomposition ends with a 'honeycomb', a center cell with its 6 neighbors, with 7 cells and 18 edges that fulfills $E(P_\circ) = \frac{4}{3}C(P_\circ) + \frac{26}{3}$; see Figure 15(i) Now, we reverse our decomposition; that is, 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 four edges. Thus, $E(P_\circ) \leq \frac{4}{3}C(P_\circ) + \frac{26}{3}$ is fulfilled in every step. \square

For the same class of polygons, we can show that SmartDFS_\circ behaves slightly better than the bound in Theorem 10.

Lemma 12 *A simple hexagonal grid polygon, P_\circ , with $C(P_\circ)$ cells and $E(P_\circ)$ edges, and without any narrow passage or split cells in the first layer can be explored using no more steps than*

$$S(P_\circ) \leq C(P_\circ) + \frac{1}{4}E(P_\circ) - \frac{9}{2}.$$

Proof. In Theorem 10 we have seen that $S(P_\circ) \leq C(P_\circ) + \frac{1}{4}E(P_\circ) - \frac{5}{2}$ holds. To show this theorem, we used Lemma 9 on page 17 as an upper bound for the shortest path back from the last explored cell to the start cell. Lemma 9 bounds the shortest path from a cell, c , in the first layer of P_\circ to

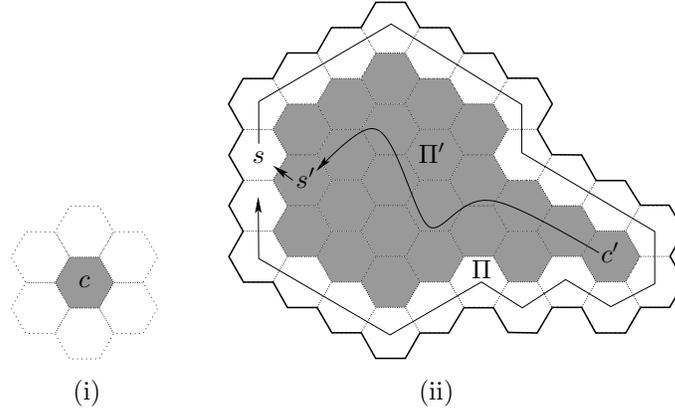


Figure 15: (i) The minimal polygon that has neither narrow passages nor split cells in the first layer, (ii) for polygons without narrow passages or split cells in the first layer, the last explored cell, c' , lies in the 1-offset, P' (shaded).

the cell c' that maximizes the distance to c inside P_\circ ; thus, c' is located in the first layer of P_\circ , too.

Because P_\circ has neither narrow passages nor split cells in the first layer, we can explore the first layer of P_\circ completely before we visit another layer, see Figure 15(ii). Therefore, the last explored cell, c' , of P_\circ is located in the 1-offset of P_\circ . Let P' denote the 1-offset of P_\circ , and s' the first visited cell in P' . Remark that s and s' are neighbors, so the shortest path from s' to s is only one step. Now, the shortest path, Π , from c' to s in P_\circ is bounded by a shortest path, Π' , from c' to s' in P' and a shortest path from s' to s :

$$|\Pi| \leq |\Pi'| + 1.$$

The path Π' , in turn, is bounded using Lemma 9 by

$$|\Pi'| \leq \frac{1}{4}E(P') - \frac{3}{2}.$$

By Lemma 6 (page 9), $E(P') \leq E(P_\circ) - 12$ holds, and altogether we get

$$|\Pi| \leq \frac{1}{4}E(P_\circ) - \frac{7}{2},$$

which is two steps shorter than stated in Lemma 9. □

Now, we can prove the following

Theorem 13 *The strategy SmartDFS_\circ is $\frac{4}{3}$ -competitive.*

Proof. Let P_\circ be a simple grid polygon. In the first stage, we remove all narrow passages from P_\circ and get a sequence of (sub-)polygons P_i , $i = 1, \dots, k$,

without narrow passages. For every P_i , $i = 1, \dots, k - 1$, the optimal strategy in P_\circ explores the part of P_\circ 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_\circ .

Now, we observe a (sub-)polygon P_i . We show by induction on the number of split cells in the first layer that $S(P_i) \leq \frac{4}{3}C(P_i) - \frac{7}{3}$ holds. Note that this bound is exactly achieved in a polygon as shown in Figure 16: For the middle part—a corridor of width 3—, SmartDFS_\circ needs four steps for three cells. Additionally, we have seven cells (the first row and the last two rows) that are explored optimally. Thus, we have $\frac{4}{3}(C(P_i) - 7) + 7 = \frac{4}{3}C(P_i) - \frac{7}{3}$ steps.

