An Improved Strategy for Exploring a Grid Polygon *

Agnieszka Kolenderska[1], Adrian Kosowski[1,2],
Michał Małafiejski[1], and Paweł Żyliński[3]

[1] Dept of Algorithms and System Modeling, Gdańsk University of Technology
{agnieszka,adrian,mima}@kaims.pl
[2] LaBRI - Université Bordeaux 1 - CNRS
kosowski@labri.fr
[3] Institute of Computer Science, University of Gdańsk
pz@inf.univ.gda.pl

Abstract. We study the problem of exploring a simple grid polygon using a mobile robot. The robot starts from a location which is adjacent to the boundary of the polygon, and after exploring all the squares, has to return to its starting location. The robot is equipped with memory, but has no prior knowledge of the explored terrain. The view of the terrain is restricted to the four squares directly adjacent to the robot's current location. The performance of the exploration strategy is measured in terms of the competitive ratio, with respect to the length of the optimal path for an exploration with complete knowledge of the terrain.
We propose a new exploration strategy which achieves a competitive ratio of 5/4, whereas the previously best approach [Icking, Kamphans, Klein, and Langetepe; *Proc. COCOON'05*] has a competitive ratio of 4/3. The analysis for our algorithm is tight. Moreover, we show that no exploration strategy is ever better than 20/17-competitive, thus improving the previous lower bound of 7/6.

Key words: Exploration problem, Path planning, Grid polygon, Competitive ratio.

1 Introduction

In this paper we investigate one of the basic path planning problems for a mobile robot, namely that of exploring the area of an unknown polygon. This type of geometric problem has received a lot of attention, both in a continuous and a discrete setting. In the continuous variant, sometimes referred to as the *milling problem* [2] or the *mobile robot covering problem* [5], the robot has non-zero spacial dimensions and moves along a continuous trajectory, with the goal of covering each point of the explored environment at least once. Herein, we focus on the discrete variant of the problem, defined for polygons composed of unit

* The research was partially funded by the KBN Grant 4 T11C 047 25, by the ANR project "ALADDIN", and by the INRIA project "CEPAGE".

cells; this can in fact also be regarded as a special case of a graph exploration problem.

The explored environment is assumed to be a simple two-dimensional orthogonal polygon with integer coordinates. The environment is divided into unit squares (called *cells*), with coordinates of corners belonging to the integer grid. The robot is always assumed to occupy the whole of a single cell. The robot's motion is restricted to one of four directions: East, West, North or South. Time is measured in discrete steps, and in one step, the robot can only move from one cell to a cell which is directly adjacent along a side (at a distance of one). The task of the robot is to perform an exploration of all the cells of the polygon and return to the cell of its initial location. The robot can only proceed through cells which are located within the explored polygon, and the starting cell is assumed to be adjacent to the boundary of the polygon.

In such a scenario, one can define different limitations to the robot's capabilities: its knowledge of the polygon, memory, ability to leave markers in the terrain. In this paper we assume that the robot is equipped with sufficient memory and processing power, but has no prior knowledge of the explored polygon. Upon entering a cell and before deciding on the next move, the robot has only a sense of the direction of its current heading (a compass), and a local view of the surroundings, indicating which of the four cells directly adjacent to its current location belong to the polygon. Thus, the next move in the exploration has to be computed based on the fragment of the map of the terrain which the robot has already discovered. The behaviour of the robot in some way resembles an on-line algorithm. Our goal is to design the best possible strategy for the robot, and to compare it to the optimal strategy for the off-line version of the problem, i.e., the scenario in which the robot is initially given a map of the entire terrain, together with a marker representing its initial location.

Related work. The off-line version of the problem can be rephrased in terms of the unweighted Travelling Salesman Problem (TSP) on a special class of graphs. The input graph is then some (planar) subgraph of the two-dimensional grid, with nodes defined as cells of the polygon, and edges connecting each cell with the four cells neighbouring along a side. This problem was shown to be NP-hard for polygons with holes by Itai, Papadimitriou and Szwarcfiter [12]; the hardness of the problem for polygons without holes remains an open problem. On the other hand, since the considered TSP instance is planar and has a natural metric, the problem admits a Polynomial-Time Approximation Scheme, using the techniques of Arora [3], Grigni, Koutsoupias, and Papadimitriou [8], or Mitchell [14]. These results hold regardless of whether the polygon is allowed to have holes or not. The case without holes also admits some simple and efficient approximation algorithms, for example the linear-time 6/5-approximation of Arkin, Fekete, and Mitchell [2].

For the case of a robot having only a local view of the adjacent cells, a simple exploration is achieved by performing a DFS exploration of the cells. In general, such a route is a 2-competitive solution to the TSP problem, since it traverses

Type of environment	Upper bound	Ref.	Lower bound	Ref.
Polygon with holes	$2C-2$	DFS	$2\,S_{OPT}-c$	[5,10]
	$B+C$	SpiralSTC [5]		
	$C+\frac{1}{2}E+W+3H-2$	CellExplore [10]		
Polygon without holes	$\frac{4}{3}\,S_{OPT}$	SmartDFS [11]	$\frac{7}{6}\,S_{OPT}-c$	[11]
	$\frac{5}{4}\,S_{OPT}$	**Thm. 2**	$\frac{20}{17}\,S_{OPT}-c$	**Thm. 3**

Table 1. Bounds on the cover length of an unknown polygon for a robot with local view (S_{OPT} – length of the optimal cover path with full knowledge of the polygon, C – polygon area, B – number of boundary cells, E – perimeter of the polygon, W – sinuosity of the polygon [13], H – number of holes, c – an additive constant).

twice each of the edges of the spanning tree of the cell graph. In fact, for polygons with holes such an approach is essentially the best possible: it is known [5,10] that there does not exist a strategy which achieves a competitive ratio better than 2. However, some strategies for covering polygons with holes were constructed by Gabriel and Rimon [5], and Icking et al. [10]. In these approaches, the length of the exploration is bounded in terms of the number of holes, area, perimeter, and certain other parameters of the polygon (see Table 1); hence, in some cases the performance of these strategies is better than that of DFS. For the case when the covered polygon has no holes, Icking et al. [11] put forward a modification of the DFS approach which they called SmartDFS. SmartDFS was shown in [11] to be a 4/3-competitive algorithm, and it was also proved that there does not exist an approach to the covering problem which is better than 7/6-competitive.

Recently, Herrmann, Kamphans, and Langetepe [9] have investigated the related problem of exploration with local view of polygons on the triangular and hexagonal grids, i.e., in settings where a cell is a triangle or a hexagon. They showed that there does not exist an algorithm which is better than 7/6-competitive for a triangular polygon, and better than 13/11-competitive for a hexagonal polygon. They also proposed a modified variant of SmartDFS for hexagonal and triangular polygons. The corresponding off-line versions of the problem, for a robot with full knowledge of the map in a triangular or hexagonal grid polygon, are known to be NP-hard [1,7].

Contribution and outline of the paper. The paper is organised as follows. In Section 2 we propose a new strategy with local view for exploring orthogonal grid polygons without holes, and prove that it achieves a competitive ratio of 5/4, thus improving the ratio of 4/3 achieved by the SmartDFS approach. We show that the analysis of the competitive ratio of our algorithm is tight. In Section 3 we provide a new lower bound of 20/17 on the competitive ratio of any

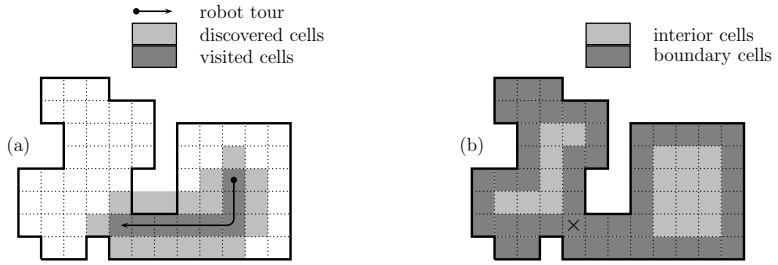


Fig. 1. (a) Discovered and visited cells. (b) Interior and boundary cells. The unique split cell is marked with a cross (×).

exploration algorithm with local view for the studied class of polygons. Some concluding remarks are made in Section 4.

Notation. We divide the polygon P and the surrounding plane into unit squares (cells). Each cell of the polygon is either not adjacent to the boundary of the polygon, or has at least one corner lying on the boundary. This divides the set of cells of the polygon into two disjoint sets: the *skeleton* (*interior cells*) and the *boundary* cells (Fig. 1(b)). We call a cell *discovered* if at least one of its neighbors has already been visited by the robot (Fig. 1(a)).

