Journal of Combinatorial Optimization (2021) 41:625–639
https://doi.org/10.1007/s10878-021-00705-5

The simple grid polygon exploration problem

Qi Wei[1] · Jie Sun[1] · Xuehou Tan[2] · Xiaolin Yao[3] · Yonggong Ren[1]

Accepted: 19 January 2021 / Published online: 3 February 2021

Abstract
This paper considers an on-line exploration problem. We use a mobile robot to explore an unknown simple grid polygon. The robot is assumed to have limited sensing capability that can only detect the four basic cells adjacent to it. The robot's task is to explore each cell and to return to the start. To explore a cell, the robot has to enter it. We ask for a short exploration tour, that is, minimizing the number of multiple-visit cells. The competitive ratio is used to measure the performance of our strategy. It is the ratio between the length of the on-line exploration path and the length of the shortest path achieved with full knowledge of the polygon. As our main result, we present a new on-line exploration strategy and prove that the robot's path is at most 7/6 times longer than the shortest path. This matches the previously known lower bound.

Keywords Path planning · On-line Exploration · Competitive analysis · Grid polygon · Mobile robot

This work was supported in part by National Natural Science Foundation of China (No. 61702242, 61976109) and Doctoral Scientific Research Foundation of Liaoning Province (No. 2019-BS-153, 2019-BS-014).

✉ Qi Wei
qwei2009@hotmail.com

✉ Yonggong Ren
ryg@lnnu.edu.cn

1 School of Computer and Information Technology, Liaoning Normal University, South Liushu Street 1, Dalian 116081, China

2 School of Information Science and Technology, Tokai University, 4-1-1 Kitakaname, Hiratsuka 259-1292, Japan

3 School of Information Technology and Management, Dalian Neusoft University of Information, software park road 8, Dalian 116021, China

1 Introduction

Exploring an unknown environment is fundamental for many tasks of autonomous mobile robots. It has important applications when the environment is dangerous for humans Strom et al. (2017); Almeida et al. (2019), such as nuclear base or disaster area. The exploration problem has received much attention in computational geometry, online algorithms and robotics Ghosh and Klein (2010); Chalopin et al (2015); Foerster and Wattenhofer (2016).

During the exploration, the robot has to see all points of an unknown environment and return to the starting point. The length of the exploration path should be as short as possible because of the cost saving requirements. A lot of variations on capability of robot and type of environment have been studied Ghosh and Klein (2010); Ortolf and Schindelhauer (2012).

The unknown environment is modeled as a simple polygon in this paper. Deng et al. (1991) first showed that competitive strategy exists for polygon exploration. They proposed a 2016-competitive strategy inspired by a greedy off-line approach. Later, Hoffmann et al. (2002) improved the competitive ratio to 26.5, then it was improved to 6.7 by Tan and Wei (2015). There is still a gap between the competitive ratio 6.7 and the lower bound 1.28 Hagius et al. (2004).

In general, the robot is assumed to be equipped with a vision sensor that can see any point in its visibility region at any distance. This is not realistic, because any vision sensor can reliably gather information from the scene within a range. In this paper, we consider a short-sighted robot model. We assume that the robot can sense only four grid cells adjacent to its current position. Then, we subdivide the polygonal environment according to the size of the grid cell. We call this kind of environment the grid polygon. To explore the grid polygon, the robot must visit each grid cell at least once.

Exploration of grid polygon with or without holes has been both extensively studied. For grid polygon with holes, Gabriely and Rimon (2003) proposed two Spanning Tree Covering (STC) strategies and showed that the robot's path is at most $(n + m)$ times longer than the optimal off-line exploration path, where m and n denote the number of grid cells and the number of boundary cells respectively. They also proved that the lower bound for this problem is 2. Icking et al. (2002) proposed an exploration strategy whose competitive ratio is $A + \frac{1}{2}B + C + 3D - 2$, where A denotes area of the grid polygon, B denotes the perimeter of the grid polygon, C denotes the number of holes and D denotes the number of sinuosities. Except for the case that the gird polygon is a corridor of width one, it performs better than above two STC strategies. For grid polygon without holes, Icking et al. (2005) proposed an exploration strategy SmartDFS based on the modification of DFS approach, and showed that the competitive ratio is 4/3. They also proved that the lower bound for this problem is 7/6. Doi et al. (2018) proposed a strategy for exploring a simple grid polygon with the minimum number of modules. Keshavarz-Kohjerdi and Bagheri (2016) proposed a linear time algorithm for computing a Hamiltonian path in L-shaped grid polygon if it does exist.

Another scenario is the graph exploration. In this setting, the robot is located at a vertex of an unknown graph. It's task is to visit all vertices by traversing edges. Foerster and Wattenhofer (2016) considered the exploration of directed graph in both deterministic and randomized version. They proved upper and lower bounds for the competitive ratio. Dereniowski et al. (2015) proposed a collaborative graph exploration strategy which takes time $O(D)$, where D is the distance from the start node to the furthest node. Megow et al. (2012) proposed a graph construction to show whether a constant competitive exploration strategy for general graphs exists.

In this paper, we consider the exploration of an unknown simple grid polygon using a mobile robot. We present a new exploration strategy for this problem and show that the competitive ratio is 7/6. This result matches the lower bound proposed by Icking et al. (2005) and settles the open problem.