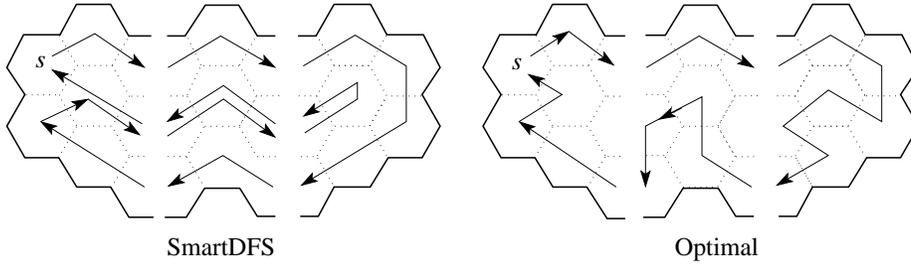


Figure 16: The competitive factor of $\frac{4}{3}$ is exactly achieved: In a corridor of width 3, $S(P_\circ) = \frac{4}{3}S_{\text{Opt}}(P_\circ) - \frac{7}{3}$ holds.

If P_i has no split cell in the first layer (induction base), we can apply Lemma 12 and Lemma 11:

$$\begin{aligned}
 S(P_i) &\leq C(P_i) + \frac{1}{4}E(P_i) - \frac{9}{2} \\
 &\leq C(P_i) + \frac{1}{4}\left(\frac{4}{3}C(P_i) + \frac{26}{3}\right) - \frac{9}{2} \\
 &= \frac{4}{3}C(P_i) - \frac{7}{3}.
 \end{aligned}$$

Two cases occur if we meet a split cell, c , in the first layer, see Figure 17. In the first case, the new component was never visited before (component of type II, see page 13). Here, we define $Q := \{c\}$. The second case occurs, because the explorer meets a cell, c' , that is in the first layer and touches the current cell, c , see for example Figure 17(ii). In this case, we define Q as $\{c, c'\}$.

Similar to the proof of Theorem 10, we split the polygon P_i into two parts, both including Q . Let P'' denote the part that includes the component of type I or II, P' the other part. For $|Q| = 1$, see Figure 17(i), we conclude

4.2 The Analysis of SmartDFS $_{\Delta}$

In this section, we analyze the performance of our strategy on a triangular grid. The proof follows the same outline as the proof in the preceding section. The basic idea is an induction on the split cells. Thus, we point out only the differences between SmartDFS $_{\circ}$ and SmartDFS $_{\Delta}$. The first difference concerns the ℓ -offset:

Lemma 14 *The ℓ -offset of a simple, triangular grid polygon, P_{Δ} , has at least 6ℓ edges fewer than P_{Δ} .*

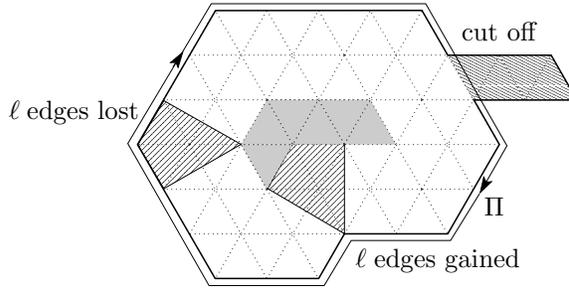


Figure 18: The 2-offset (shaded) of a grid polygon P_{Δ} .

Proof. As in Lemma 6, we can cut off blind alleys that are narrower than 2ℓ , because those parts of P_{Δ} do not affect the ℓ -offset. We walk clockwise around the boundary of the remaining polygon, see Figure 18. For every 60° left turn the offset gains at most ℓ edges and for every 60° right turn the offset loses at least ℓ edges. Similar to the proof of Lemma 6, there are six more 60° right turns than left turns (again, we count 120° turns as two 60° turns). Altogether, we lose at least six edges for every layer. \square

In line with SmartDFS $_{\circ}$, we subdivide a polygon, P_{Δ} , into three parts when we meet a split cell, c , in layer ℓ :

$$P_{\Delta} = K_1 \dot{\cup} K_2 \dot{\cup} \{ \text{visited cells of } P_{\Delta} \},$$

where K_1 and K_2 denote the connected components of the set of unvisited cells, and K_2 is explored first. Again, we have three possible types of components, see Figure 19 and Figure 20:

