Models and Algorithms for Online Exploration and Search

Tom Kamphans[1]

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

April 04, 2006

Outline

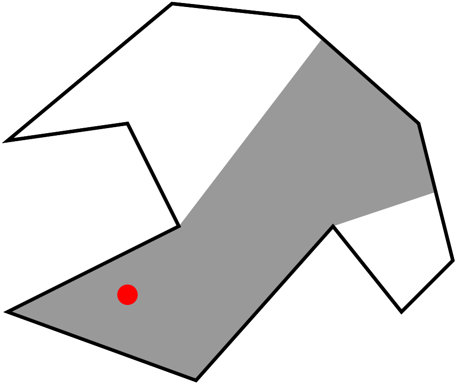
1. Introduction

2. Exploring Grid Polygons
   - Introduction
   - Simple Grid Polygons
   - Grid Polygons with Holes

3. Search

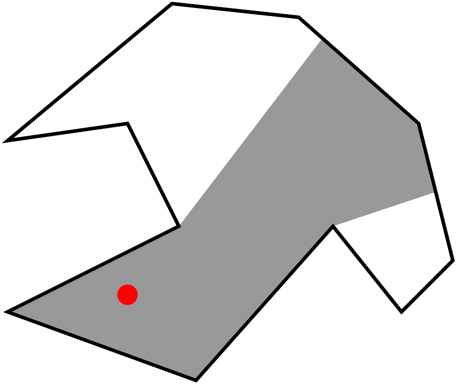
Exploration and Search

- Planning a path for an autonomous vehicle
- Exploration:
  Move around, until everything was ’seen’
- Searching:
  Move around, until target is found



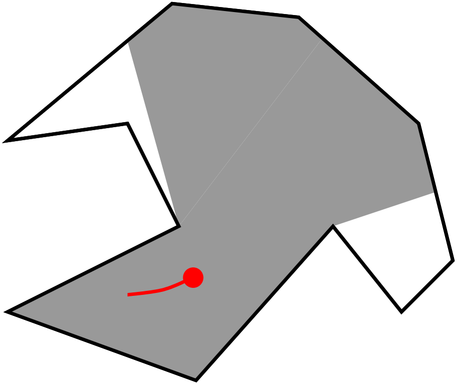
Exploration and Search

- Planning a path for an autonomous vehicle
- Exploration:
  Move around, until everything was ’seen’
- Searching:
  Move around, until target is found



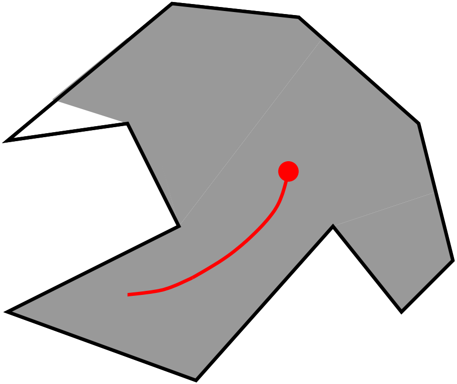
Exploration and Search

- Planning a path for an autonomous vehicle
- Exploration:
  Move around, until everything was 'seen'
- Searching:
  Move around, until target is found



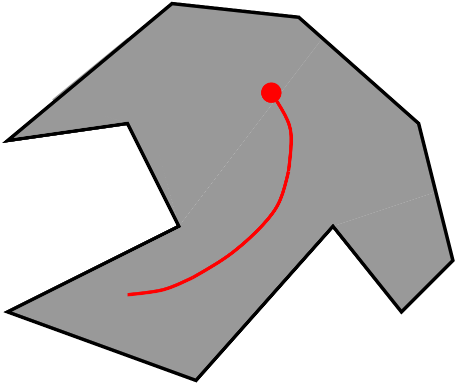
Exploration and Search

- Planning a path for an autonomous vehicle
- Exploration:
  Move around, until everything was 'seen'
- Searching:
  Move around, until target is found



