Exploring Simple Grid Polygons

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

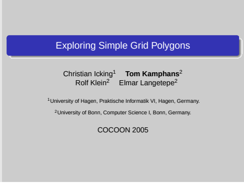
[1]University of Hagen, Praktische Informatik VI, Hagen, Germany.

[2]University of Bonn, Computer Science I, Bonn, Germany.

COCOON 2005

2005-08-29

Exploring Simple Grid Polygons



Exploring Simple Grid Polygons

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

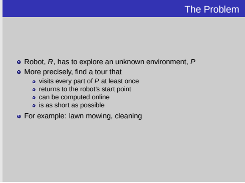
[1]University of Hagen, Praktische Informatik VI, Hagen, Germany.

[2]University of Bonn, Computer Science I, Bonn, Germany.

COCOON 2005

The Problem

- Robot, R, has to explore an unknown environment, P
- More precisely, find a tour that
  - visits every part of P at least once
  - returns to the robot's start point
  - can be computed online
  - is as short as possible
- For example: lawn mowing, cleaning

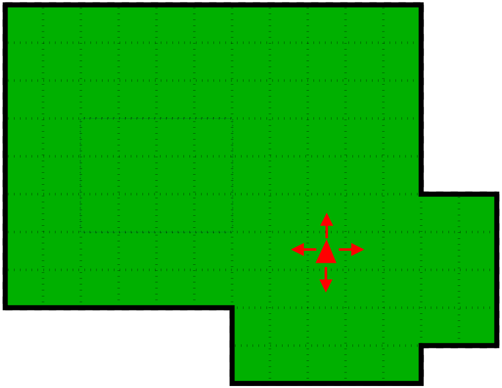


- Robot, *R*, has to explore an unknown environment, *P*
- More precisely, find a tour that
  - visits every part of *P* at least once
  - returns to the robot's start point
  - can be computed online
  - is as short as possible
- For example: lawn mowing, cleaning

We consider the problem of a robot that has to explore an environment that is unknown to the robot.

More precisely, we want to find a path that visits every part of the environment at least once and returns to start point. The environment is unknown to the robot, so we want to compute our path online. We use the length of the robot's path as quality measure; thus, we want to keep the path as short as possible.

Environment and Robot

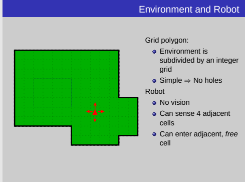


Grid polygon:

- Environment is subdivided by an integer grid
- Simple ⇒ No holes

Robot

- No vision
- Can sense 4 adjacent cells
- Can enter adjacent, *free* cell



Environment and Robot

Grid polygon:
- Environment is subdivided by an integer grid
- Simple ⇒ No holes

Robot
- No vision
- Can sense 4 adjacent cells
- Can enter adjacent, *free* cell

We deal with a special kind of environments we call *grid-polygons*, that is, we assume, that the environment is subdivided by an integer grid.
Imagine, we want to mow a grass like this, then we can subdivide the environment according to the size of the tool.
Here, we restrict ourselves to *simple* polygons, that is, there are no obstacles inside the polygon.

We assume, that the robot can sense only the four cells that are adjacent to the current position. The robot can move from one cell to an adjacent cell that is part of the polygon.

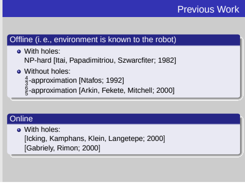
Previous Work

Offline (i. e., environment is known to the robot)

- With holes:
  NP-hard [Itai, Papadimitriou, Szwarcfiter; 1982]
- Without holes:
  $\frac{4}{3}$-approximation [Ntafos; 1992]
  $\frac{6}{5}$-approximation [Arkin, Fekete, Mitchell; 2000]

Online

- With holes:
  [Icking, Kamphans, Klein, Langetepe; 2000]
  [Gabriely, Rimon; 2000]