- I. K_i is *completely* surrounded by layer ℓ
- II. K_i is *not* surrounded by layer ℓ
- III. K_i is *partially* surrounded by layer ℓ

4.2.1 An Upper Bound on the Number of Steps

As in the Section 4.2.1, Q is the split cell broadened by q layers, with

$$q := \begin{cases} \ell, & \text{if } K_2 \text{ is of type I} \\ \ell - 1, & \text{if } K_2 \text{ is of type II} \end{cases}.$$

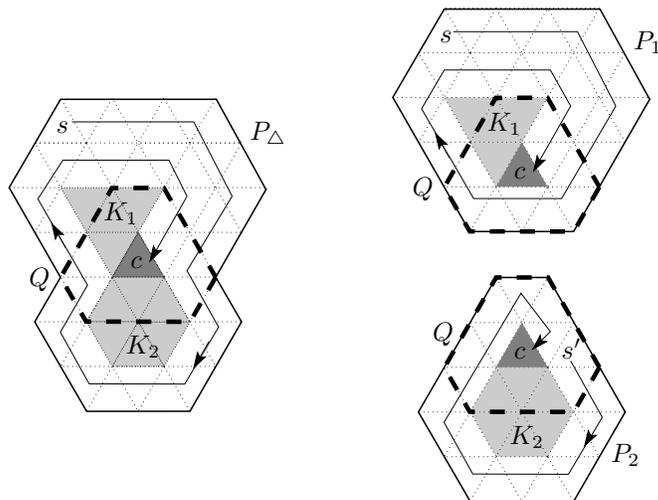


Figure 19: A decomposition of P_Δ : K_1 is of type III, K_2 of type II.

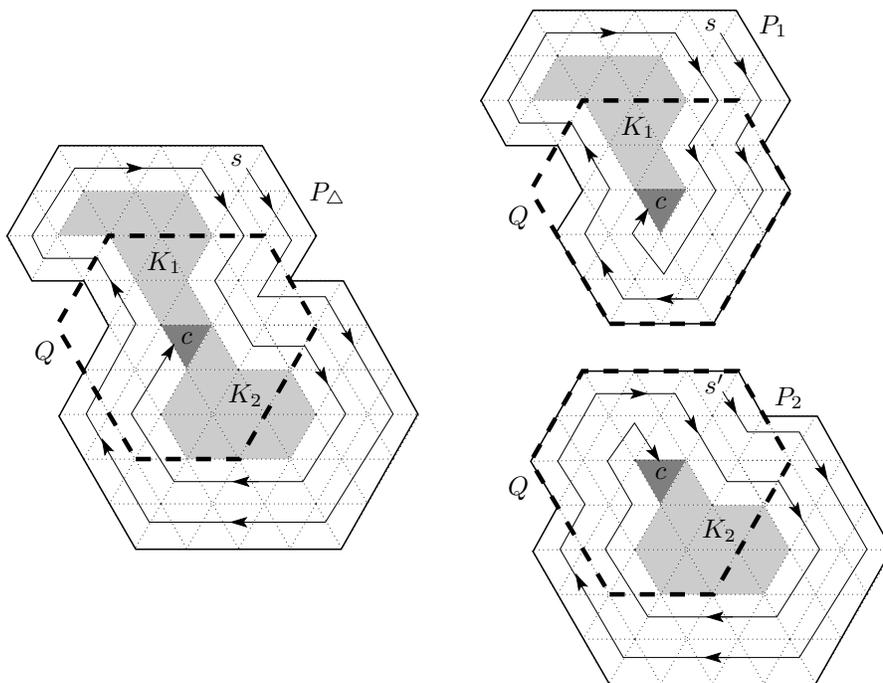


Figure 20: A decomposition of P_Δ : K_1 is of type III, K_2 of type I. (Q may exceed P_Δ .)

Further, we choose $P_2 \subset P_\Delta \cup Q$ such that $K_2 \cup \{c\}$ is the q -offset of P_2 , and $P_1 := ((P_\Delta \setminus P_2) \cup Q) \cap P_\Delta$.