We will say that a cell $x \subset P$ is a *split* cell of a simple polygon P, if region $P \backslash x$ is not a polygon (or polygon with holes), i.e., the line of the outline of $P \backslash x$ is not self-avoiding; see Fig. 1(b) for an example. Note that $P \backslash x$ is a union of simple polygons; we will call he minimal number of disjoint simple polygons which cover polygon $P \backslash x$ the *multiplicity* of the split cell x. We define the *split number* of the polygon as $T = T_2 + 2T_3 + 3T_4$, where T_i is the number of split cells with multiplicity i, for $i = 2, 3, 4$.

The total number of cells of the polygon P is denoted by $C(P)$, or simply C if this does not lead to misunderstanding. We denote the length of the optimal length of a solution which can be achieved by a robot with full knowledge (a map) by S_{OPT}, and the length of the robot's path computed by algorithm A by S_A. We say that exploration algorithm A is α-competitive if there exists a constant c such that for any polygon P we have $S_A(P) \leq \alpha S_{OPT}(P) + c$.

Throughout the paper we will assume that the robot is located in a corner cell of the polygon, and that the initial heading of the robot is such that the cell behind and it and the cell directly to the left are outside the polygon, and the cell in front of the robot is within the polygon. It is straightforward to show that for the considered algorithms the starting cell can be moved to an arbitrary cell adjacent to the boundary, without affecting the competitive ratio (only the additive constant c). Indeed, if the starting cell s is a boundary cell, then $P \setminus s$ is a union of 1, 2, 3, or 4 simple polygons. We can explore each of these polygons

separately, starting and ending at a corner cell adjacent to s, and finally return to s at a constant additive cost.

2 A 5/4-Competitive Algorithm

Our approach to the problem relies on an extensive modification of the Depth First Search (DFS) strategy. A slightly extended implementation of DFS could work as follows: the robot maintains a stack of unvisited cells. At every step, it adds to unvisited the cells which are adjacent to the robot's location. Visited cells are purged from the unvisited stack, and the robot then proceeds along the shortest path to the top-most cell of the stack, if the stack is non-empty. If the stack is empty, this means that the whole polygon has been explored, and the robot returns to its starting location. This is also the general idea behind the SmartDFS approach put forward in [11]. In fact, such an exploration strategy is shown to achieve a competitive ratio of 4/3, as long as the adjacent cells are pushed onto the unvisited stack in the following order: first the cell to the right, then the cell in front of, and finally the cell to the left of the robot (with respect to its current heading). Intuitively, the robot will thus traverse the unvisited boundary cells of the polygon in the clockwise direction, as long as this is possible, and when some sub-polygon consisting of unvisited cells has been completely surrounded by the robot's path, the robot first embarks upon a sub-exploration of this polygon. A worst-case example for SmartDFS is an exploration of the rectangular polygon of dimensions $3 \times n$ [11] in which this strategy will, in particular, visit twice $n-3$ of the cells belonging to the skeleton of the polygon.

In order to avoid such a problem and to achieve a competitive ratio of 5/4, in our strategy, we modify the DFS approach by adding a special rule set for handling situations in which the polygon intuitively "narrows down to a width of 3". The proposed strategy will behave differently from SmartDFS when one of the cells adjacent to the robot's location is identified as either a so-called *dead-end* or a *bottle-neck*, and such cells will be visited first. Both these notions are formally defined below (see Fig. 2 for a simple example displaying the general idea of the modifications).

Definition 1. *A cell c is called a* dead-end *at a given step of exploration if all of the following conditions are jointly fulfilled:*

- *Cell c has not yet been visited by the robot and it is adjacent along a side to the cell where the robot is currently located.*
- *Exactly* 3 *of the cells adjacent to c along a side have already been visited by the robot.*
- *There does not exist any cell c' outside of polygon P, such that c' has already been discovered by the robot and c' is adjacent to c along a side or across a corner.*

Definition 2. *A cell c is called a* bottle-neck *at a given step of exploration if all of the following conditions are jointly fulfilled:*

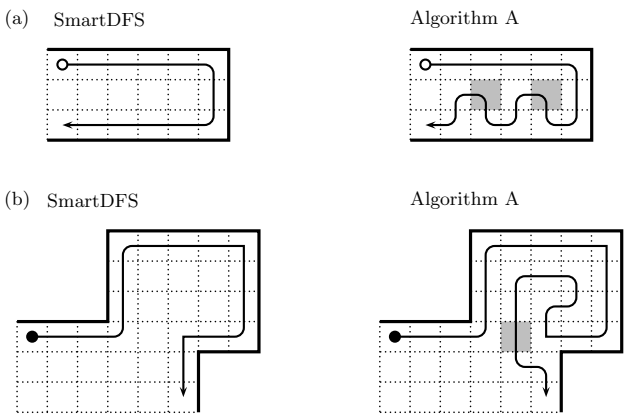


Fig. 2. Differences between our approach (Algorithm A) and SmartDFS [11] when handling: (a) dead-ends, (b) bottle-necks.

- *Cell c has not yet been visited by the robot and it is adjacent along a side to the cell where the robot is currently located.*
- *The union of the set of already explored cells and cell c surrounds all the sides of the outline of some polygon $P' \subset P$, such that P' consists of unvisited cells, only, and P' contains the cell adjacent to c along the right side of c (for directions as understood when entering c from the robot's current location).*
- *There does not exist any cell c' outside of polygon P', such that c' has already been discovered by the robot and c' is adjacent to c along a side.*
- *The cell located directly in front of the robot's location (with respect to its current heading) is located within the polygon.*

Observe that the definitions of a dead-end and a bottle-neck are such that the robot can always verify whether a given cell is a dead-end or a bottle-neck, based only on its current local view and its history of exploration.

2.1 The Algorithm

Our approach is defined precisely as follows. Throughout execution, the robot maintains a stack of cells called unvisited (as in DFS) and an additional stack called priority, used to implement the modified rules applied when encountering a dead-end or a bottle-neck. Both these stacks are initially empty. At each step of exploration, the precise procedure applied by the robot is given in the form of Algorithm A. In the pseudocode, the rules marked with the symbol (*) denote those, which are an extension with respect to the standard DFS approach. The robot terminates exploration when it re-enters its initial location, and the unvisited stack is empty.

2.2 Analysis of the Competitive Ratio

Observe that for any polygon, the optimal length of the off-line cover path fulfills the bound $S_{OPT} \geq C + T$. This is because in any exploration, each of the C

Algorithm A Procedure for determining the next move of the robot.

At each time step do:

1. Update the stacks of unvisited and priority cells:
   - Push on top of the unvisited stack the 3 adjacent cells which are directly to the right of the robot, in front of the robot, and to the left of the robot (with respect to its current heading), in precisely that order, i.e., with the left cell coming topmost.
   - Search through the unvisited stack and purge from it all those cells, which have already been visited by the robot.
   - (*) Search through the priority stack and purge from it all those cells, which have already been visited by the robot.
2. Set the destination of the robot:
   - (*) If the priority stack is non-empty, then set: destination := top of the priority stack.
   - If the priority stack is empty, then:
     - If the unvisited stack is non-empty, then set: destination := top of the unvisited stack.
     - Otherwise, set: destination := initial location of the robot
3. (*) If the current location of the robot is adjacent to a dead-end or a bottle-neck, then alter the destination of the robot:
   - Push the current destination on top of the stack of priority cells.
   - Set destination := location of the adjacent dead-end/bottle-neck. If there is more than one dead-end or bottle-neck directly adjacent to the robot, we give preference to the one to the right of the robot, then the one in front of the robot, finally the one to the left of the robot.
4. Perform a move of the robot:
   - If the current destination is not a bottle-neck, move the robot to an adjacent cell which is closer to the destination than the robot's current location (with respect to the shortest path metric within the set of cells already discovered by the robot). If more than one such cell exists, break ties by arbitrarily choosing a cell which has not yet been visited.
   - (*) Otherwise, move the robot to its destination in the minimum number of steps, so as to cover on the way the entire sub-polygon P' (the unvisited sub-polygon surrounded by previously visited cells and the bottle-neck cell destination). The optimal route can always be determined, since the boundary of the covered sub-polygon is known.

cells of the polygon has to be visited at least once, and each of the split cells has to be visited more times, depending on the multiplicity of the split cell. Hence, we confine ourselves to bounding the length of path S_A with respect to $(C + T)$. More precisely, to prove that the approach given by Algorithm A has a competitive ratio of 5/4, we will establish the following claim.

Theorem 1. *For any polygon P of area $C > 2$, we have $S_A \leq \frac{5C+5T-3}{4}$.*

Proof. First, observe that until a bottle-neck or dead-end are encountered, the algorithm always progresses along the boundary of the polygon. Thus we obtain a class of "narrow" polygons which are traversed optimally by Algorithm A.