Previous Work

Offline (i. e., environment is known to the robot)

- With holes:
  NP-hard [Itai, Papadimitriou, Szwarcfiter; 1982]
- Without holes:
  $\frac{4}{3}$-approximation [Ntafos; 1992]
  $\frac{6}{5}$-approximation [Arkin, Fekete, Mitchell; 2000]

Online

- With holes:
  [Icking, Kamphans, Klein, Langetepe; 2000]
  [Gabriely, Rimon; 2000]

It is known, that the offline case where the environment is known to the robot is NP-hard for polygons with holes. For polygons without holes there are some approximations.

The online case for polygons with holes was considered by ourselves and independently by Gabriely and Rimon.

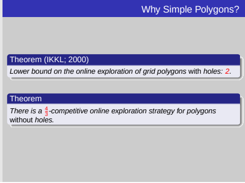
Why Simple Polygons?

Theorem (IKKL; 2000)

Lower bound on the online exploration of grid polygons with *holes: 2.*

Theorem

There is a $\frac{4}{3}$-competitive online exploration strategy for polygons without *holes.*



Theorem (IKKL; 2000)

Lower bound on the online exploration of grid polygons with *holes: 2.*

Theorem

There is a $\frac{4}{3}$-competitive online exploration strategy for polygons without *holes.*

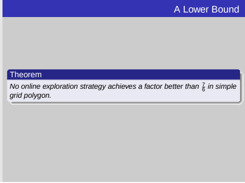
So, the case of general grid polygons is already solved, so why are we interested in simple grid polygons? Not surprisingly, we can explore simple polygons more efficiently. More precisely, we have a lower bound of 2 for general grid polygons.

On the other hand, we will see that we can achieve a competitive factor of 4/3 in SIMPLE grid polygons.

A Lower Bound

Theorem

No online exploration strategy achieves a factor better than $\frac{7}{6}$ in simple grid polygon.

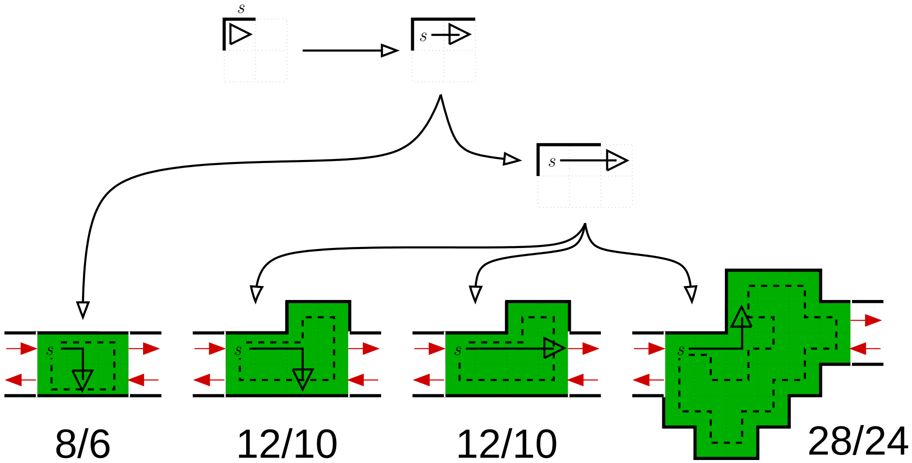


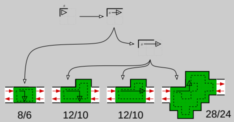
Theorem

No online exploration strategy achieves a factor better than $\frac{7}{6}$ in simple grid polygon.

But first, let me give you a lower bound on our problem: It can be shown that no online exploration strategy can achieve a factor better than 7/6.