2 Preliminaries

This section formally defines the simple gird polygon exploration problem and gives some preliminary results.

2.1 Definitions

Let P be a simple grid polygon, a polygon consists of a list of basic cells and has no holes in it. Two basic cells p and q are adjacent to each other in case they have one common edge. We write N to denote the number of boundary edges and M to denote the total number of basic cells of P. N and M can also be considered as the perimeter and area of P.

The robot is modeled as a short-sighted one. It can only discover four adjacent cells. When a cell is discovered, the robot knows that it is blocked or free to visit. At the beginning, the robot has no previous information of P. It is located at a boundary cell s and has its back to the boundary of P. It can move from the current cell to one of the adjacent free cells in one step. The robot can store the discovered cells and make movement decisions based on them. The goal of the robot is to walk a closed path that visits each basic cell at least once and returns to the start cell s. Once a basic cell has been visited, we say it is explored. Note that all points of P have been seen when the robot completes this tour. We want the exploration tour to be as short as possible, so we should minimize the number of the cells visited more than once. See Fig. 1 for an example.

We denote a path from basic cell p to q by $\Pi(p, q)$. There is a sequence of basic cells contained in this path, $p, c_1, c_2, \ldots, q$. The robot can compute the length of $\Pi(p, q)$ if all cells in $\Pi(p, q)$ are already known to the robot. Since each basic cell has the same size, we use the number of steps from cell to cell to measure the length of the path. We denote the cell the robot currently located in by c_{cp}.

In this paper, the performance of the strategy is evaluated by the competitive ratio,

$$C = sup_P \frac{|\Pi(P)|}{|\Pi_{opt}(P)|},$$

where $|\Pi(P)|$ is the length of the robot's path, $|\Pi_{opt}(P)|$ is the length of the shortest path for the situation that the robot has full knowledge of P.

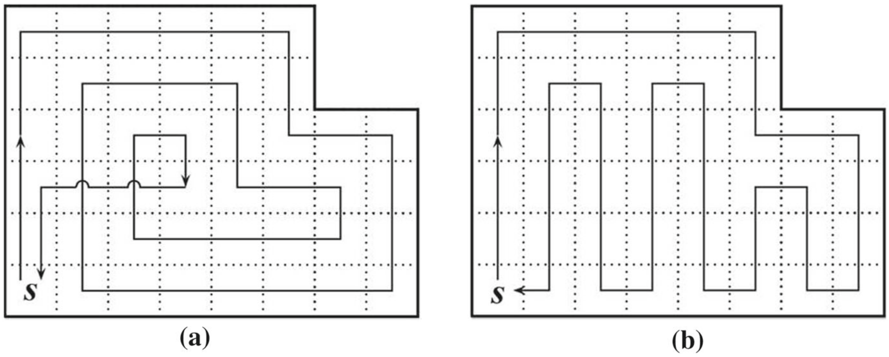

Fig. 1 **a** An instance of the exploration tour. **b** The shortest exploration tour

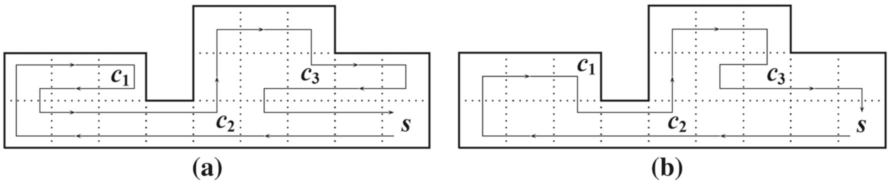

Fig. 2 **a** An exploration tour walked by DFS. **b** An exploration tour walked by SmartDFS

2.2 Preliminary results

For the problem of exploring an unknown simple grid polygon, depth-first-search (DFS) is a simple strategy which can be applied to complete the exploration task Icking et al. (2005). It follows a rule called left-hand to continue the path, that is, the robot prefers to turn left over go straight over turn right to an adjacent unexplored cell. When the robot enters a cell with no adjacent unexplored cells, it backtracks to the last cell with adjacent unexplored cell to go on the exploration. This yields an exploration tour with at most $2M - 2$ steps. Since $|\Pi_{opt}(P)|$ needs at least M steps, the competitive ratio of DFS is 2. See Fig. 2a for an example.

SmartDFS Icking et al. (2005) is a modified strategy based on DFS. Two improvements have been made to shorten the exploration tour. Firstly, when the robot enters a cell with no adjacent unexplored cells, it returns directly to the last cell with adjacent unexplored cell rather than backtracks to that cell. See Fig. 2 for an example. After c_1 has been visited, SmartDFS walks a shortest path to c_2. Note that the cells in the shortest path are already known to the robot. This is more efficient than DFS. Secondly, the robot can identify and handle a special kind of cell called split cell (see Definition 1). See Fig. 2 for an example. After c_3 has been visited, SmartDFS identifies it as a split cell. It divides the unexplored cells into two parts. Then, the robot visits these two parts in the following order: the part farther away from s, the part nearer to s. This is more efficient than DFS. SmartDFS reports a exploration tour at most 4/3 times longer than $|\Pi_{opt}(P)|$.