Lemma 1. *If Algorithm A does not encounter any bottle-necks or dead-ends for polygon P, then $S_A = C + T$.*

To prove the theorem, let us now assume that P is a counter-example for our claim, i.e., a polygon such that $S_A(P) > \frac{5C(P)+5T(P)-3}{4}$, which is minimal in the following sense: out of all counter-examples, P has the minimal area of its skeleton, and of all such polygons, it has the minimal area $C(P)$. We will show through a sequence of lemmas that polygon P does not exist.

First, we obtain that P must have a non-empty skeleton.

Lemma 2. *For any polygon P, such that the skeleton of the polygon P is empty and $C > 2$, Algorithm A covers P in $S_A \leq \frac{5C+5T-3}{4}$ steps.*

The proof of the lemma relies on the observation that a polygon with an empty skeleton cannot contain a 3×3 polygon as a sub-polygon; we postpone it to the Appendix.

As a consequence of the lemma, we may assume that the skeleton of P is non-empty. Now, we consider the structure of the boundary cells of any polygon P. For a boundary cell c, let $dist(c) \geq 1$ denote the length of the shortest path from c to a nearest cell of the skeleton of P, according to the metric in which cells adjacent along sides or across corners are at a distance of 1 from each other (i.e., boundary cells adjacent to the skeleton along a side or across a corner have $dist(c) = 1$).

Note that the set of boundary cells of a polygon has an empty skeleton. We now observe that for our minimal counter-example P, the polygon cannot contain any boundary cells with $dist(c) > 2$. The proof uses simple local arguments (similar to those in the proof of Lemma 2) to show that otherwise there would exist a smaller counter-example P' to our claim; we omit the details from this extended abstract.

Lemma 3. *For the minimal counter-example P, any boundary cell c fulfills $dist(c) \leq 2$.*

We now proceed to the main part of the proof. Taking into account the above lemma and the fact that there exists only a finite number of polygons having a given skeleton, subject to the condition $dist(c) \leq 2$ for any boundary cell c, we will now analyze the structure of polygon P by characterizing its skeleton, only.

A skeleton of a polygon will be called *elementary* if it forms a connected region of the plane (possibly across corners) and consists of at most 5 cells. There exist exactly 83 elementary skeletons which are distinct up to isometry. Taking into account Lemma 3, by an exhaustive computer search, we verify the following claim.

Lemma 4. *The minimal counter-example P does not have an elementary skeleton.*

To complete the proof of the theorem, we now show that there does not exist a minimum counter-example P which has a non-elementary skeleton. First, observe that if polygon P had no dead-ends and no bottle-necks, then it would not be a valid counter-example by Lemma 1. Likewise, supposing that P had no bottle-necks, and the only dead-ends which appeared in the exploration were traversed as shown in Fig. 3(a). Then, we immediately obtain $S_A(P) \leq \frac{7}{6}C(P) + T(P)$, and once again P is not a valid counter-example. Hence, at some point during the exploration we must encounter a bottle-neck or a dead-end which is traversed differently than those from Fig. 3(a). Now, we apply a local replacement argument to show that the claim of the theorem holds for P if it holds for all polygons with a skeleton smaller than that of P. The part of the polygon in which we apply the replacement is determined by the earliest step during the exploration of P, at which any one of the following events occurs:

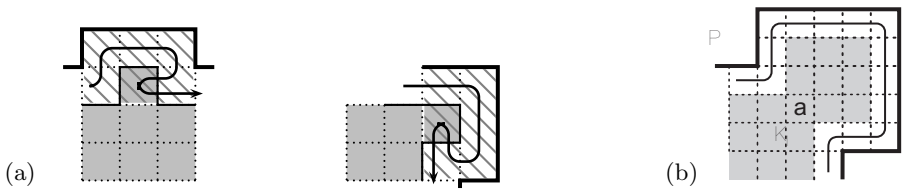


Fig. 3. Situations encountered during exploration: (a) special kinds of dead-ends, (b) split cell a of the set of unvisited cells (across a corner).

(1) A dead-end is encountered, and it is traversed differently than that shown in Fig. 3(a).
(2) A bottle-neck is encountered.
(3) A cell a is encountered, such that a is adjacent across a corner to the robot's location, and a is a split cell with respect to the set of unvisited cells of the polygon, see Fig. 3(b).

The proof is completed by analysing each of the cases (1), (2), and (3); we postpone this to the Appendix. □

2.3 A Tight Example

Theorem 2. *The competitive ratio of Algorithm A is exactly* 5/4.

Proof. The tight example is obtained by exploring the polygon shown in Fig. 4. The length of the optimal off-line tour is $S_{OPT} = C + 2$, and the length of robot's tour computed by Algorithm A is $S_A = \frac{5C-4}{4}$. The ratio of these two values precisely corresponds to the competitive ratio of the algorithm:

$$\frac{S_A}{S_{OPT}} = \frac{5C-4}{4C+8} \longrightarrow_{C\to\infty} \frac{5}{4}.$$

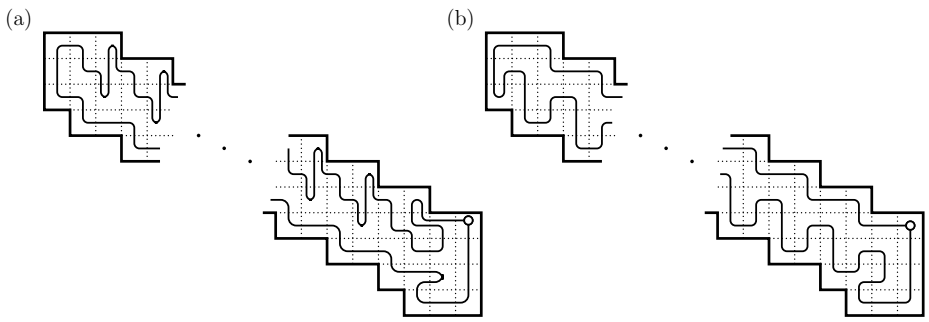


Fig. 4. A tight example for the competitive ratio: **(a)** The exploration performed by Algorithm A. **(b)** The optimal off-line covering path. The exploration starts from the cell marked with the circle.

3 Lower Bound on the Competitive Ratio

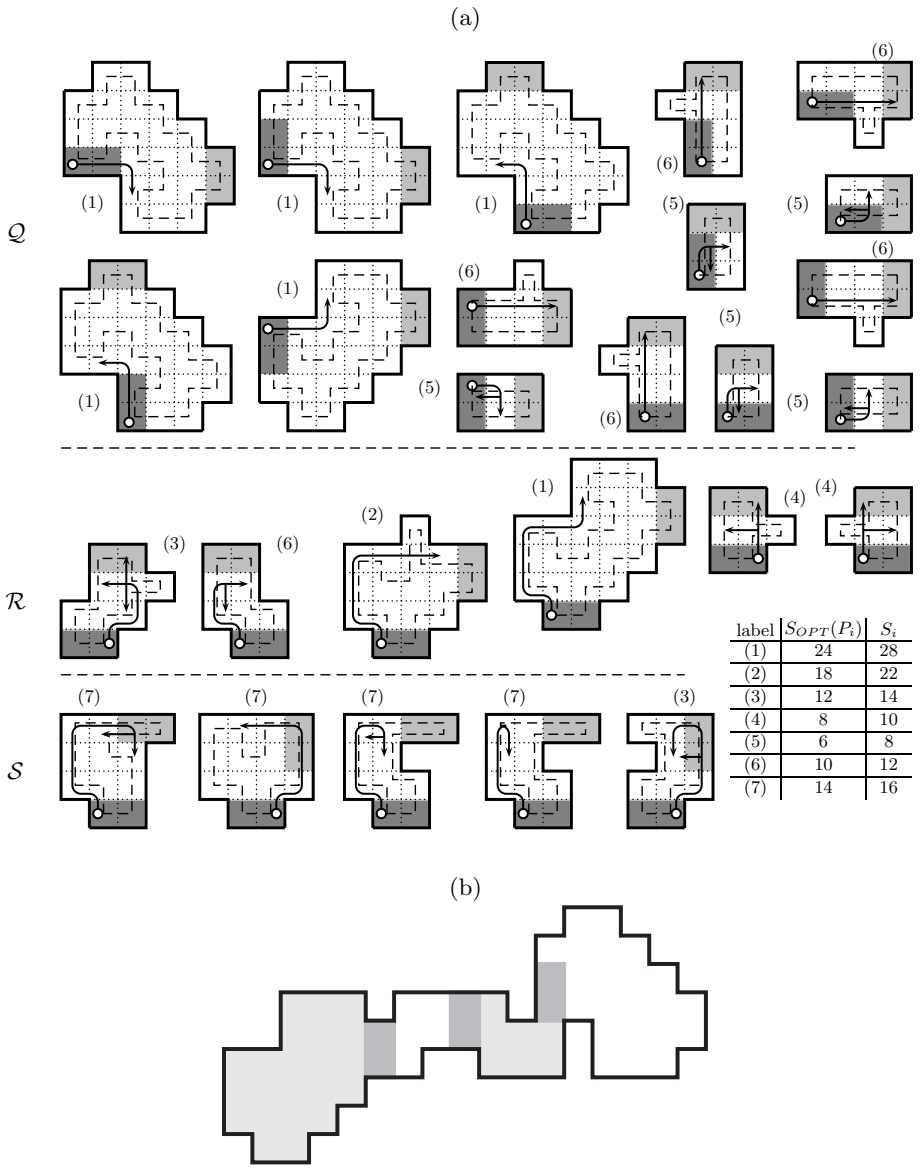
Icking et al. [11] have shown that no exploration strategy of an orthogonal polygon is better than $\frac{7}{6}$-competitive. We obtain a tighter bound by substantially extending their construction of the family of polygons which serves as a counter-example.