Proof: Lower Bound

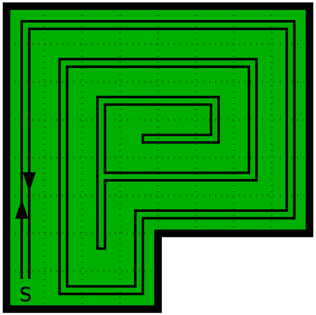




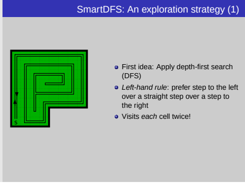
We assume, that the robot starts in a corner of the polygon and moves one step to the east. If the robot walks south, we can used a mirrored construction.
For the second step, the robot has two possibilities: First, it may walk to the south, leaving the polygons boundary. Second, it could continue with a step to the east.
In the first case, we close the polygon like this. Now, whatever the robot does, it needs at least 8 steps, whereas the optimal solution needs only 6 steps.
In the second case, the robot has three possiblities: It can move S/E/N.
In the first two cases, we close the polygon like this, and get a l.b. of $\frac{12}{10}$.
In the more interesting case, the robot continues to follow the wall and we close the polygon is this way. Now, the robot needs at least 28 steps, the optimal strategy needs 24 steps.

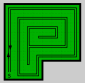
These blocks have limited size. To get a general lower bound, we have to construct polygons of arbitrary size. We can do this by repeating this construction: As soon as the robot leaves one block, it enters the start cell of the next block and the 'game' starts again.

SmartDFS: An exploration strategy (1)



- First idea: Apply depth-first search (DFS)
- *Left-hand rule*: prefer step to the left over a straight step over a step to the right
- Visits *each* cell twice!

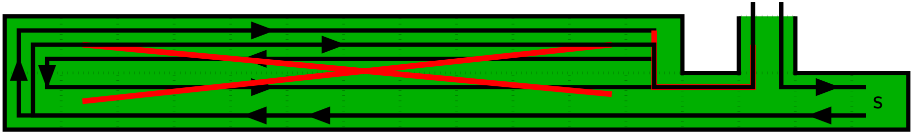




- First idea: Apply depth-first search (DFS)
- *Left-hand rule*: prefer step to the left over a straight step over a step to the right
- Visits *each* cell twice!

A first idea for the exploration is, to use a simple depth-first search. We use the *left-hand rule*, that is, we always keep the polygons boundary and the visited cells on the left side of the robot. Of course, DFS visits each cell twice.

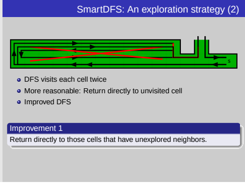
SmartDFS: An exploration strategy (2)



- DFS visits each cell twice
- More reasonable: Return directly to unvisited cell
- Improved DFS

Improvement 1

Return directly to those cells that have unexplored neighbors.



SmartDFS: An exploration strategy (2)

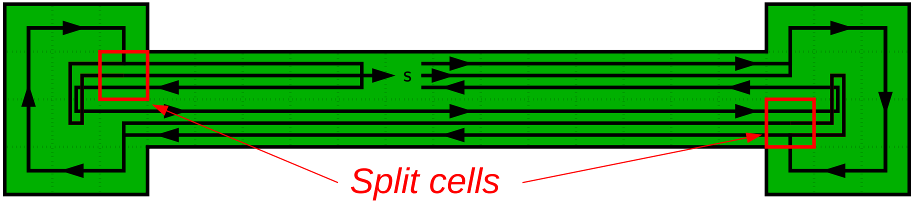
- DFS visits each cell twice
- More reasonable: Return directly to unvisited cell
- Improved DFS

Improvement 1
Return directly to those cells that have unexplored neighbors.

Visiting each cell twice is not very efficient, we can do better. In this example, DFS visits each cell in the long corridor twice. Of course, it is more reasonable to omit the second visit of the long corridor and walk directly to the unexplored cell $\otimes$ so we get a path like this.

So the first improvement to DFS is to return directly to those cells that have unexplored neighbors.