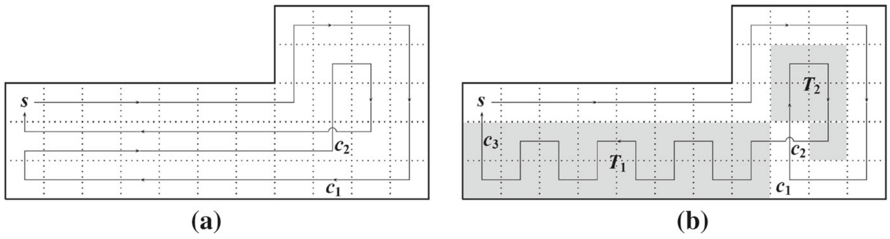

Fig. 3 The first improvement: Identify the split cell as early as possible. **a** An exploration tour walked by SmartDFS. **b** An exploration tour walked by our strategy

3 The exploration strategy

This section gives a new on-line exploration strategy based on SmartDFS. Two improvements have been made to shorten the exploration tour.

Definition 1 During the exploration, a basic cell is called a split one if it can divide the unexplored cells into several subgraphs.

The first improvement is to identify the split cell as early as possible. For each basic cell, our strategy identify whether it is a split cell when it has been discovered. This is different from SmartDFS. Note that SmartDFS identify the split cell while visiting it. See Fig. 3 for an example. After c_1 has been visited, SmartDFS continues its exploration with left-hand rule, whereas our strategy identifies c_2 as a split cell which divides the unexplored cells into two subgraphs, T_1 and T_2. Then our strategy explores T_2 and T_1 in turn. Obviously, our strategy is more efficient than SmartDFS.

To decide the visiting order of the divided subgraphs, we define a critical basic cell called $stage$ for each subgraph.

Definition 2 After each subgraph of P has been explored, our strategy guides the robot to a critical basic cell. It is called the corresponding stage cell of this subgraph.

A stage cell can be considered as the ending of the exploration of a subgraph. See Fig. 3b for an example. Here s and c_2 are stage cells of T_1 and T_2, respectively. P can be seen as the parent of T_1 and T_2, its stage cell is s. By the following visiting order rule, we decide the visiting order of the divided subgraphs.

1. At the beginning of the exploration, s is recognized as the $stage$ cell of P.
2. When the first split cell c_{s1} is identified by the robot, the unexplored cells divides into several subgraphs. P is the parent of these divided subgraphs, and s is their parent stage cell. Firstly, the robot finds the nearest cell c_r to the parent stage cell s from the cells discovered but not yet explored. Secondly, the robot makes a decision of the visiting order of the divided subgraphs. It prefers to visit the divided subgraph which contains c_r at last, and visits other divided subgraphs with the left-hand rule. Thirdly, the robot assigns the stage cell to each divided subgraph according to the above visiting order. The stage cell of the subgraph visited at last is its parent stage cell s; the stage cell of other subgraph is the split cell c_{s1}.

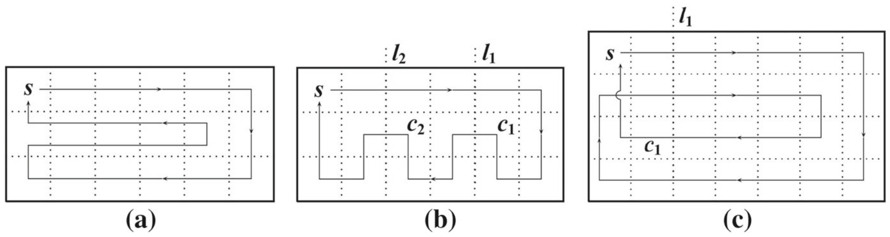

Fig. 4 The second improvement: Identify and handle the situation of a corridor of width 3. **a** An exploration tour walked by SmartDFS. **b** and **c** Exploration tours walked by our strategy

3. When the second split cell c_{s2} is identified in the subgraph of P, the robot decides the visiting order and assigns the stage cell as rule 2. And so on.

See Fig. 3b for an example. When c_2 is identified as the split cell, the robot finds the nearest cell c_3 to s from the cells discovered but not yet explored. Then the robot decides the visiting order: T_2, T_1. And assigns the stage cells c_2 and s to T_2 and T_1 respectively.

During the exploration of P, a sequence of split cells, $c_{s1}, \ldots, c_{sk}$ may occur. We handle these split cells in a recursive way. When a split cell $c_{si} (1 \leq i \leq k)$ is identified, we use the rule discussed above to decide the visiting order and assign the corresponding stage cell of each divided subgraph.

A subgraph of size $n \times 3$ ($n \geq 3$) limits the performance of SmartDFS because the interior cells should be visited twice. See Fig. 4a for an example. This situation is called a corridor of width 3. Since it is contained in most simple grid polygons, the handling of it is very important.

The second improvement is to identify and handle the situation of a corridor of width 3. To identify this situation, we define a critical basic cell called half-surrounded cell (see for example, c_1 and c_2 in Fig. 4b).

Definition 3 During the exploration, a basic cell c_h is called a half-surrounded one if: (i) c_h is adjacent to the robot's current location c_{cp}. (ii) c_h is discovered but not yet explored. (iii) There are 5 connected explored cells around c_h, including c_{cp}. They forms a U-shape and exactly 3 of them distribute along a side of c_h.