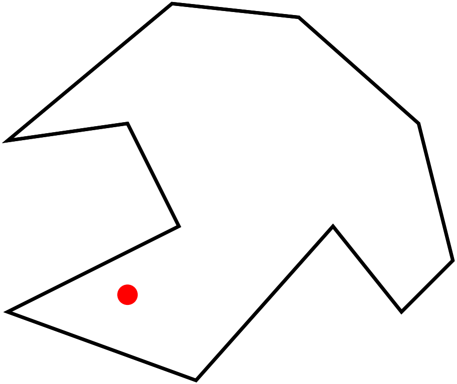
Exploration and Search

- Planning a path for an autonomous vehicle
- Exploration:
  Move around, until everything was 'seen'
- Searching:
  Move around, until target is found



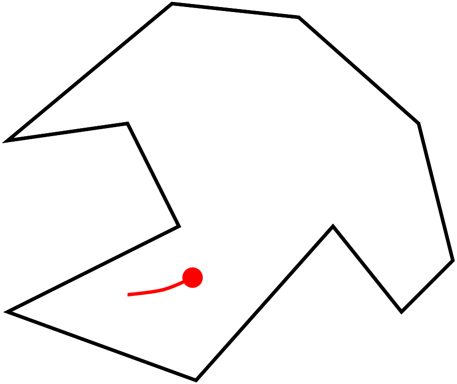
Exploration and Search

- Planning a path for an autonomous vehicle
- Exploration:
  Move around, until everything was ’seen’
- Searching:
  Move around, until target is found



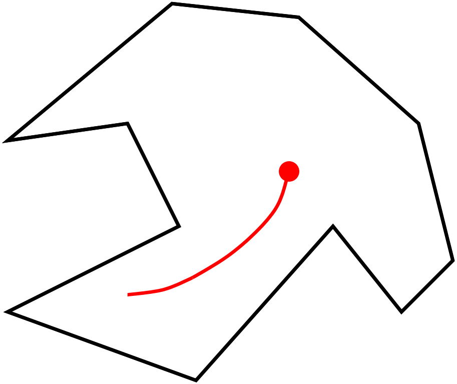
Exploration and Search

- Planning a path for an autonomous vehicle
- Exploration:
  Move around, until everything was ’seen’
- Searching:
  Move around, until target is found



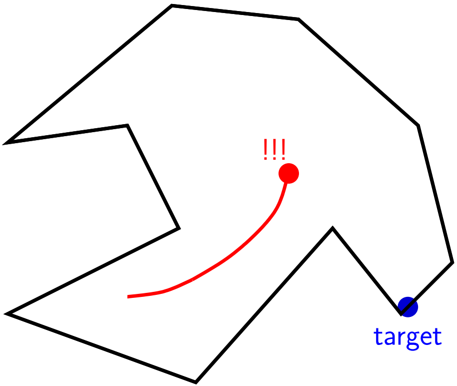
Exploration and Search

- Planning a path for an autonomous vehicle
- Exploration:
  Move around, until everything was 'seen'
- Searching:
  Move around, until target is found



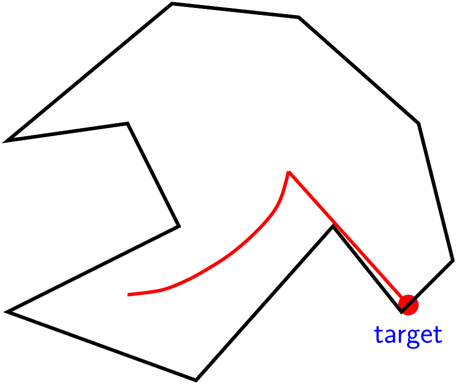
Exploration and Search

- Planning a path for an autonomous vehicle
- Exploration:
  Move around, until everything was 'seen'
- Searching:
  Move around, until target is found



Exploration and Search

- Planning a path for an autonomous vehicle
- Exploration:
  Move around, until everything was 'seen'
- Searching:
  Move around, until target is found



Models

’Real world’ ⟶ ’Computable world’

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: $^{|\mathrm{ONL}|}/_{|\mathrm{OPT}|}$
  - Other ratios (*search ratio*)