Theorem 3. *Let A be an exploration algorithm using a local view. If A is α-competitive, then $\alpha \geq \frac{20}{17}$.*

Proof. To prove the theorem, we construct a family of polygons $\mathcal{P}$ in such a way that, for any exploration algorithm E, there exists a polygon $P \in \mathcal{P}$ of arbitrarily large area $C(P)$, such that $S_E(P) \geq \frac{20}{17} S_{OPT}(P) - 2$.

The considered family $\mathcal{P}$ is built up by connecting into a "chain" the elementary polygons from sets $\mathcal{Q}$, $\mathcal{R}$, $\mathcal{S}$, shown in Fig. 5. More precisely, we put $\mathcal{P} = \bigcup_{n \geq 1} \mathcal{P}_n$, and elements of the family $\mathcal{P}_n$, for all $n \geq 1$ are constructed from exactly n polygons from $\mathcal{Q} \cup \mathcal{R} \cup \mathcal{S}$ according to the approach described below. For $P \in \mathcal{P}_n$, we will write $P = (P_1, P_2, \ldots, P_n)$, where we have $P_i \in \mathcal{Q} \cup \mathcal{R} \cup \mathcal{S}$, and the following rules are applied:

- We have $P = P_1 \cup P_2 \cup \cdots \cup P_n$.
- The intersection $P_i \cap P_j$ is non-empty only for polygons adjacent in the sequence, i.e., when $|i - j| \leq 1$.
- The intersection $P_i \cap P_{i+1}$, for all $1 \leq i < n$, consists of exactly two adjacent cells of each of these polygons, as shaded in Fig. 5, located next to their North-East and South-West corners, respectively. Note that not every two polygons $P_i, P_{i+1} \in \mathcal{Q} \cup \mathcal{R} \cup \mathcal{S}$ can be put together so as to fulfill this condition.
- The following additional rules are fulfilled: $P_1 \in \mathcal{Q}$. Moreover, for all $1 \leq i < n$, we have $P_{i+1} \in \mathcal{Q}$ if and only if $P_i \in \mathcal{S}$.



label	$S_{OPT}(P_i)$	S_i
(1)	24	28
(2)	18	22
(3)	12	14
(4)	8	10
(5)	6	8
(6)	10	12
(7)	14	16

Fig. 5. Construction of polygons for the lower bound: (a) Sets of polygons $\mathcal{Q}$, $\mathcal{R}$, $\mathcal{S}$. Sets $\mathcal{R}$ and $\mathcal{S}$ also contain the respective polygons reflected with respect to the line North-East / South-West. (b) Example of a polygon belonging to $\mathcal{P}_4$.

Given any algorithm E, for any $n \geq 1$ we now construct by means of E a sub-optimal robot's route $\tau = (c_0, c_1, c_2, \ldots, c_S)$ covering some polygon $P = (P_1, P_2, \ldots, P_n) \in \mathcal{P}_n$, see Appendix B. Here, c_t denotes the location of the robot

after t steps of exploration, S is the length of the route, and $c_S = c_0$. Let S_i denote the length of route τ when restricted to the cells belonging to polygon P_i, i.e., the subsequence of τ obtained by removing all cells from outside P_i and then compacting identical adjacent elements. By the construction of polygon P, we have: $S \geq 2 + \sum_{1 \leq i \leq n}(S_i - 2)$. Moreover, the property of the constructed route τ is such that $S_i \geq S_{OPT}(P_i)$, with the minimal possible value of S_i for different types of polygons $P_i \in \mathcal{Q} \cup \mathcal{R} \cup \mathcal{S}$ listed in Fig. 5. Taking into account that for any polygon $P \in \mathcal{P}_n$ we have $S_{OPT}(P) = 2 + \sum_{1 \leq i \leq n}(S_{OPT}(P_i) - 2)$, we obtain:

$$\frac{S}{S_{OPT}(P)} \geq \frac{\sum_{1 \leq i \leq n}(S_i - 2)}{\sum_{1 \leq i \leq n}(S_{OPT}(P_i) - 2)} - O(1/n).$$

By finding the minimum possible value of the above expression for the values stated in Fig. 5, subject to the constraint that not more than $n/2$ of the polygons P_i belong to $\mathcal{S}$ (by the construction of family $\mathcal{P}_n$), we obtain $\frac{S}{S_{OPT}(P)} \geq \frac{20}{17} - O(1/n)$. Such a bound is in fact asymptotically tight when we have, for odd values of i, that $P_i \in \mathcal{Q} \cup \mathcal{R}$ with $S_i = 28$ and $S_{OPT}(P_i) = 24$, whereas for even values of i, $P_i \in \mathcal{S}$ with $S_i = 16$ and $S_{OPT}(P_i) = 14$. Note that: $\frac{(28-2)+(16-2)}{(24-2)+(14-2)} = \frac{20}{17}$. This completes the proof of the theorem.

4 Final remarks

We have presented a new exploration strategy for a robot with limited view in a grid polygons, having an improved competitive ratio of 5/4. The strategy from Algorithm A requires memory which is linear with respect to the input size. Moreover, by modifying the final step of the algorithm so that the sub-exploration of the cut-off polygon using a recursive call to Algorithm A, rather than by the optimal off-line approach, one can also implement the local actions of the robot in polynomial time, without affecting the competitive ratio of the algorithm.

It would be interesting to ask about the effect of additional restrictions on the memory of the robot on the competitive ratio of the approach. In the wider context of graph exploration, explorations with bounded memory have been the topic of intensive study [4,6,15]. For our problem, it is easy to see that $\Theta(\log C)$ memory is necessary and sufficient to cover the polygon and to terminate at the starting location after $O(C)$ steps, where C is the area of the polygon. This is because the location of the robot can be identified by using coordinates of size $\Theta(\log C)$, whereas exploration of a single column of the polygon, as well as traversal along its boundary, require at most constant memory. It would be interesting to study the trade-off between the amount of allowed memory and the precise value of the competitive ratio of the strategy. One may also ask about competitive strategies in which the robot has extremely limited memory, but is allowed to leave and detect pebbles on the cell of its location and the cells adjacent to it.

Finally, we note that the lower bound on the competitive ratio shown in Section 3 holds for any exploration algorithm in the considered scenario, but

it only describes the worst-case performance of algorithms. It is possible that there exist randomized algorithms with a competitive ratio better than 20/17 in expectation; we leave this as a topic for future study.

Acknowledgement. We acknowledge the use of resources of TASK Academic Supercomputer Center to run a benchmark of Algorithm A against a test set of 10^{11} small polygons, and thank Jacek Dąbrowski for his kind assistance.