After a half-surrounded cell c_h has been identified, our strategy handles it in two cases. We make a tangent l to the opening of the U- shape at c_h. It divides P into two components. In the first case, the stage cell of current subgraph and c_h locate in different components, then the robot deviates from the left-hand rule and visits c_h next. In the second case, the stage cell of current subgraph and c_h locate in the same component, then the robot continues its exploration tour with left-hand rule. We call the c_h in the first case real half-surrounded cell and the c_h in the second case general half-surrounded cell. See Fig 4 for an example. In Fig. 4b, c_1 and c_2 are identified as real half-surrounded cells and handled as case 1. In Fig. 4c, c_1 is identified as general half-surrounded cell and handled as case 2. Obviously, our strategy behaves more efficiently than SmartDFS in the situation of a corridor of width 3.

Based on above two improvements, we propose our strategy listed in the following form.

Algorithm 1 Explore a simple gird polygon

```
1: procedure $ExploreGridPolygon(Polygon, start, stage)$;
2:     $ToVisit := NULL$;
3:     $update(ToVisit)$;
4:     while $ToVisit$ is not empty do
5:         $target := First(ToVisit)$;
6:         if $isSplitCell(target)$ then
7:             $split := target$;
8:             determine the visiting order of the divided subgraphs by the visiting order
               rule(Without loss of generality, assume that there are two subgraphs $T_1$ and
               $T_2$, and the visiting order is $T_2$, $T_1$);
9:             walk on the shortest path into $T_2$;
10:            $ExploreGridPolygon(T_2, s_2, split)$;
11:            walk on the shortest path into $T_1$;
12:            $ExploreGridPolygon(T_1, s_1, stage)$;
13:        else
14:            walk on the shortest path to $target$;
15:        end if
16:        $update(ToVisit)$;
17:    end while
18:    walk on the shortest path back to $stage$;
19: end procedure
```

The procedure $ExploreGridPolygon(Polygon, start, stage)$ is first called when the robot starts its exploration. The parameters $Polygon$, $start$ and $stage$ are P, s and s respectively. Throughout the execution of the procedure $ExploreGridPolygon$, our strategy maintains a stack called $ToVisit$. It is used to store the cell which have been discovered but not yet explored.

We use the function $update(ToVisit)$ to update $ToVisit$. Initially, it purges the visited cells from $ToVisit$. What's more, it identifies whether the adjacent unvisited cell is a split or a real half-surrounded cell. Finally, it pushes the adjacent unvisited cells except the split or real half-surrounded cell onto $ToVisit$ according to the left-hand rule, and then pushes the adjacent split or real half-surrounded cell onto $ToVisit$.

Our strategy repeatedly handles the first cell, $target$, of $ToVisit$ until $ToVisit$ becomes empty. We use the function $isSplitCell(target)$ to identify whether the target cell is a split cell or not. If it is a split cell, our strategy first decides the visiting order of the divided subgraphs by the visiting order rule discussed above, and then recursively calls the procedure $ExploreGridPolygon(Polygon, start, stage)$ to explore them in turn. Otherwise, our strategy guides the robot to the $target$ along the shortest path. Once the last target of $ToVisit$ has been explored, the robot walks on the shortest path back to the $stage$ cell.

4 Competitive analysis

This section analyzes the performance of the strategy by the competitive ratio. It shows that the robot's path is at most 7/6 times longer than the shortest path.

Starting form s, our strategy explores P layer by layer. It proceeds from outside to inside. The first layer is the boundary cells of P.

Definition 4 Let P_{li} denote the ith layer of P. $P - \sum_{j=1}^{i} P_{lj}$ is called the i-offset of P.

Note that the robot knows the cell's layer while visiting it. A useful property of i-offset is listed as follows. It shows the relation between $N(P)$ and $N(i\text{-offset})$, where $N(P)$ and $N(i\text{-offset})$ denote the number of boundary edges of P and i-offset respectively.

Lemma 1 *Icking et al. (2005) The i-offset of a simple grid polygon, P, has at least $8i$ edges fewer than P.*

The split cell is a critical one in the exploration. After it has been identified, current graph splits into several subgraphs. Now we analyze the handling of the split cell. For convenience, we give an instance shown in Fig. 5. After c_1 has been identified in the 4th layer, P splits into four subgraphs:

$$P = T_1 \cup T_2 \cup c_1 \cup \{explored\ cells\},$$

where T_1 and T_2 are divided subgraphs of unvisited cells. By the visiting order rule discussed in Sect. 3, the visiting order of the unvisited subgraphs can be decided as: $T_2 \cup c_1, T_1$. For convenience to analyze this situation, we extend T_1 and T_2 with several layers of explored cells. Then we get two extended polygons P_1 and P_2 such that P can be considered as a combination of them. There is an overlap square Q around c_1 between P_1 and P_2. We set the width of Q to be $2(a-1)+1$, where a is the number of the layer in which c_1 is located. We require that the exploration tour in $P_2 \backslash Q$ and in $P_1 \backslash Q$ are the same as that in P. We set P_2 such that $T_2 \cup c_1$ is the $(a-1)$-offset of P_2, and $P_1 = ((P \backslash P_2) \cup Q) \cap P$.