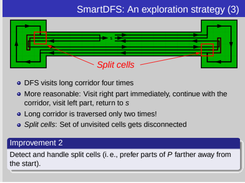
SmartDFS: An exploration strategy (3)



- DFS visits long corridor four times
- More reasonable: Visit right part immediately, continue with the corridor, visit left part, return to *s*
- Long corridor is traversed only two times!
- *Split cells*: Set of unvisited cells gets disconnected

Improvement 2

Detect and handle split cells (i. e., prefer parts of *P* farther away from the start).



SmartDFS: An exploration strategy (3)

- DFS visits long corridor four times
- More reasonable: Visit right part immediately, continue with the corridor, visit left part, return to *s*
- Long corridor is traversed only two times!
- *Split cells*: Set of unvisited cells gets disconnected

Improvement 2

Detect and handle split cells (i. e., prefer parts of *P* farther away from the start).

But we still can do better. In this case, even the improved version of DFS visits the long corridor in the middle four times.
It is more reasonable to explore the polygon up to this cell. Now, we diverge from the left-hand rule and explore the right part first. Then, we continue with the corridor, visit the left part and return to the start. Now, we visit the corridor in the middle only two times!
The cells on which we diverge from the left-hand rule have the property, that the unvisited cells split in two components after the cell is visited, we call cells like this *split cells*.

Thus, the second improvement to DFS is to detect split and handle split cells. Basically, we want to deal with components that are farther away from the start first.

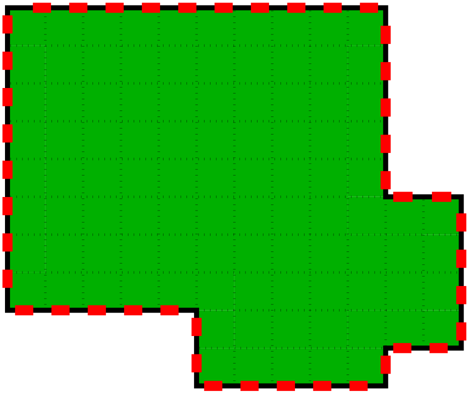
Java Applet

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

Java Applet

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

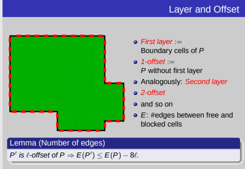
Layer and Offset



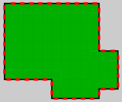
- *First layer* := Boundary cells of P
- *1-offset* := P without first layer
- Analogously: *Second layer*
- *2-offset*
- and so on
- E: #edges between free and blocked cells

Lemma (Number of edges)

P' is ℓ-offset of P $\Rightarrow E(P') \leq E(P) - 8\ell$.



Layer and Offset



- *First layer* :=
  Boundary cells of *P*
- *1-offset* :=
  P without first layer
- Analogously: *Second layer*
- *2-offset*
- and so on
- *E*: #edges between free and blocked cells

Lemma (Number of edges)

P' is ℓ-offset of $P \Rightarrow E(P') \leq E(P) - 8\ell$.

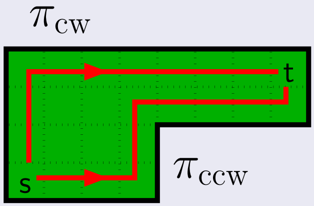
We need some definitions and lemmata for the analysis of our strategy. First, we call the boundary cells of *P* the *first layer* of *P*. *P* without its first layer is called the *1-offset* of *P*. The boundary cells of the 1-offset are called the second layer of *P*, *P* without its first and second layer is called the 2-offset and so on.
Now, *E* be the number of edges between a free cell and a blocked cell. In this polygon, for example, we count the edges shown in red.
It is easy to see that the ℓ-offset of a polygon *P* has at least 8ℓ edges less than *P*.