References

1. E. M. Arkin, S. P. Fekete, K. Islam, H. Meijer, J. S. B. Mitchell, Y. Nunez-Rodriguez, V. Polishchuk, D. Rappaport, and H. Xiao. Not being (super)thin or solid is hard: A study of grid hamiltonicity. *Computational Geometry: Theory and Applications*, 42(6-7):582–605, 2009.
2. E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry: Theory and Applications*, 17(1-2):25–50, 2000.
3. S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998.
4. P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg. Graph exploration by a finite automaton. *Theoretical Computer Science*, 345(2-3):331–344, 2005.
5. Y. Gabriely and E. Rimon. Competitive on-line coverage of grid environments by a mobile robot. *Computational Geometry: Theory and Applications*, 24(3):197–224, April 2003.
6. L. Gąsieniec, A. Pelc, T. Radzik, and X. Zhang. Tree exploration with logarithmic memory. In: *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms (SODA'07)*, pp. 585–594, 2007.
7. V. S. Gordon, Y. L. Orlovich, and F. Werner. Hamiltonian properties of triangular grid graphs. *Discrete Mathematics*, 308(24):6166 – 6188, 2008.
8. M. Grigni, E. Koutsoupias, and C. Papadimitriou. An approximation scheme for planar graph TSP. In *Proceedings of the Thirty-Sixth Annual IEEE Symposium on the Foundations of Computer Science(FOCS '95)*, pages 387–411, 1995.
9. D. Herrmann, T. Kamphans, and E. Langetepe. Exploring simple triangular and hexagonal grid polygons online. In *Abstracts of the 24th European Workshop on Computational Geometry*, pages 177–180, 2008.
10. C. Icking, T. Kamphans, R. Klein, and E. Langetepe. Exploring an unknown cellular environment. In *Abstracts of the 16th European Workshop on Computational Geometry*, pages 140–143, 2000.
11. C. Icking, T. Kamphans, R. Klein, and E. Langetepe. Exploring simple grid polygons. In *Proceedings of the 11th International Computing and Combinatorics Conference (COCOON'05)*, Vol. 3596 of Lecture Notes in Computer Science, pages 524–533, 2005.
12. A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982.
13. T. Kamphans. Models and Algorithms for Online Exploration and Search. Ph.D. thesis, Rheinischen Friedrich-Wilhelms-Universität Bonn, 2005.
14. J. S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999.

15. O. Reingold. Undirected ST-Connectivity in Log-Space. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC'05)*, pages 376–385, 2005.

A Analysis of Algorithm A (Theorem 1)

A.1 Proof of Lemma 2

Proof. Clearly, no bottle-necks are ever encountered in the exploration of polygon P. Moreover, if polygon P has no dead-end, then the algorithm covers P in $C+T$ steps. If a dead-end is encountered in P, the proof proceeds by induction on the area of P.

Let us consider the first dead-end encountered during the exploration. Let us mark this cell by 6, as shown in Fig 6(a). Then, three of the cells adjacent to the cell 6 have already been visited. Let us denote these cells by $1, 3, 5$, respectively. Moreover, all the enumerated cells must belong to the polygon, and the cell marked with x does not belong to the polygon. With no loss of generality we may thus assume that the polygon is of the shape shown in Fig. 6(b). Let us divide the polygon into two parts, P_1 and P_2, as shown in the figure. It is immediate to see that $C(P_1) > 2$, hence the inductive claim holds for P_1. There are two possible cases.

- If $C(P_2) = 2$ then $C(P) = C(P_1) + 6$, $T(P) = T(P_1) + 2$ and

$$S(P) = S(P_1) + 10 \leq \frac{5}{4}(C(P_1) + T(P_1) - 3 + 8) = \frac{5}{4}(C(P) + T(P) - 3)$$

- If $C(P_2) > 2$ then $C(P) = C(P_1) + C(P_2) + 4$ and $T(P) = T(P_1) + T(P_2) + 2$.

$$\begin{aligned} S(P) = S(P_1) + S(P_2) + 8 &\leq \frac{5}{4}(C(P_1) + T(P_1) - 3) + \frac{5}{4}(C(P_2) + T(P_2) - 3) + 8 \\ &< \frac{5}{4}(C(P) + T(P) - 3) \end{aligned}$$

Fig. 6. Dead-end encountered during the exploration of a polygon with an empty skeleton.

A.2 Analysis of Case (1): a dead-end

We can assume that P is a polygon with a skeleton size of n, and that the skeleton of P is not elementary. Then, next to the encountered dead-end (traversed differently than in Fig. 3(a)), the skeleton of polygon P must either include one of endings shown in Fig. 7, or else be disconnected.

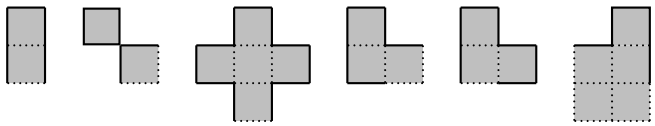


Fig. 7. Possible endings of the skeleton. Solid lines denote the outline of the skeleton.

We now attempt to cut off a part of polygon P to obtain a new polygon P', in such a way that the robot's tour within P' is the same as within P, except for a local modification. By an analysis of all possible cases, it can be shown that when considering an ending of the skeleton, at least one of the modifications from Fig. 8-12 can be applied. If the skeleton is disconnected, and these modifications cannot be applied, then one of the modifications from Fig. 13 can be applied instead.

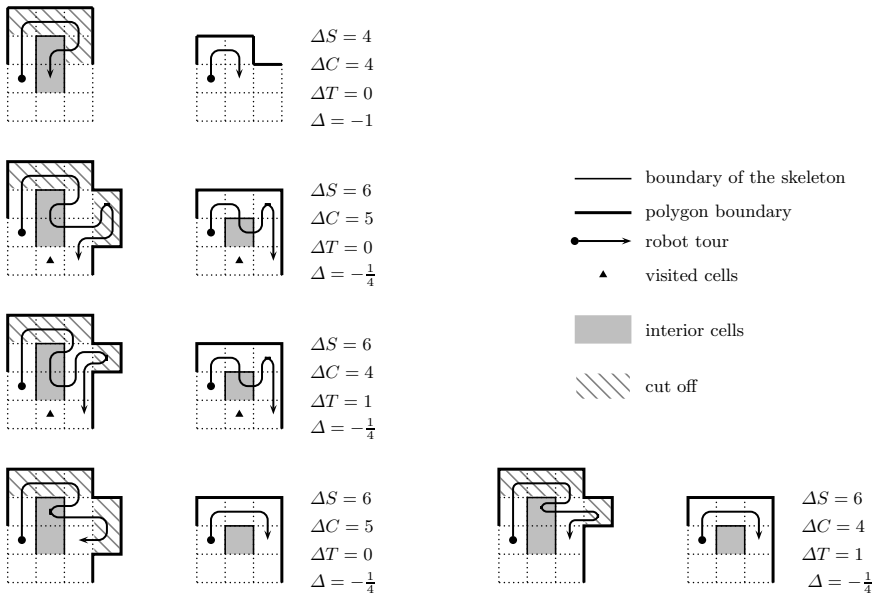


Fig. 8. Reductions of endings of the polygon skeleton. Polygon P is shown in odd columns, the outcome P' of the modification is shown in even columns.

Let us now compute the value of the following expression

$$\Delta = \Delta S - \frac{5}{4}(\Delta C + \Delta T) \tag{1}$$

where: $\Delta S = S_A(P) - S_A(P')$, $\Delta C = C(P) - C(P')$, $\Delta T = T(P) - T(P')$.

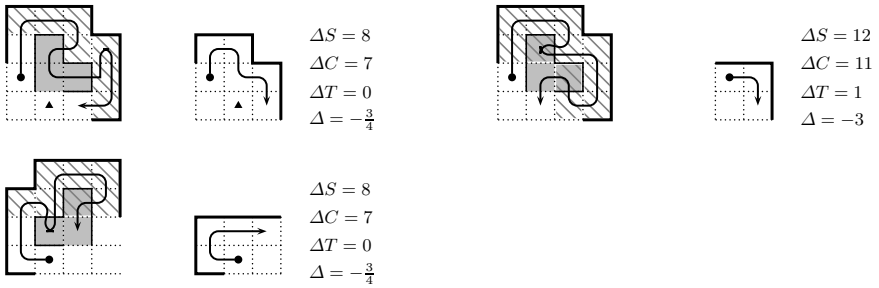


Fig. 9. Reductions of endings of the polygon skeleton (continued).

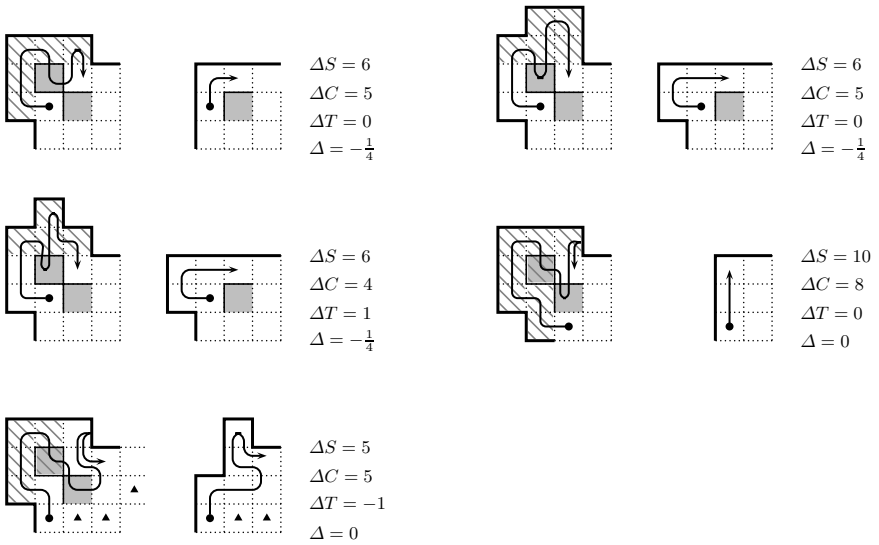


Fig. 10. Reductions of endings of the polygon skeleton (continued).