For the task of exploring a simple grid polygon P, each strategy needs no less than M steps, where M is the total number of basic cells of P. Thus, the length of the robot's path can be expressed as

$$|\Pi(P)| = M(P) + excess(P),$$

where excess(P) denotes the excess cells visited by the robot.

Lemma 2 *Assume that P is a simple grid polygon and c_1 is the first split cell identified by the robot. T_1, T_2, P_1 and P_2 are defined as above. Thus,*

$$excess(P) \le excess(P_1) + excess(T_2 \cup c_1) + 1.$$

Proof As discussed above, we know that there is no excess before c_1 has been identified. After that, the robot first visits $T_2 \cup c_1$ and returns to c_1. For the exploration in this subgraph, we consider $excess(T_2 \cup c_1)$ as the excess. Then the robot visits T_1 and returns to s. For the exploration in this subgraph, we consider $excess(P_1)$ as the excess. One step is added because c_1 is visited twice. Altogether, the bound of $excess(P)$ can be achieved. □

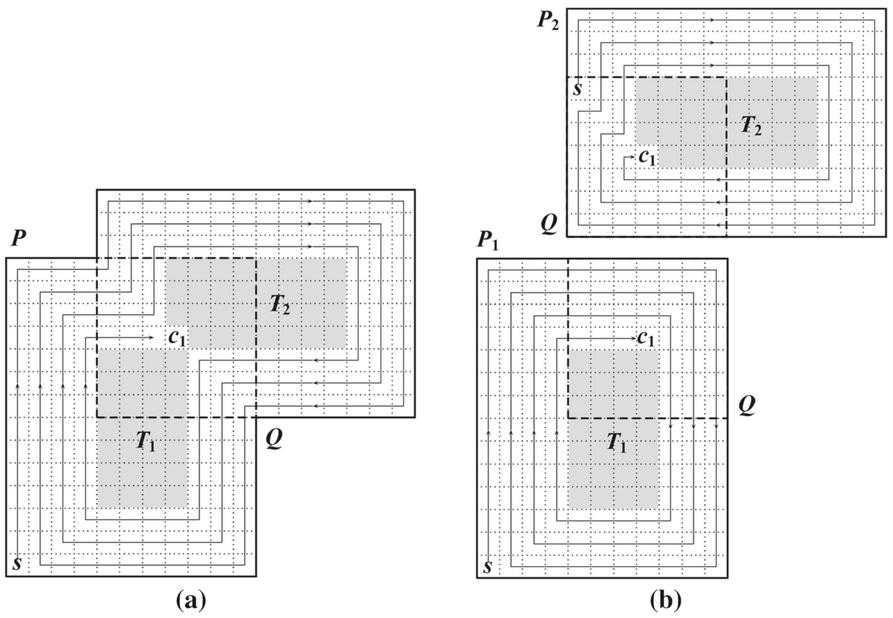

Fig. 5 The handling of the *split cell* in our strategy

Lemma 3 *Assume that P is a simple grid polygon and c_1 is the first split cell identified by the robot. T_1, T_2, P_1 and P_2 are defined as above. Thus,*

$$N(P_1) + N(P_2) = N(P) + N(Q).$$

Lemma 3 shows the relation between the number of edges of P_1 and P_2 and the number of edges of P and Q. See Fig. 5 for an example.

We call the corridor of width one or two narrow passage. Note that our strategy can explore narrow passages optimally. After all the cells of P have been visited, the robot has to return to the start cell s. The following lemma analyze the bound of the length of the return path $|\Pi_{return}(P)|$.

Lemma 4 *Assume that P is a simple grid polygon with N boundary edges and without any narrow passages. The length of the return path can be bounded by*

$$|\Pi_{return}(P)| \leq \frac{1}{4}N(P) - 1.$$

Proof We consider P in three cases to analyze the bound of $|\Pi_{return}(P)|$. In the first case, P is set to be a corridor of width three. Since there is no narrow passages in P, this case is the basis of the others. The robot starts from either a corner or other boundary cell. See for example, Fig. 6a, b. In both situations, we can achieve $|\Pi_{return}(P)| \leq 2$ by our strategy. Further, the number of the boundary cells of P satisfies $N(P) \geq 12$ because the length of P is greater than or equal to three (otherwise, P is a narrow