Shortest Paths Lengths

Lemma (Shortest Path)

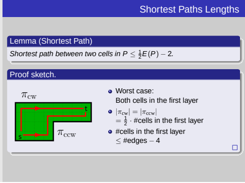
Shortest path between two cells in $P \leq \frac{1}{2}E(P) - 2$.

Proof sketch.



- Worst case:
  Both cells in the first layer
- $|\pi_{\mathrm{cw}}| = |\pi_{\mathrm{ccw}}|$
  $= \frac{1}{2} \cdot$ #cells in the first layer
- #cells in the first layer
  $\leq$ #edges $- 4$

$\square$

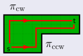


Shortest Paths Lengths

Lemma (Shortest Path)

Shortest path between two cells in $P \leq \frac{1}{2}E(P) - 2$.

Proof sketch.



- Worst case: Both cells in the first layer
- $|\pi_{\mathrm{cw}}| = |\pi_{\mathrm{ccw}}| = \frac{1}{2} \cdot$ #cells in the first layer
- #cells in the first layer $\leq$ #edges $- 4$

□

Next, we can bound the length of a shortest path inside a grid polygon by half the number of edges -2.
To show this, we assume the worst case, that is, both cells are located in the first layer of P. Now, observe the paths in first layer clockwise and counter-clockwise from s to t. In the worst case, both path have the same length: half the number of cells in the first layer. It is easy to see that the number of cells in the first layer is at least 4 less than the number of edges in the first layer, so we get our statement.

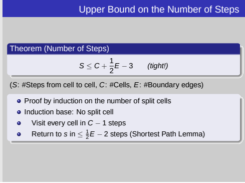
Upper Bound on the Number of Steps

Theorem (Number of Steps)

$$S \leq C + \frac{1}{2}E - 3 \qquad \textit{(tight!)}$$

(S: #Steps from cell to cell, C: #Cells, E: #Boundary edges)

- Proof by induction on the number of split cells
- Induction base: No split cell
- Visit every cell in $C - 1$ steps
- Return to s in $\leq \frac{1}{2}E - 2$ steps (Shortest Path Lemma)



Theorem (Number of Steps)

$$S \leq C + \frac{1}{2}E - 3 \qquad \textit{(tight!)}$$

(S: #Steps from cell to cell, C: #Cells, E: #Boundary edges)

- Proof by induction on the number of split cells
- Induction base: No split cell
  - Visit every cell in $C - 1$ steps
  - Return to s in $\leq \frac{1}{2}E - 2$ steps (Shortest Path Lemma)

Now, we are able to give a first performance result for our exploration strategy: The number of steps from cell to cell is bound by the number of cells plus half the number of edges minus three. And this bound is tight!

We can show this by an induction on the number of split cells. In the induction base we have no split cell, thus, we need $C - 1$ steps to explore the whole polygon and — using our lemma — at most $\frac{1}{2}E - 2$ for the path back to the start.

Competitivity

Theorem (Competitivity)

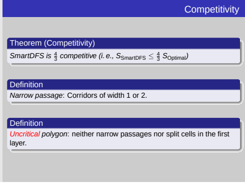
SmartDFS is $\frac{4}{3}$ competitive (i. e., $S_{\text{SmartDFS}} \leq \frac{4}{3}\, S_{\text{Optimal}}$)

Definition

Narrow passage: Corridors of width 1 or 2.

Definition

Uncritical polygon: neither narrow passages nor split cells in the first layer.



Competitivity

Theorem (Competitivity)
SmartDFS is $\frac{4}{3}$ competitive (i. e., $S_{\text{SmartDFS}} \leq \frac{4}{3}\, S_{\text{Optimal}}$)

Definition
Narrow passage: Corridors of width 1 or 2.

Definition
Uncritical polygon: neither narrow passages nor split cells in the first layer.