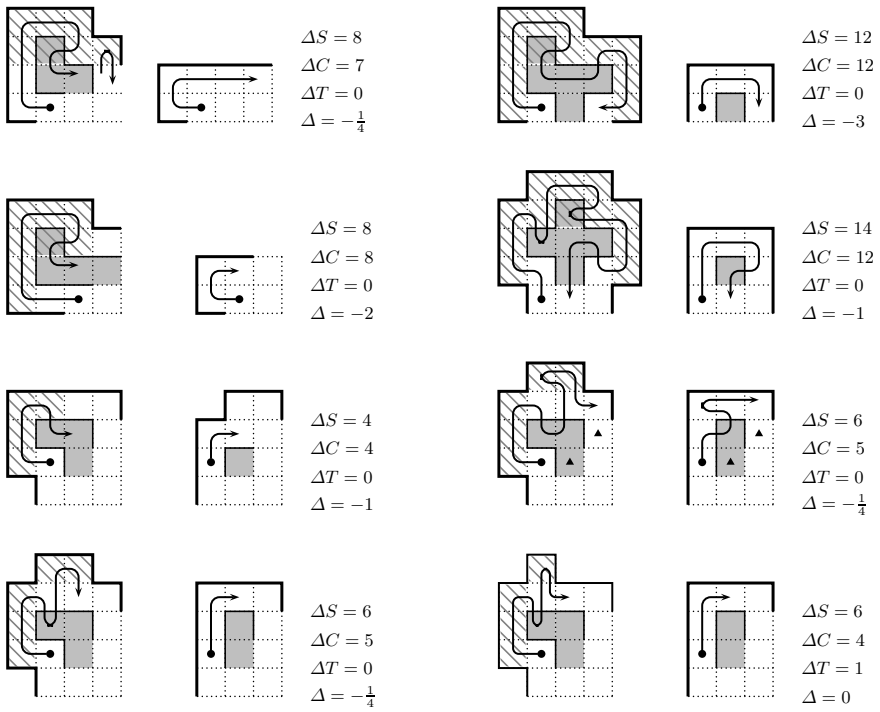


Fig. 11. Reductions of endings of the polygon skeleton (continued).

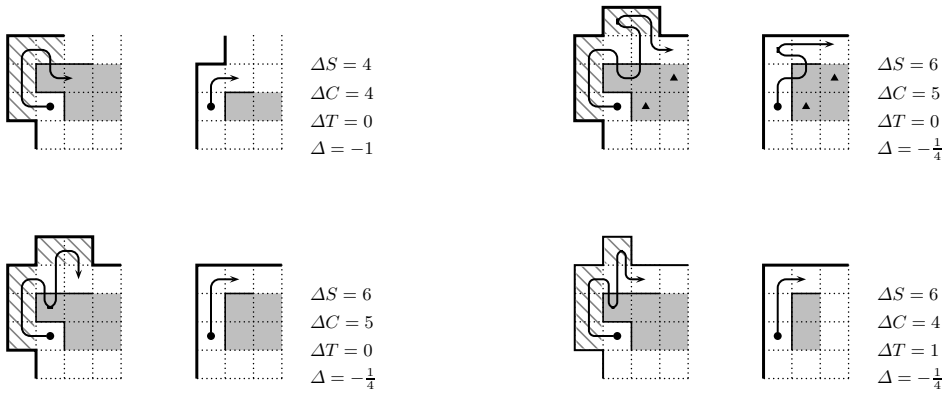


Fig. 12. Reductions of endings of the polygon skeleton (continued).

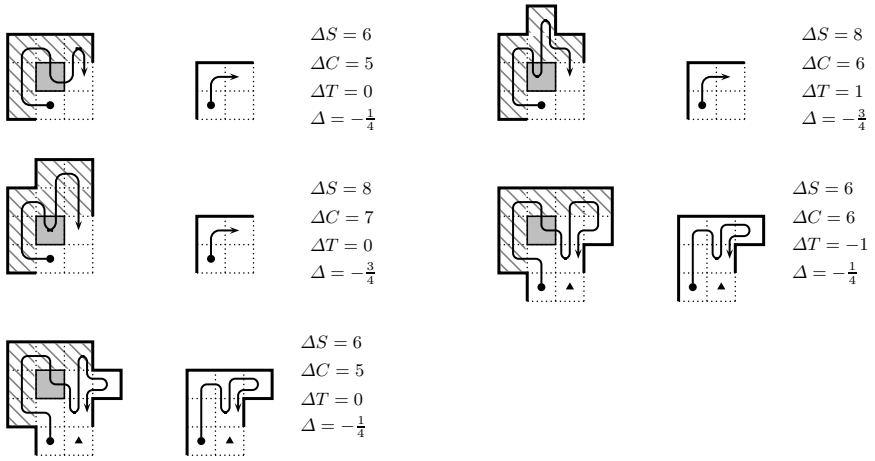


Fig. 13. Reductions for a disconnected skeleton of the polygon. Polygon P is shown in odd columns, the outcome P' of the modification is shown in even columns.

Since P' has a smaller skeleton than P, and P was by assumption the minimal counter-example to the claim, we have:

$$S_A(P') \leq \frac{5C(P') + 5T(P') - 3}{4}. \tag{2}$$

Consequently:

$$S_A(P) \leq S_A(P') + \frac{5}{4}(\Delta C + \Delta T) \leq \frac{5}{4}C(P) - \frac{3}{4} + \frac{5}{4}T(P) = \frac{5C(P) + 5T(P) - 3}{4},$$

hence, P is not a counter-example to the claim.

A.3 Analysis of Case (2): a bottle-neck

A schematic list of possibilities corresponding to this case is shown in Fig. 14, where c denotes the bottle-neck (note that in some cases c is adjacent to the outline of P along an edge, yet it remains a bottle-neck since the robot is unaware of this at the time when the bottle-neck is identified). For each of these cases, we define a polygon K_1 as shown in Fig. 14, as the polygon consisting of cut-off unvisited cells, possibly augmented by the bottle-neck c and the cell of the robot's current location. A crucial observation is that since by definition of Case (2), since no relevant dead-ends or split cells across a corner were detected before the bottle-neck, by the properties of Algorithm A, polygon K_1 will have an empty set of split cells, $T(K_1) = 0$.

As shown in Fig. 14, for each of the possibilities, we divide polygon P into two polygons P_1 and P_2, such that $P = P_1 \cup P_2$, $K_1 \subset P_1$, up to local modifications, the robot visits cells of polygon P, when restricted to each of the sub-polygons