We split the overall length of the exploration path, Π , into two parts, $C(P_\Delta)$ and $\text{excess}(P_\Delta)$, with $|\Pi| = C(P_\Delta) + \text{excess}(P_\Delta)$. Lemma 7 and Lemma 8 hold also for triangular polygons:

Lemma 15 *Let P_Δ be a simple triangular grid polygon. Let the agent visit the first split cell, c , which splits the unvisited cells of P_Δ into two components K_1 and K_2 , where K_2 is of type I or II. With the preceding notations we have*

$$\text{excess}(P_\Delta) \leq \text{excess}(P_1) + \text{excess}(K_2 \cup \{c\}) + 1 .$$

Lemma 16 *Let P_Δ be a simple triangular grid polygon, and let P_1, P_2 and Q be defined as earlier. The number of edges satisfy the equation*

$$E(P_1) + E(P_2) = E(P_\Delta) + E(Q) .$$

In contrast to hexagonal and square polygons, we need all but three edges for an upper bound on the length of a shortest path:

Lemma 17 *Let Π be the shortest path between two cells in a triangular grid polygon P_Δ . The length of Π is bounded by*

$$|\Pi| \leq E(P_\Delta) - 3 .$$

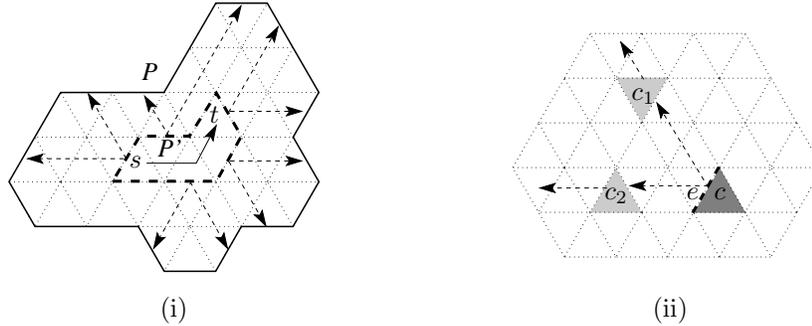


Figure 21: (i) Possible projections of an edge of P' (bold, dashed) onto the edges of P_Δ , (ii) the cell c cannot project its northwestern edge onto the boundary of P_Δ , because c_1 and c_2 are already charging the boundary edges of P_Δ .

Proof. Let P' be the grid polygon that is defined by the cells that Π visits. We prove: $E(P') \leq E(P)$; that is, the number of edges around the path cannot exceed the number of edges in P_Δ . Thus, we have to find for every

edge in P' a corresponding edge in P_Δ : We project every edge, e , of a cell, c , in P' along the axes of the grid onto the boundary of P_Δ —provided that e is not already an edge in P_Δ . For every edge in P' there are two possible axes for the projection, see Figure 21(i).

Now, let us assume, we would doubly charge one edge in P_Δ . Then there is an edge, e , of a cell, c , of Π where both axes are blocked by other cells, c_1 and c_2 on Π , that both project edges onto the boundary of P_Δ , see Figure 21(ii).

Now, we have two cases: Either, c lies in Π between c_1 and c_2 , or outside the path segment between c_1 and c_2 (in this case, let c_2 lie between c_1 and c).

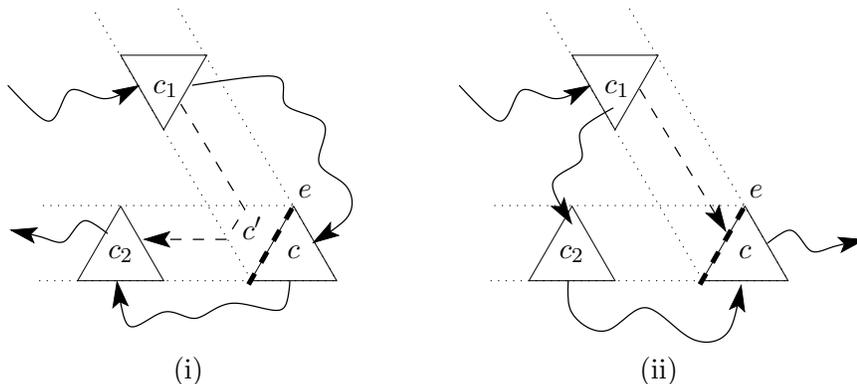


Figure 22: (i) c lies in Π between c_1 and c_2 , (ii) c_2 lies in Π between c and c_1 . In both cases the path can be shortened (dashed).