The second performance result is, that our exploration strategy, SmartDFS, is competitive with a factor of $\frac{4}{3}$, that is, the path generated by our strategy is never longer than $\frac{4}{3}$ times the optimal solution.
In the following, corridors of width 1 or 2 play an important rule, so we refer to them as *narrow passages*. More precisely, a cell belongs to a narrow passage, if we can remove this cell without changing the layer number of any other cell.

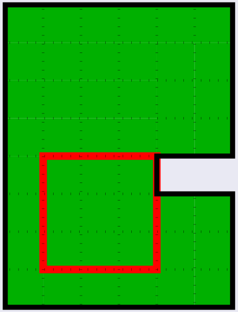
We call polygons, that have neigher narrow passages nor split in the first layer *uncritical* polygons.

Competitivity (2)

Lemma (Edges in uncritical polygons)

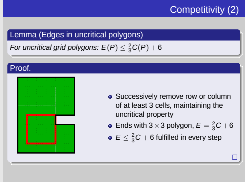
For uncritical grid polygons: $E(P) \leq \frac{2}{3}C(P) + 6$

Proof.



- Successively remove row or column of at least 3 cells, maintaining the uncritical property
- Ends with 3×3 polygon, $E = \frac{2}{3}C + 6$
- $E \leq \frac{2}{3}C + 6$ fulfilled in every step

$\square$

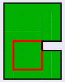


Competitivity (2)

Lemma (Edges in uncritical polygons)

For uncritical grid polygons: $E(P) \leq \frac{2}{3}C(P) + 6$

Proof.



- Successively remove row or column of at least 3 cells, maintaining the uncritical property
- Ends with 3×3 polygon, $E = \frac{2}{3}C + 6$
- $E \leq \frac{2}{3}C + 6$ fulfilled in every step

□

For the special class of uncritical polygons, we can proof two lemmata. First, we can bound the number of edges in such a polygon by $\frac{2}{3}$ times the number of cells plus 6.

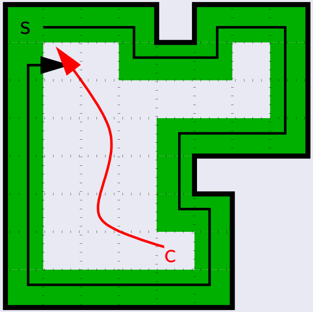
To proof this, we remove successively a row or column of at least three cells, keeping the property that the polygon is uncritical. This decomposition ends with a three times three block of cells that fulfills this equation. Now, if we reverse our decomposition process, we add at most 2 edges and at least three cells in every step; thus, this inequation is fulfulled in every step.

Competitivity (3)

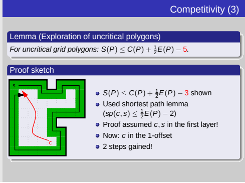
Lemma (Exploration of uncritical polygons)

For uncritical grid polygons: $S(P) \leq C(P) + \frac{1}{2}E(P) - 5$.

Proof sketch



- $S(P) \leq C(P) + \frac{1}{2}E(P) - 3$ shown
- Used shortest path lemma ($sp(c, s) \leq \frac{1}{2}E(P) - 2$)
- Proof assumed c, s in the first layer!
- Now: c in the 1-offset
- 2 steps gained!

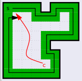


Competitivity (3)

Lemma (Exploration of uncritical polygons)

For uncritical grid polygons: $S(P) \leq C(P) + \frac{1}{2}E(P) - 5$.

Proof sketch



- $S(P) \leq C(P) + \frac{1}{2}E(P) - 3$ shown
- Used shortest path lemma ($sp(c, s) \leq \frac{1}{2}E(P) - 2$)
- Proof assumed c, s in the first layer!
- Now: c in the 1-offset
- 2 steps gained!