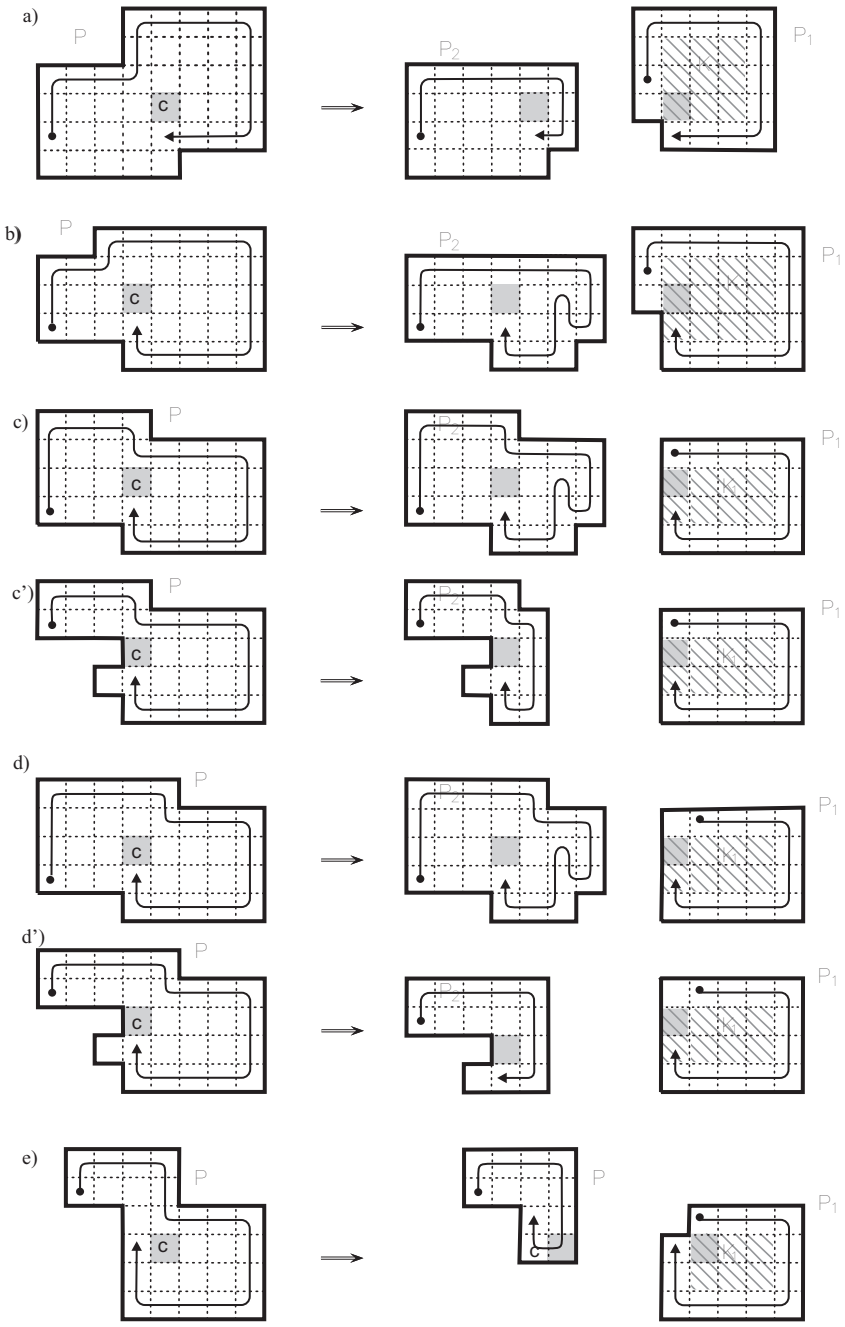


Fig. 14. Possible arrangements of cut-off polygons next to a bottle-neck.

P_1 and P_2, in the same order as for each of these polygons explored separately. It is also evident that $T(P) = T(P_1) + T(P_2)$.

The length $S_A(P_1)$ of the robot's tour of P_1 is easily decomposed into the sum of three components, $S_A(P_1) = S_K + S_{L1} + S_{L2}$ or $S_A(P_1) = S_K + S_{L1} + S_{L2} + 1$ (depending on the considered case), where:

- S_K is the length of the sub-exploration for polygon K_1. Since $T(K_1) = 0$, we have $S_K \leq \frac{1}{4}(5C(K_1) - 3)$ by the inductive assumption in Theorem 1.
- S_{L1} is the length of the exploration associated with the region L_1, which consists of those cells of the skeleton of P_1 and those cells of the first layer of boundary cells of P_1 (i.e., cells c' with $dist(c') = 1$ in P_1), which do not belong to K_1. Taking into account Fig. 3(a), we put $S_{L1} \leq \frac{7}{6}C(L_1)$.
- S_{L2} is the length of the exploration associated with the region L_2, which consists of the second layer of boundary cells of P_1 (i.e., cells c' with $dist(c') = 2$ in P_1). We have $S_{L2} = C(L_2) + T(P_1)$.

Hence, $S_A(P_1) \leq \frac{1}{4}(5C(K_1) - 3) + \frac{7}{6}C(L_1) + C(L_2) + T(P_1)$ or $S_A(P_1) \leq \frac{1}{4}(5C(K_1) - 3) + \frac{7}{6}C(L_1) + C(L_2) + T(P_1) + 1$. By taking into account that $C(P_1) = C(K_1) + C(L_1) + C(L_2)$, for sufficiently large polygons (with $C(K_1) > 10$) we obtain the following relations:

- For cases (a,e): $S_A(P_1) \leq \frac{5}{4}C(P_1) + \frac{5}{4}T(P_1) - 2$.
- For cases (b): $S_A(P_1) \leq \frac{5}{4}C(P_1) + \frac{5}{4}T(P_1) - 3\frac{1}{4}$.
- For cases (c-d'): $S_A(P_1) \leq \frac{5}{4}C(P_1) + \frac{5}{4}T(P_1) - 2\frac{3}{4}$.

The above relations also hold when $C(K_1) \leq 10$; we verify this by exhaustively checking all cases.

We now observe that since P_2 also has a smaller skeleton than P, by assumption we have from Theorem 1, $S_A(P_2) \leq \frac{1}{4}(5C(P_2) + 5T(P_2) - 3)$. Moreover, the following relations hold:

- For cases (a,c',d'): $C(P) = C(P_1) + C(P_2) - 8$ and $S_A(P) \leq S_A(P_1) + S_A(P_2) - 8$.
- For case (b): $C(P) = C(P_1) + C(P_2) - 17$ and $S_A(P) \leq S(P_1) + S_A(P_2) - 18$.
- For cases (c,d): $C(P) = C(P_1) + C(P_2) - 15$ and $S_A(P) \leq S(P_1) + S_A(P_2) - 16$.
- For case (e): $C(P) = C(P_1) + C(P_2) - 3$ and $S_A(P) \leq S(P_1) + S_A(P_2) - 2$.

In each of the cases (a-e), by combining the corresponding relations for $C(P)$, $S_A(P)$, $S_A(P_1)$, and $S_A(P_2)$, we once again obtain $S_A(P) \leq \frac{1}{4}(5C(P) + 5T(P) - 3)$, hence P is not a counter-example to the claim.

A.4 Analysis of Case (3): a split cell across the corner

Finally, we consider the case when the robot first encounters a split-cell a, as shown in Fig. 15. Note that such a split cell is not a bottle-neck, hence the robot will embark on a sub-exploration of polygon K_1 only when it reaches a bottle-neck in some later step. Nevertheless, with respect to cell a we can divide polygon P into two polygons P_1 and P_2, such that $P = P_1 \cup P_2$, $K_1 = K_{11} \cup K_{12}$,

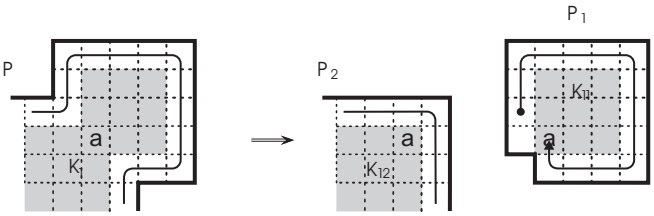


Fig. 15. Local reduction of the polygon next to a corner split-cell of the set of unvisited cells.

$K_{11} \subset P_1$, and, up to local modifications, the robot visits cells of polygon P, when restricted to each of the sub-polygons P_1 and P_2, in the same order as for each of these polygons explored separately. We note that polygon K_{11} has no split cells. Acting similarly as in Case (2), we obtain $T(P) = T(P_1) + T(P_2)$, $C(P) = C(P_1) + C(P_2) - 8$, and $S(P_1) \leq \frac{5}{4}C(P_1) + \frac{5}{4}T(P_1) - 2$. This leads to the relation $S_A(P) \leq \frac{1}{4}(5C(P) + 5T(P) - 3)$, hence once again, P is not a counter-example to the claim of the theorem.

B Lower bound: covering a polygon $P \in \mathcal{P}_n$

The construction below simulates an adversary which discloses local views to the robot. In this way, a polygon $P \in \mathcal{P}_n$ is selected depending on the decisions taken by the robot during its traversal τ, so that traversal τ is sub-optimal for polygon P. We define three separate blocks of rules for disclosing the view (cases (1), (2), and (3)), each of which reveals to the robot exactly one polygon P_i. Case (1) reveals a polygon $P_i \in \mathcal{Q}$, while cases (2) and (3) reveal a polygon from either $\mathcal{R}$ and $\mathcal{S}$. We will repeat such a procedure exactly n times. Initially, the first polygon P_1 is constructed according to case (1); subsequent choices of cases (1), (2), and (3) depend on the route chosen by the robot.

The robot starts with an initial view described in Fig. 16 (a), and assume without loss of generality that it decides to goes East in the first step.