In the first case, we can shorten the path between c_1 and c_2 by moving straight from c_1 to the cell c' that is adjacent to c and e (c' must be a free cell in P_Δ ; otherwise, e would be an edge in P_Δ); see Figure 22(i). In the other case, we can shorten the path, too: We replace the path between c_1 and c over c_2 by the straight path from c_1 to c ; see Figure 22(ii). (The straight path between c_1 and c must be in P_Δ ; otherwise, there would be a blocked cell and an edge; thus, we would be able to project e onto the boundary of P_Δ). Altogether, we can shorten Π . This is a contradiction to the assumption that Π is a shortest path. Thus, for every edge, e' in P_Δ there is at most one edge in P' that is projected onto e' (note that we project only from the interior of P to the boundary); that is, $E(P') \leq E(P)$ holds.

In a corridor K of width 1 we have $C(K) = E(K) - 2$ (this equation holds for a single, triangular cell, and for every further cell in a corridor of width 1 we add one cell and precisely one edge). P' is such a corridor. Thus, we have

$$|\Pi| = C(P') - 1 = E(P') - 3 \leq E(P) - 3.$$

□

Now, we are able to show our main theorem:

Theorem 18 *Let P_Δ be a simple triangular grid polygon with $C(P_\Delta)$ cells and $E(P_\Delta)$ edges. P_Δ can be explored with*

$$S(P_\Delta) \leq C(P_\Delta) + E(P_\Delta) - 4$$

steps. This bound is tight.

Proof. The outline of this proof is the same as in Theorem 10. We show by an induction on the number of components that $\text{excess}(P_\Delta) \leq E(P_\Delta) - 4$ holds.

For the induction base we consider a polygon without any split cell which can be explored in $C(P_\Delta) - 1$ steps. By Lemma 17 the shortest path back to the start cell is bounded by $E(P_\Delta) - 3$; thus, $\text{excess}(P_\Delta) \leq E(P_\Delta) - 4$ holds.

Now, let c be the first split cell detected in P_Δ . When reaching c , we have the components K_1 and K_2 ; we explore K_2 first. P_1 , P_2 , and Q are defined as earlier. As shown in Lemma 15 we have

$$\text{excess}(P_\Delta) \leq \text{excess}(P_1) + \text{excess}(K_2 \cup \{c\}) + 1.$$

Now, we apply the induction hypothesis to P_1 and $K_2 \cup \{c\}$ and get

$$\text{excess}(P_\Delta) \leq E(P_1) - 4 + E(K_2 \cup \{c\}) - 4 + 1.$$

Applying Lemma 14 to the q -offset $K_2 \cup \{c\}$ of P_2 yields

$$\text{excess}(P_\Delta) \leq E(P_1) + E(P_2) - 6q - 7$$

From Lemma 14 we conclude $E(Q) = 6q + 3$ (A triangle has 3 edges and Q gains 6 edges per layer). With Lemma 16 we have $\text{excess}(P_\Delta) \leq E(P) - 4$. This bound is exactly achieved in a corridor of width 1. \square

4.2.2 Competitive Factor

Again, we also want to analyze our strategy in the competitive framework. As in the preceding section, *narrow passages* (i.e., corridors of width 1 or 2) are explored optimally, so we consider only polygons without such narrow passages or split cells in the first layer. Analogously to Lemma 11 (see Figure 23(i)) and Lemma 12 we can show:

Lemma 19 *For a simple grid polygon, P_Δ , with $C(P_\Delta)$ cells and $E(P_\Delta)$ edges, and without any narrow passage or split cells in the first layer, we have*

$$E(P_\Delta) \leq \frac{1}{3} C(P_\Delta) + \frac{14}{3}.$$

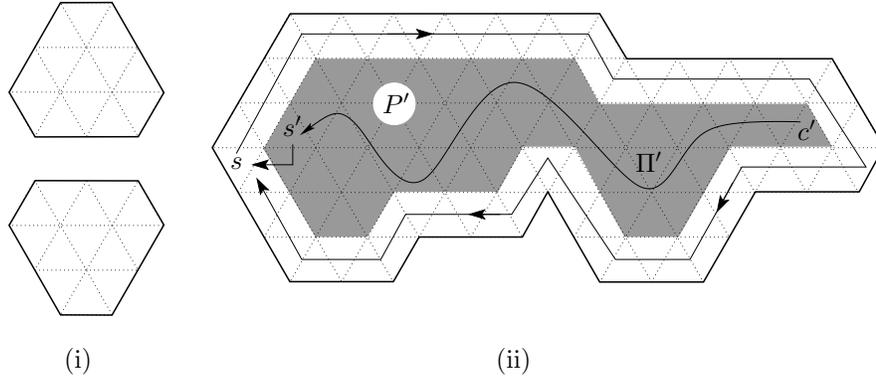


Figure 23: (i) Minimal polygons that have neither narrow passages nor split cells in the first layer, (ii) for polygons without narrow passages or split cells in the first layer, the last explored cell, c' , lies in the 1-offset, P' (shaded).