Models

’Real world’ ⟶ ’Computable world’

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: $^{|ONL|}/_{|OPT|}$
  - Other ratios (*search ratio*)

Models

'Real world' ⟶ 'Computable world'

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: $^{|\mathrm{ONL}|}/_{|\mathrm{OPT}|}$
  - Other ratios (*search ratio*)

Models

'Real world' ⟶ 'Computable world'

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: $|\mathrm{ONL}| / |\mathrm{OPT}|$
  - Other ratios (*search ratio*)

Models

'Real world' ⟶ 'Computable world'

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: $^{|\mathrm{ONL}|}/_{|\mathrm{OPT}|}$
  - Other ratios (*search ratio*)

Models

'Real world' ⟶ 'Computable world'

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: $^{|\text{ONL}|}/_{|\text{OPT}|}$
  - Other ratios (*search ratio*)

Models

’Real world’ ⟶ ’Computable world’

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: $^{|\text{ONL}|}/_{|\text{OPT}|}$
  - Other ratios (*search ratio*)

Models

’Real world’ $\longrightarrow$ ’Computable world’

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: $^{|\mathrm{ONL}|}/_{|\mathrm{OPT}|}$
  - Other ratios (*search ratio*)

Models

’Real world’ ⟶ ’Computable world’

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: $^{|\mathrm{ONL}|}/_{|\mathrm{OPT}|}$
  - Other ratios (*search ratio*)

Models

’Real world’ $\longrightarrow$ ’Computable world’

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: $|\mathrm{ONL}| / |\mathrm{OPT}|$
  - Other ratios (*search ratio*)

Models

’Real world’ ⟶ ’Computable world’

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: $|ONL| / |OPT|$
  - Other ratios (*search ratio*)

Models

’Real world’ ⟶ ’Computable world’

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: ${}^{|\mathrm{ONL}|}/_{|\mathrm{OPT}|}$
  - Other ratios (*search ratio*)

Models

'Real world' ⟶ 'Computable world'

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: $|ONL|/|OPT|$
  - Other ratios (*search ratio*)

Models

’Real world’ $\longrightarrow$ ’Computable world’

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: $|\text{ONL}| / |\text{OPT}|$
  - Other ratios (*search ratio*)

Models

'Real world' ⟶ 'Computable world'

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: $^{|ONL|}/_{|OPT|}$
  - Other ratios (*search ratio*)

Models

’Real world’ $\longrightarrow$ ’Computable world’

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: $|\text{ONL}| / |\text{OPT}|$
  - Other ratios (*search ratio*)

Models

’Real world’ ⟶ ’Computable world’

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: $^{|\mathrm{ONL}|}/_{|\mathrm{OPT}|}$
  - Other ratios (*search ratio*)

Models

’Real world’ $\longrightarrow$ ’Computable world’

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: $^{|\mathrm{ONL}|}/_{|\mathrm{OPT}|}$
  - Other ratios (*search ratio*)

Models

'Real world' $\longrightarrow$ 'Computable world'

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: ${}^{|\text{ONL}|}/_{|\text{OPT}|}$
  - Other ratios (*search ratio*)

Models

'Real world' ⟶ 'Computable world'

- Robot
  - Shape (point, circle, polygon), sensors (touch, vision), motion restrictions, computational abilities
  - Errors in sensors and motion
- Environment
  - Graph, polygon, obstacles (none/rect./polygonal/curved),
  - *Grid environments*
- Costs
  - Measure: path length, number of turns/scans
  - Dimensions of the environment
  - Competitive ratio: $^{|\mathrm{ONL}|}/_{|\mathrm{OPT}|}$
  - Other ratios (*search ratio*)

Outline

1 Introduction

2 Exploring Grid Polygons
- Introduction
- Simple Grid Polygons
- Grid Polygons with Holes

3 Search

The Problem

- Robot has to explore an unknown environment, *P*
- Find a tour in *P* 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

The Problem

- Robot has to explore an unknown environment, *P*
- Find a tour in *P* 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