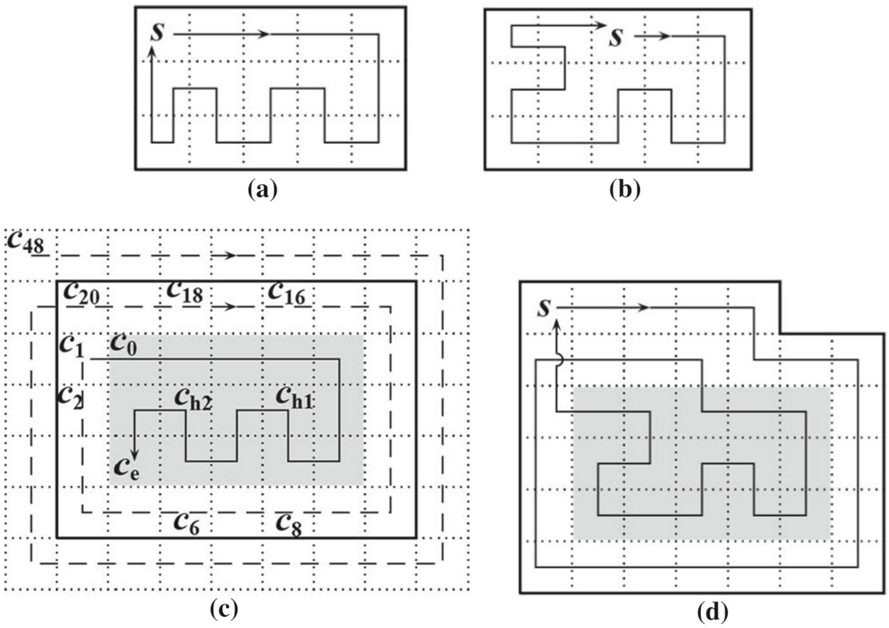

Fig. 6 Analysis of the length of the return path:**a** and **b** are two corridors of width 3, **c** and **d** are extended polygons of **a** and **b** respectively

passage) and the width of P is three. Thus, the bound $|\Pi_{return}(P)| \leq \frac{1}{4}N(P) - 1$ holds.

In the second case, we consider P as an extended polygon of a corridor of width three without split cells. See for example, Fig. 6c, d. We write P_c to denote the corridor contained in P. Each possible P in this case can be achieved by adding several cells onto P_c. We add the cell c_i $(i = 1, 2, \dots)$ anticlockwise around P_c one by one, while keeping the entry cell of P_c (c_0) the same. Once a cell has been added, a polygon P_i with starting cell c_i is achieved. See for example Fig. 6c. Two layers of cells c_i $(i = 1, 2, \dots, 48)$ have been added onto P_c, and a list of polygons P_i $(i = 1, 2, \dots, 48)$ have been achieved.

Now, we analyze the bound of $|\Pi_{return}(P_i)|$. Firstly, we consider P_i $(i = 1, 2, \dots, 20)$, i.e., extended polygons achieved by adding one layer of cells one by one around P_c. Note that P_c is the 1-offset of P_{20}. We write s_i and e_i to denote the first and last visited cell of P_i. c_{h1} and c_{h2} are two half-surrounded cells in P_c. Table 1 shows the bound of $|\Pi_{return}(P_i)|$. Since each of P_i $(i = 1, 2, 4, 5, 10, 11, 14, 15)$ has a narrow passage, we do not consider them in this case. For each P_i in Table 1, we compute $\frac{1}{4}N(P_i) - 1$ by $N(P_c) \geq 12$ discussed in case 1 and then compare it with $|\Pi_{return}(P_i)|$. Thus, the bound $|\Pi_{return}(P_i)| \leq \frac{1}{4}N(P_i) - 1$ holds. Secondly, we consider P_i $(i = 21, 22, \dots, 48)$. Note that P_c is the 2-offset of P_{48}. For each P_i $(i = 21, 22, \dots, 48)$ with no narrow passages, we can always find a grid polygon P_k which is achieved by adding the first layer of cells and satisfies $|\Pi(e_i, s_k)| \leq |\Pi_{return}(P_k)|$ and $|\Pi(s_k, s_i)| \leq 2$. P_k can be used to show that the bound holds,

Table 1 The bound of $|\Pi_{return}|$ in P_i $(i = 1, 2, \ldots, 20$

P_i	s_i	e_i	$N(P_i)$	$\frac{1}{4}N(P_i) - 1$	$\|\Pi_{return}(P_i)\|$
P_3	c_3	c_e	$N(P_c) + 2$	≥ 2	1
P_6	c_6	c_{h2}	$N(P_c) + 4$	≥ 3	2
P_7	c_7	c_{h2}	$N(P_c) + 4$	≥ 3	3
P_8	c_8	c_{h1}	$N(P_c) + 4$	≥ 3	2
P_9	c_9	c_{h1}	$N(P_c) + 4$	≥ 3	3
P_{12}	c_{12}	c_{h1}	$N(P_c) + 6$	≥ 3	2
P_{13}	c_{13}	c_{h1}	$N(P_c) + 6$	≥ 3	3
P_{16}	c_{16}	c_{h1}	$N(P_c) + 8$	≥ 4	2
P_{17}	c_{17}	c_{h2}	$N(P_c) + 8$	≥ 4	3
P_{18}	c_{18}	c_{h2}	$N(P_c) + 8$	≥ 4	2
P_{19}	c_{19}	c_e	$N(P_c) + 8$	≥ 4	3
P_{20}	c_{20}	c_e	$N(P_c) + 8$	≥ 4	4

$$\begin{aligned}
|\Pi_{return}(P_i)| &= |\Pi(e_i, s_i)| \\
&\leq |\Pi(e_i, s_k)| + |\Pi(s_k, s_i)| \\
&\leq |\Pi_{return}(P_k)| + |\Pi(s_k, s_i)| \\
&\leq \frac{1}{4}N(P_k) - 1 + 2 \\
&= \frac{1}{4}(N(P_k) + 8) - 1 \\
&\leq \frac{1}{4}N(P_i) - 1.
\end{aligned}$$

See P_{48} in Fig. 6c for an example. P_{20} can be found to prove that the bound holds. Analogously, we can add three or more layers of cells around P_c and prove that the bound holds for the achieved polygons.