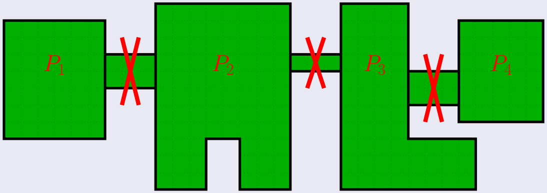
Second, we can show that we can explore uncritical polygons better than general simple polygons.
We already showed this upper bound for arbitrary simple polygons. In the induction base, we used the shortest path lemma, but to proof this lemma, we assumed that both cells are in the first layer of P.
Now, we return from a cell in the 1-offset of P to s, and so we gain two steps.

Competitivity Proof

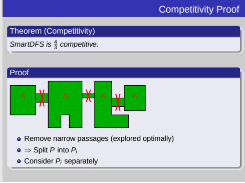
Theorem (Competitivity)

SmartDFS is $\frac{4}{3}$ competitive.

Proof



- Remove narrow passages (explored optimally)
- $\Rightarrow$ Split P into P_i
- Consider P_i separately


Competitivity Proof

Theorem (Competitivity)

SmartDFS is $\frac{4}{3}$ competitive.

Proof

- Remove narrow passages (explored optimally)
- $\Rightarrow$ Split P into P_i
- Consider P_i separately

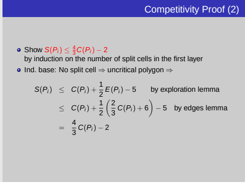
With these two lemmata, we can show our theorem. First, we remove all narrow passages, because they are explored optimally.

Thus, we split P into a sequence of polygons P_i, which can be considered separately without changing the competitive factor.

Competitivity Proof (2)

- Show $S(P_i) \leq \frac{4}{3}C(P_i) - 2$ by induction on the number of split cells in the first layer
- Ind. base: No split cell $\Rightarrow$ uncritical polygon $\Rightarrow$

$$\begin{aligned} S(P_i) &\leq C(P_i) + \frac{1}{2}E(P_i) - 5 && \text{by exploration lemma} \\ &\leq C(P_i) + \frac{1}{2}\left(\frac{2}{3}C(P_i) + 6\right) - 5 && \text{by edges lemma} \\ &= \frac{4}{3}C(P_i) - 2 \end{aligned}$$



Competitivity Proof (2)

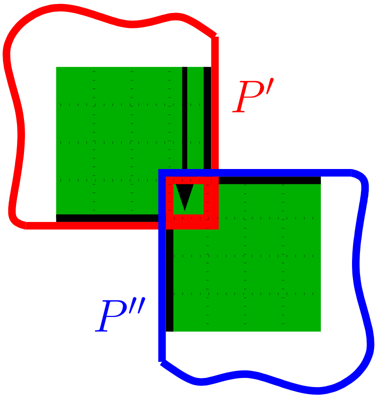
- Show $S(P_i) \leq \frac{4}{3}C(P_i) - 2$ by induction on the number of split cells in the first layer
- Ind. base: No split cell $\Rightarrow$ uncritical polygon $\Rightarrow$

$$\begin{aligned} S(P_i) &\leq C(P_i) + \frac{1}{2}E(P_i) - 5 && \text{by exploration lemma} \\ &\leq C(P_i) + \frac{1}{2}\left(\frac{2}{3}C(P_i) + 6\right) - 5 && \text{by edges lemma} \\ &= \frac{4}{3}C(P_i) - 2 \end{aligned}$$

Now, we show this inequation by an induction on the number of split cells in the first layer of P_i. If there is no split cell, then our polygon is uncritical and our result follows from the two lemmatas on uncritical polygons.