(1) The cell North (Fig. 16 (a)), South (Fig. 16 (a')) or West(Fig. 16 (a")) to the robot does not belong to the polygon and in the previous step robot moved East, East or North, respectively.
First, let's consider case (a). Further, the robot can move in one of three directions: South, West or East, which is shown if Fig. 16 (b) and (c).
   – If the robot moves due South or West we do not add any cells to the polygon adjacent to a bold edge of the polygon (Fig. 16 (b)). Thus the robot must visit two of the white cells twice. So, the competitive ratio in the white region is equal to at least $\frac{8}{6}$, which is more than $\frac{20}{17}$.
   – If the robot moves due East we add some cells to the polygon, as demonstrated in Fig. 16 (c). Now, the robot can move in one of four directions: East, West, South and North, as shown in Fig. 16 (d-e).

In the cases shown in Fig. 16 (b,d) we add some cells to the polygon. We observe that the robot eventually has to visit at least two of the white cells twice.

In the case shown in Fig. 16 (e) we add some cells to the polygon. We observe that the robot eventually has to visit four of the white cells twice.

Once the robot has entered one of the gray cells, we continue the construction as described in cases (2) and (3), depending on the orientation of the robot.

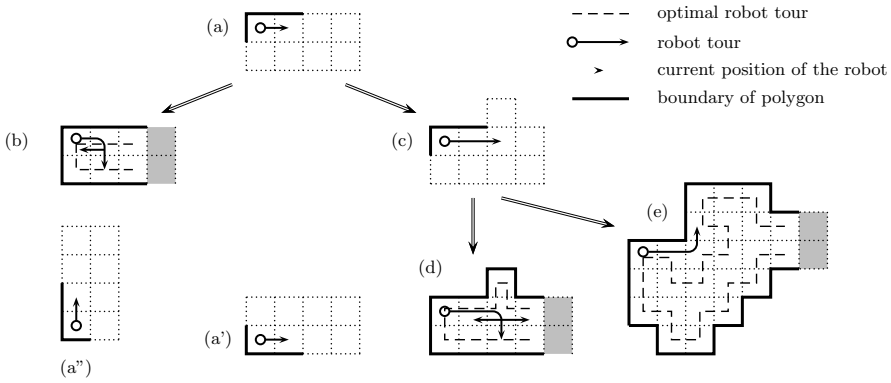


Fig. 16. Adversary vs. robot: construction of $P_i \in \mathcal{Q}$ according to case (1).

The construction of the polygon in cases (a') and (a") is identical (up to isometry of the plane) to case (a).

(2) In the previous step the robot moved North (Fig. 17 (a)) or East (Fig. 17 (a')).
The analysis of both cases is identical (up to isometry of the plane), so we describe only the case from Fig. 17 (a).
Considering the situation illustrated in Fig. 17 (a), the robot can now move to the North, South, East or West, which is shown in Fig. 17 (b)-(d).
   - If the robot moves due North, South or East we add new cells to the polygon, as demonstrated in Fig. 17 (b)-(c). In later steps, once the robot has moved out from the visible part of polygon, we continue construction according to cases (2) and (3).
   - If the robot moves West, we add some new cells to the polygon as shown in Fig. 17 (d). In the next step, the robot must move back (East) or move North.
     - If the robot moves East, we add new cells to the polygon according to Fig. 17 (e) (we do not depict the robot's path for this case). If at some later time the robot moves out from the visible part of polygon we continue the construction according to cases (2) and (3).

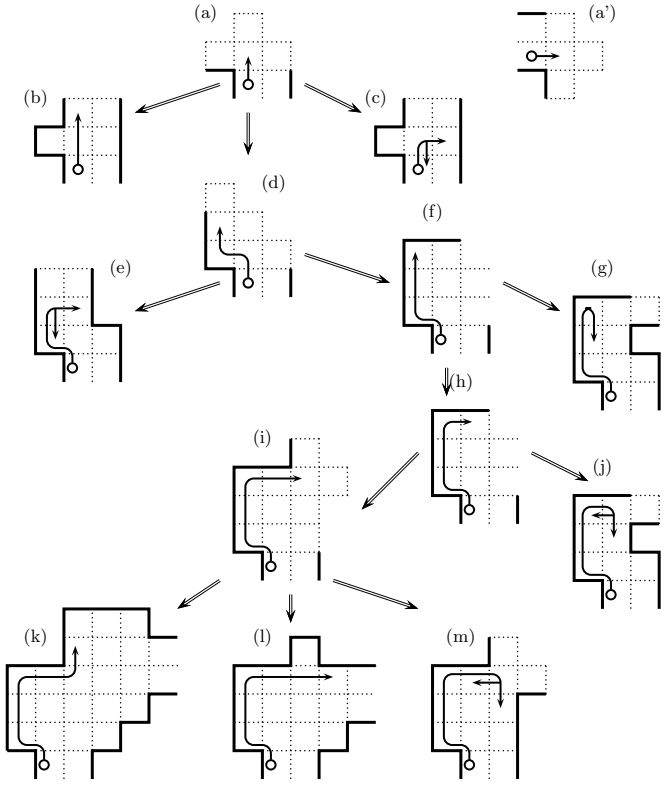


Fig. 17. Adversary vs. robot: construction of $P_i \in \mathcal{R} \cup \mathcal{S}$ according to case (2).

- If the robot moves North, then in the next step it can move East, move North, or move back (South).
  - ∗ If the robot moves East or South, we also add new cells to the polygon according to Fig. 17 (e). If at some later time the robot moves out from the visible part of polygon we continue the construction according to cases (2) and (3).
  - ∗ If the robot moves North (Fig. 17 (f)), we add some cells to the polygon. It can now either turn back (move South), or move East.
    - · If the robot moves South, we add new cells to the polygon, as shown in Fig. 17 (g). If at some later time the robot moves out from the visible part of polygon we continue the construction according to cases (2) and (3).
    - · If the robot moves East (Fig. 17 (h)), then in the next step it can move East, turn back (move West), or move South.
      – If the robot moves South or West, we add new cells to the polygon, as demonstrated in Fig. 17 (j). If at some later time the robot moves out from the visible part of polygon we continue the construction according to cases (2) and (3).
      – If the robot moves East we add new cells to the polygon, as demonstrated in Fig. 17 (i). We then likewise analyze one more step the robot can make, as illustrated in Fig. 17 (k)-(m). If at some later time the robot moves out from the visible part of polygon we continue the construction according to cases (1), (2) and (3).

(3) In the previous step robot moved North (Fig. 18 (a)) or East (Fig. 18 (a')). The analysis of both cases is identical (up to isometry of the plane), so we describe only the case from Fig. 18 (a).

- If the robot moves West, South or North we add new cells to the polygon, as shown in Fig. 18 (b)-(c). If at some later time the robot moves out from the visible part of polygon we continue the construction according to cases (2) and (3).
- In the third case we add some cells to the polygon. In the next step the robot must move North (Fig. 18 (d)) or come back West. If the robot moves North, then in the next step it can move North, West, East or come back to the South.
  - If the robot comes back to the West, or first moves North and then due North, South, or West, we add new cells to the polygon, as demonstrated in Fig. 18 (e). If at some later time the robot moves out from the visible part of polygon we continue the construction according to cases (2) and (3).
  - In the last remaining case we continue the construction according to case (1). When the robot enters the cell marked with a circle and moves from that cell to the West, we add new cells to the polygon according to Fig. 18 (f). Depending on the robot's subsequent steps,

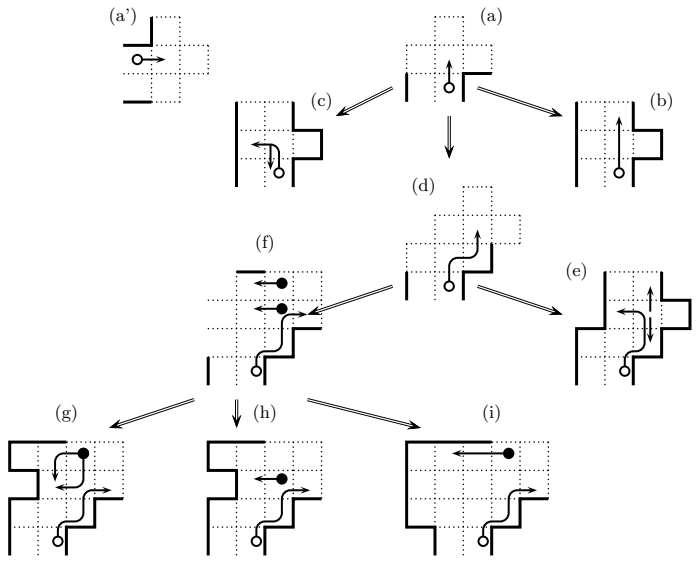


Fig. 18. Adversary vs. robot: construction of $P_i \in \mathcal{R} \cup \mathcal{S}$ according to case (3).

we add certain new cells to the polygon, as illustrated in Fig. 18 (g)-(i).

The presented construction always leads to a polygon which belongs to the family $\mathcal{P}_n$.