Lemma 20 *A simple triangular grid polygon, P_Δ , with $C(P_\Delta)$ cells and $E(P_\Delta)$ edges, and without any narrow passage or split cells in the first layer can be explored using no more steps than*

$$S(P_\Delta) \leq C(P_\Delta) + E(P_\Delta) - 6.$$

Proof. The proof is analogously to Lemma 12, but we have to count three step for the path from s' to s ; see Figure 23(ii). Thus, we have $|\Pi| \leq |\Pi'| + 3$. With $|\Pi'| \leq E(P') - 3$ (by Lemma 17) and $E(P') \leq E(P_\Delta) - 6$ (by Lemma 6), we get $|\Pi| \leq E(P_\Delta) - 6$, which is two steps shorter than the one used in the proof of Theorem 18. \square

Now, we can prove the following

Theorem 21 *The strategy SmartDFS_Δ is $\frac{4}{3}$ -competitive.*

Proof. In line with Theorem 13, we remove all narrow passages from the given polygon, P_Δ , and get a sequence of (sub-)polygons P_i , $i = 1, \dots, k$, without narrow passages. Again, we consider such a (sub-)polygon P_i and show by an induction on the number of split cells in the first layer that $S(P_i) \leq \frac{4}{3}C(P_i) - \frac{4}{3}$ holds.

With no split cell in the first layer, we can apply Lemma 19 and Lemma 20: $S(P_i) \leq C(P_i) + E(P_i) - 6 \leq C(P_i) + \frac{1}{3}C(P_i) + \frac{14}{3} - 6 = \frac{4}{3}C(P_i) - \frac{4}{3}$.

If we meet a split cell, c , in the first layer, we have two cases. Either, the new component was never visited before (see Figure 24(i)), or we touch a cell, c' , that was already visited, see for example Figure 24(ii) and (iii). In the first case, let $Q := \{c\}$; in the second case let Q enclose the shortest path from c to c' .

Similar to Theorem 13, we split the polygon. In each case, we have $C(P_i) = C(P') + C(P'') - |Q|$ and $S(P_i) = S(P') + S(P'') - 2(|Q| - 1)$,

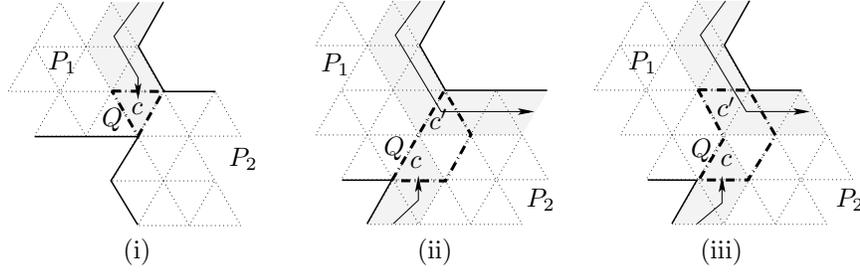


Figure 24: Three cases of split cells, (i) component of type II, (ii) and (iii) component of type I.

because Q is a corridor of width 1. Applying the induction hypothesis to P' and P'' yields

$$\begin{aligned}
 S(P_i) &= S(P') + S(P'') - 2|Q| + 2 \\
 &\leq \frac{4}{3}C(P') - \frac{4}{3} + \frac{4}{3}C(P'') - \frac{4}{3} - 2|Q| + 2 \\
 &= \frac{4}{3}C(P_i) - \frac{2}{3}|Q| - \frac{2}{3} \leq \frac{4}{3}C(P_i) - \frac{4}{3} \quad (|Q| \geq 1).
 \end{aligned}$$