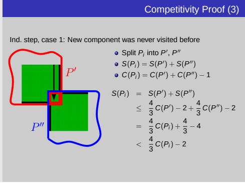
Competitivity Proof (3)

Ind. step, case 1: New component was never visited before



- Split P_i into P', P''
- $S(P_i) = S(P') + S(P'')$
- $C(P_i) = C(P') + C(P'') - 1$

$$\begin{aligned} S(P_i) &= S(P') + S(P'') \\ &\leq \frac{4}{3}C(P') - 2 + \frac{4}{3}C(P'') - 2 \\ &= \frac{4}{3}C(P_i) + \frac{4}{3} - 4 \\ &< \frac{4}{3}C(P_i) - 2 \end{aligned}$$



Competitivity Proof (3)

Ind. step, case 1: New component was never visited before

- Split P_i into P', P''
- $S(P_i) = S(P') + S(P'')$
- $C(P_i) = C(P') + C(P'') - 1$

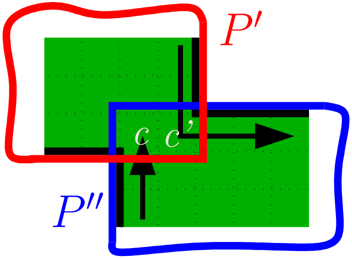
$$\begin{aligned} S(P_i) &= S(P') + S(P'') \\ &\leq \frac{4}{3}C(P') - 2 + \frac{4}{3}C(P'') - 2 \\ &= \frac{4}{3}C(P_i) + \frac{4}{3} - 4 \\ &< \frac{4}{3}C(P_i) - 2 \end{aligned}$$

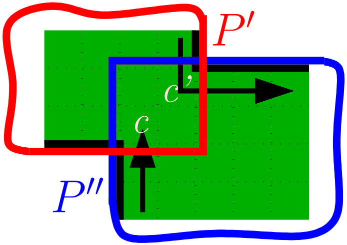
If there is a split cell in the first layer, we have to consider two cases. In the first case, the new component was never visited before. We split our polygon into P' and P''. It is easy to see that these two equations for the number of cells and steps hold.

Now, we just use the induction hypothesis on the number of steps in P' and P'' and with some simplifications we obtain our result.

Competitivity Proof (4)

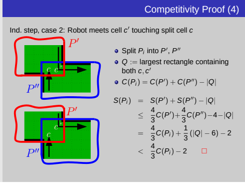
Ind. step, case 2: Robot meets cell c' touching split cell c





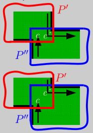
- Split P_i into P', P''
- $Q :=$ largest rectangle containing both c, c'
- $C(P_i) = C(P') + C(P'') - |Q|$

$$\begin{aligned} S(P_i) &= S(P') + S(P'') - |Q| \\ &\le \frac{4}{3}C(P') + \frac{4}{3}C(P'') - 4 - |Q| \\ &= \frac{4}{3}C(P_i) + \frac{1}{3}(|Q| - 6) - 2 \\ &< \frac{4}{3}C(P_i) - 2 \qquad \square \end{aligned}$$



Competitivity Proof (4)

Ind. step, case 2: Robot meets cell c' touching split cell c



- Split P_i into P', P''
- $Q :=$ largest rectangle containing both c, c'
- $C(P_i) = C(P') + C(P'') - |Q|$

$$\begin{aligned} S(P_i) &= S(P') + S(P'') - |Q| \\ &\leq \frac{4}{3}C(P') + \frac{4}{3}C(P'') - 4 - |Q| \\ &= \frac{4}{3}C(P_i) + \frac{1}{3}(|Q| - 6) - 2 \\ &< \frac{4}{3}C(P_i) - 2 \qquad \square \end{aligned}$$

In the second case, the has surrounded the new component and meets a visited cell, c', that touches the split cell c. Again, we split P_i into P' and P''. Now, let Q be the largest rectangle that contains both c and c''.

Now, we have this equations for the number of cells and steps. Again, we apply the induction hypothesis to P' and P'' and get our result.

This finishes the proof.

Summary

Problem: Online exploration of simple grid polygons

- Lower Bound: $\frac{7}{6}$
- Exploration strategy *SmartDFS*
- $S \leq C + \frac{1}{2}E - 3$
- $\frac{4}{3}$-competitive

- ToDo: Close the gap!

Thank you!