In the third case, we consider P with split cells. It can be seen as a combination of extended polygons considered in the second case. See Fig. 5 for an example. Let P_1 denote the extended polygon which contains the last visited cell of P. As discussed above, we have

$$\begin{aligned}
|\Pi_{return}(P)| &= |\Pi_{return}(P_1)| \\
&\leq \frac{1}{4}N(P_1) - 1 \\
&< \frac{1}{4}N(P) - 1.
\end{aligned}$$

Thus, the bound holds for this case.

Altogether, the proof is complete. □

Theorem 1 *Assume that P is a simple grid polygon with N boundary edges, M basic cells and without any narrow passages. The length of the exploration tour achieved* *by our strategy can be bounded by*

$$|\Pi(P)| \leq M(P) + \frac{1}{4}N(P) - 2.$$

Proof Each strategy requires no less than M steps to explore P. By induction, we prove that $excess(P) \leq \frac{1}{4}N(P) - 2$.

First of all, we consider the situation that P has no split cells as the base of the induction. For visiting each cell of P, $M-1$ steps should be used at least. By Lemma 4, $|\Pi_{return}(P)|$ can be bounded by $\frac{1}{4}N(P)-1$. Thus, we have $excess(P) \leq \frac{1}{4}N(P)-2$.

Next, we consider the situation that P splits into several subgraphs by the split cells. Without loss of generality, assume that c_1 is the first split cell identified by the robot. After c_1 has been identified, the unexplored part of P splits into two subgraphs T_1 and T_2. Two extended polygons P_1 and P_2 can be defined as above with an overlap square Q around c_1. By our strategy, the robot visits $T_2 \cup c_1$ and T_1 in turn and returns to s. From Lemma 2, we know that

$$excess(P) \leq excess(P_1) + excess(T_2 \cup c_1) + 1.$$

By induction hypothesis, we have

$$excess(P) \leq \frac{1}{4}N(P_1) - 2 + \frac{1}{4}N(T_2 \cup c_1) - 2 + 1.$$

With Lemma 1 and Lemma 3, we have

$$\begin{aligned} excess(P) &\leq \frac{1}{4}N(P_1) - 2 + \frac{1}{4}(N(P_2) - 8i) - 2 + 1 \\ &= \frac{1}{4}(N(P_1) + N(P_2)) - 2i - 3 \\ &\leq \frac{1}{4}(N(P) + 4(2i+1)) - 2i - 3 \\ &= \frac{1}{4}N(P) - 2. \end{aligned}$$

Thus, the proof is complete. □

Lemma 5 *Icking et al. (2005) Assume that P is a simple grid polygon with N boundary edges and M basic cells. It has no narrow passages or split cells in the first layer. The relation between N and M satisfies*

$$N(P) \leq \frac{2}{3}M(P) + 6.$$

Theorem 2 *The competitive ratio of our strategy is 7/6.*

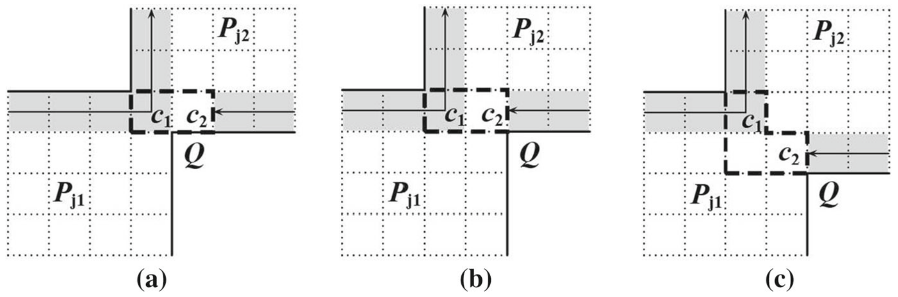

Fig. 7 Three cases of the split cell detected in the first layer. The colored cells are the ones that have been explored

Proof Since narrow passages can be explored optimally, we cut away all of them from P. Then P divides into a list of subgraphs P_j, $j = 1, 2, \ldots$. We can analyze each P_j separately. This will not affect the competitive factor of P.

We show the competitive factor of P_j by induction. Firstly, assume that the first layer of P_j contains no split cell, and we consider this situation as the base of the induction. By applying Lemma 5 and Theorem 1, we have

$$\begin{aligned}\left|\Pi(P_j)\right| &\leq M(P_j) + \frac{1}{4}N(P_j) - 2\\ &\leq M(P_j) + \frac{1}{4}(\frac{2}{3}M(P_j) + 6) - 2\\ &= \frac{7}{6}M(P_j) - \frac{1}{2}.\end{aligned}$$

Secondly, we consider the situation that a split cell c_2 is detected in the first layer of P_j. Three cases may occur, see Fig. 7 for an example. Before the first split cell c_2 is detected, the robot discovers a cell c_1 which touches c_2 in the first layer. Note that the robot will not recognize c_1 as the first split cell, because it has not enough information (known cells) of P_j when c_1 is discovered. We write Q to denote the smallest connected component which contains c_1 and c_2.