The optimal strategy needs at least C steps, which, altogether, yields a competitive factor of $\frac{4}{3}$. The factor is achieved in a polygon as shown in Figure 25: for every two rows of 12 cells in the middle, SmartDFS $_{\Delta}$ needs 16 steps. Additionally, we need 10 steps for the first and the last row of 5 cells each. Hence, for $2n + 2$ rows we have $\frac{10+16n}{10+12n}$ which converges to $\frac{4}{3}$. \square

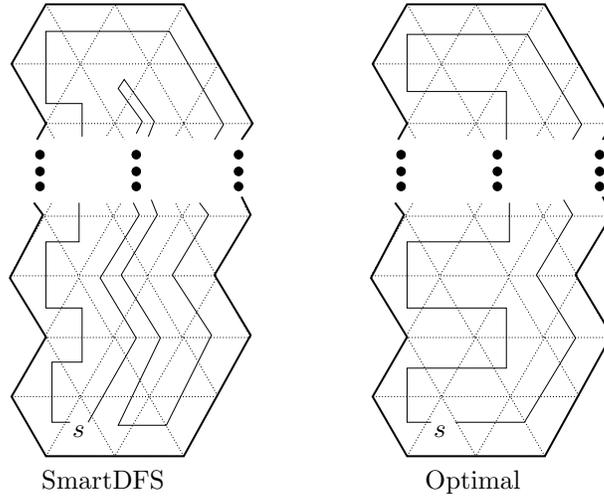


Figure 25: A polygon where the competitive factor of SmartDFS $_{\Delta}$ is achieved exactly.

4.3 Summary

We considered the online exploration of hexagonal and triangular grid polygons and adapted the strategy SmartDFS.

For hexagonal polygons we gave a lower bound of $\frac{14}{13}$ and showed that SmartDFS_○ explores polygons with C cells and E edges using no more than $C + \frac{1}{4}E - 2.5$ steps. For triangular polygons we have a lower bound of $\frac{7}{6}$ (matching the lower bound for square polygons) and an upper bound of $C + E - 4$ on the number of steps. Further, we showed that both strategies are $\frac{4}{3}$ -competitive, and that the analysis was tight (i.e., there are polygons where a factor of $\frac{4}{3}$ is exactly achieved).

An interesting observation is that—although the all three problems appear to be the same at first sight—there are some subtle differences, that are caused by the differences in the ‘connectivity’ of the grids: there are no touching cells in hexagonal grids, which seems to make the problem easier (and—in turn—makes it harder to find a lower bound). On the other hand, the lower number of neighboring cells in triangular grids allows the assumption that there are more steps needed in these kind of polygons. Indeed, we ‘need’ all edges in triangular polygons for the upper bound on the number of steps, while in square and hexagonal polygons a fraction of the edges is sufficient. In this connection, one may ask whether it is appropriate to consider—basically—the same strategy for all types of grids, or one should regard more grid-specific details in the design of an exploration strategy.

An interesting open problems is how to close the gap between the upper bound and the lower bound on the competitive factors.

References

- [1] S. Albers, K. Kursawe, and S. Schuierer. Exploring unknown environments with obstacles. *Algorithmica*, 32:123–143, 2002.
- [2] E. Arkin, S. Fekete, K. Islam, H. Meijer, J. S. Mitchell, Y. Núñez, V. Polishchuk, D. Rappaport, and H. Xiao. Not being (super)thin or solid is hard: A study of grid hamiltonicity. submitted, 2007.
- [3] E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell. Approximation algorithms for lawn mowing and milling. *Comput. Geom. Theory Appl.*, 17:25–50, 2000.
- [4] E. M. Arkin, J. S. B. Mitchell, and V. Polishchuk. Two new classes of hamiltonian graphs. In *Proc. Europ. Conf. Comb. Graph Theo. Appl.*, volume 29C of *e-Notes Discr. Math*, pages 565–569, 2007.
- [5] 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.