As discussed in Theorem 1, P_j divides into two polygons P_{j1} and P_{j2}, with an overlap square Q. The cells of Q will be counted twice if we analyze P_{j1} and P_{j2} separately. Thus, we have $\left|\Pi(P_j)\right| = \left|\Pi(P_{j1})\right| + \left|\Pi(P_{j2})\right| - M(Q)$ and $M(P_j) = M(P_{j1}) + M(P_{j2}) - M(Q)$. Further, By induction hypothesis, we have

$$\begin{aligned}\left|\Pi(P_j)\right| &= \left|\Pi(P_{j1})\right| + \left|\Pi(P_{j2})\right| - M(Q)\\ &\leq \frac{7}{6}M(P_{j1}) - \frac{1}{2} + \frac{7}{6}M(P_{j2}) - \frac{1}{2} - M(Q)\\ &= \frac{7}{6}M(P_j) + \frac{1}{6}M(Q) - 1\\ &\leq \frac{7}{6}M(P_j) - \frac{1}{2}\end{aligned}$$

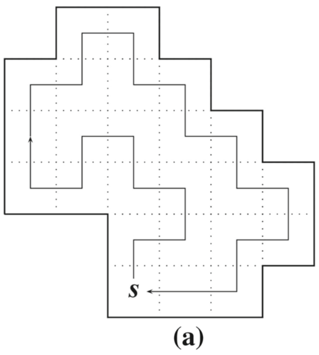

Fig. 8 A worst case instance. **a** The shortest exploration tour. **b** The exploration tour walked by our strategy

Since the length of the optimal exploration tour is no less than M steps, the proof is complete. □

Figure 8 shows an instance of the worst case. The exploration tour of our strategy needs 28 steps to explore this grid polygon, whereas the shortest exploration tour needs only 24 steps. Thus, the competitive ratio of 7/6 is achieved.

5 Conclusions

In this paper, we considered the problem of exploring an unknown simple grid polygon. We proposed a 7/6-competitive strategy which matches the lower bound. Thus improves upon the previously known 4/3-competitive strategy.

We conclude this study with two open problems as follows. First, it would be interesting to consider the situation that the grid polygon contains holes. Second, if we use two or more robots to complete this task, can we achieve more efficient strategy? We leave this as a topic for future work.

References

Almeida JPLSD, Nakashima RT, Neves-Jr F, Arruda LVRD (2019) Bio-inspired on-line path planner for cooperative exploration of unknown environment by a multi-robot system. Robot Auton Syst 112:32–48

Chalopin J, Das S, Disser Y, Mihalák M, Widmayer P (2015) Mapping simple polygons: The power of telling convex from reflex. ACM Trans Algorithms 11(4):33:1–33:16

Deng X, Kameda T, Papadimitriou C (1991) How to learn an unknown environment. In: Proceedings 32nd annual symposium of foundations of computer science, pp. 298–303

Dereniowski D, Disser Y, Kosowski A, Pająk D, Uznański P (2015) Fast collaborative graph exploration. Inf Comput 243:37–49

Doi K, Yamauchi Y, Kijima S, Yamashita M (2018) Exploration of finite 2d square grid by a metamorphic robotic system. In: Izumi T, Kuznetsov P (eds.) Stabilization, Safety, and Security of Distributed Systems, Lecture Notes in Computer Science, vol. 11201, pp. 96–110. Springer, Berlin

Foerster KT, Wattenhofer R (2016) Lower and upper competitive bounds for online directed graph exploration. Theoret Comput Sci 655:15–29

Gabriely Y, Rimon E (2003) Competitive on-line coverage of grid environments by a mobile robot. Comput Geom 24(3):197–224

Ghosh SK, Klein R (2010) Online algorithms for searching and exploration in the plane. Comput Sci Rev 4(4):189–201

Hagius R, Icking C, Langetepe E (2004) Lower bounds for the polygon exploration problems. In: 20th European Workshop Computer Geometry, pp. 135–138

Hoffmann F, Icking C, Klein R, Kriegel K (2002) The polygon exploration problem. SIAM J Comput 31(2):577–600

Icking C, Kamphans T, Klein R, Langetepe E (2002) On the competitive complexity of navigation tasks. In: Hager GD, Christensen HI, Bunke H, Klein R (eds) Sensor Based Intelligent Robots, vol 2238. Lecture Notes in Computer Science. pp 245–258. Springer, Berlin

Icking C, Kamphans T, Klein R, Langetepe E (2005) Exploring simple grid polygons. In: Wang L (ed) Computing and Combinatorics, vol 3595. Lecture Notes in Computer Science. pp 524–533. Springer, Berlin

Keshavarz-Kohjerdi F, Bagheri A (2016) Hamiltonian paths in l-shaped grid graphs. Theoret Comput Sci 621:37–56

Megow N, Mehlhorn K, Schweitzer P (2012) Online graph exploration: New results on old and new algorithms. Theoret Comput Sci 463:62–72

Ortolf C, Schindelhauer C (2012) Online multi-robot exploration of grid graphs with rectangular obstacles. In: Proceedings of the Twenty-fourth Annual ACM Symposium on Parallelism in Algorithms and Architectures, SPAA ’12, pp 27–36. ACM, New York, USA

Strom DP, Bogoslavskyi I, Stachniss C (2017) Robust exploration and homing for autonomous robots. Robot Auton Syst 90:125–135

Tan X, Wei Q (2015) An improved on-line strategy for exploring unknown polygons. Combinatorial Optimization and Applications, vol 9486. Lecture Notes in Computer Science. pp 163–177. Springer, Berlin