- [6] W. W. R. Ball and H. S. M. Coxeter. *Mathematical Recreations and Essays*. Dover Publications, New York, 13 edition, 1987.
- [7] M. A. Batalin and G. S. Sukhatme. Efficient exploration without localization. In *Proc. IEEE Internat. Conf. Robot. Autom.*, 2003.
- [8] P. Berman. On-line searching and navigation. In A. Fiat and G. Woeginger, editors, *Competitive Analysis of Algorithms*. Springer-Verlag, 1998.
- [9] M. Betke, R. L. Rivest, and M. Singh. Piecemeal learning of an unknown environment. *Machine Learning*, 18(2-3):231-254, 1995.
- [10] A. M. Bruckstein, M. Lindenbaum, and I. A. Wagner. Distributed covering by ant-robots using evaporating traces. *IEEE Trans. Robot. Autom.*, 15:918-933, 1999.
- [11] A. M. Bruckstein, M. Lindenbaum, and I. A. Wagner. Mac vs. pc - determinism and randomness as complementary approaches to robotic exploration of continuous unknown domains. *Internat. J. Robotics Res.*, 19(1):12-31, 2000.
- [12] H. Choset. Coverage for robotics - A survey of recent results. *Ann. Math. Artif. Intell.*, 31:113-126, 2001.
- [13] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Boston, 2005.
- [14] X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment I: The rectilinear case. *J. ACM*, 45(2):215-245, 1998.
- [15] G. Dudek, E. Miliotis, and I. M. Rekleitis. Multi-robot collaboration for robust exploration. *Ann. Math. Artif. Intell.*, 31:7-40, 2001.
- [16] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer*, 22(6):46-57, 1989.
- [17] H. Everett. Hamiltonian paths in non-rectangular grid graphs. Report 86-1, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, 1986.
- [18] Y. Gabriely and E. Rimon. Competitive on-line coverage of grid environments by a mobile robot. *Comput. Geom. Theory Appl.*, 24:197-224, 2003.
- [19] 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.

- [20] 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/>.
- [21] F. Hoffmann, C. Icking, R. Klein, and K. Kriegel. The polygon exploration problem. *SIAM J. Comput.*, 31:577–600, 2001.
- [22] C. Hwan-Gue and A. Zelikovsky. Spanning closed trail and Hamiltonian cycle in grid graphs. In *Proc. 6th Annu. Internat. Sympos. Algorithms Comput.*, volume 1004 of *Lecture Notes Comput. Sci.*, pages 342–351. Springer-Verlag, 1995.
- [23] C. Icking, T. Kamphans, R. Klein, and E. Langetepe. Exploring grid polygons online. Technical Report 001, Department of Computer Science I, University of Bonn, December 2005. <http://web.informatik.uni-bonn.de/I/publications/ikkl-egpol-05.pdf>.
- [24] C. Icking, T. Kamphans, R. Klein, and E. Langetepe. Exploring simple grid polygons. In *11th Internat. Comput. Combin. Conf.*, volume 3595 of *Lecture Notes Comput. Sci.*, pages 524–533. Springer, 2005.
- [25] K. Islam, H. Meijer, Y. N. Rodríguez, D. Rappaport, and H. Xiao. Hamilton circuits in hexagonal grid graphs. In *Proc. 19th Canad. Conf. Comput. Geom.*, pages 85–88, 2007.
- [26] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM J. Comput.*, 11:676–686, 1982.
- [27] T. Kamphans. *Models and Algorithms for Online Exploration and Search*. Dissertation, University of Bonn, 2005. <http://www.kamphans.de/k-maole-05.pdf>.
- [28] S. Koenig and Y. Liu. Terrain coverage with ant robots: a simulation study. In *Proc. 5th internat. Conf. Auton. agents*, pages 600–607, 2001.
- [29] J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple new method for the geometric k -MST problem. In *Proc. 7th ACM-SIAM Sympos. Discrete Algorithms*, pages 402–408, 1996.
- [30] J. S. B. Mitchell. Shortest paths and networks. In J. E. Goodman and J. O’Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 24, pages 445–466. CRC Press LLC, Boca Raton, FL, 1997.

- [31] 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.
- [32] H. P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proc. IEEE Internat. Conf. Robot. Autom.*, pages 116–121, 1985.
- [33] S. Ntafos. Watchman routes under limited visibility. *Comput. Geom. Theory Appl.*, 1(3):149–170, 1992.
- [34] V. Polishchuk, E. M. Arkin, and J. S. B. Mitchell. Hamiltonian cycles in triangular grids. In *Proc. 18th Canad. Conf. Comput. Geom.*, pages 63–66, 2006.
- [35] N. S. V. Rao, S. Karetí, W. Shi, and S. S. Iyengar. Robot navigation in unknown terrains: introductory survey of non-heuristic algorithms. Technical Report ORNL/TM-12410, Oak Ridge National Laboratory, 1993.
- [36] C. Umans and W. Lenhart. Hamiltonian cycles in solid grid graphs. In *Proc. 38th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 496–507, 1997.