The Problem

- Robot has to explore an unknown environment, P
- Find a tour in P 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

The Problem

- Robot has to explore an unknown environment, P
- Find a tour in P 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

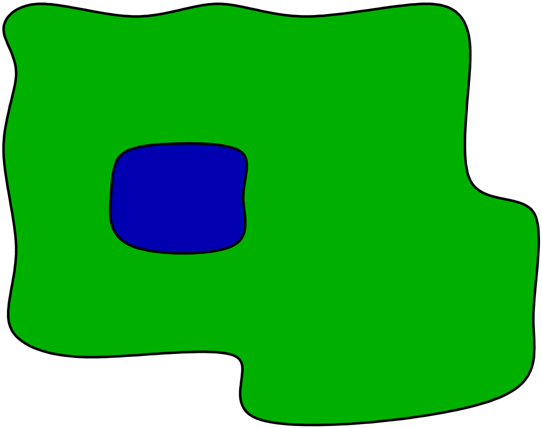
The Problem

- Robot has to explore an unknown environment, P
- Find a tour in P 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

The Problem

- Robot has to explore an unknown environment, P
- Find a tour in P 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

Environment and Robot



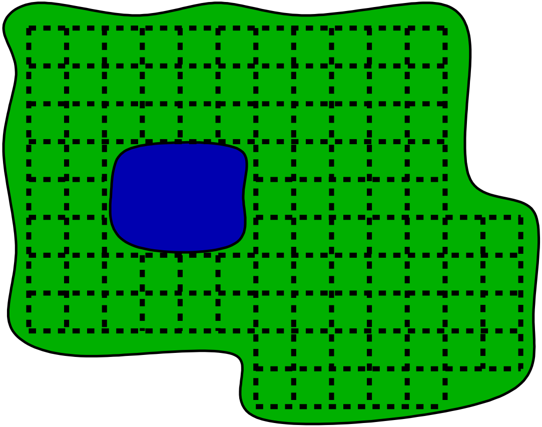
Grid polygon:

- 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



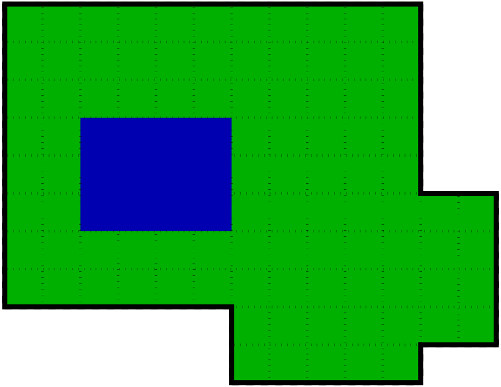
Grid polygon:

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

Robot

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

Environment and Robot



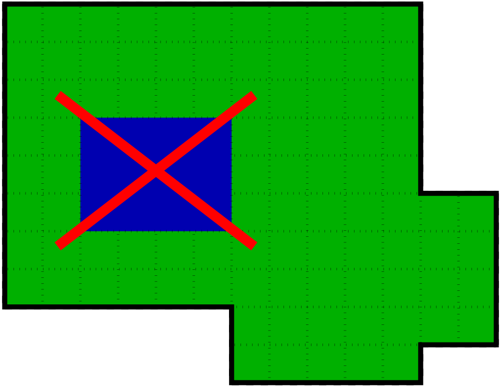
Grid polygon:

- 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



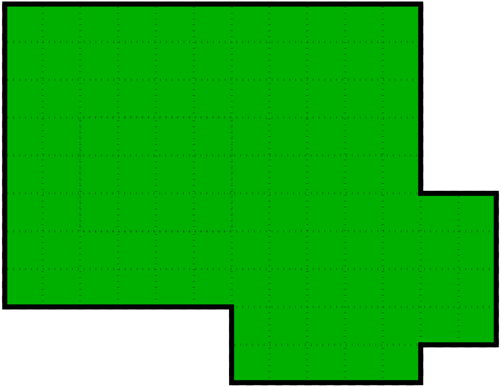
Grid polygon:

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

Robot

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

Environment and Robot



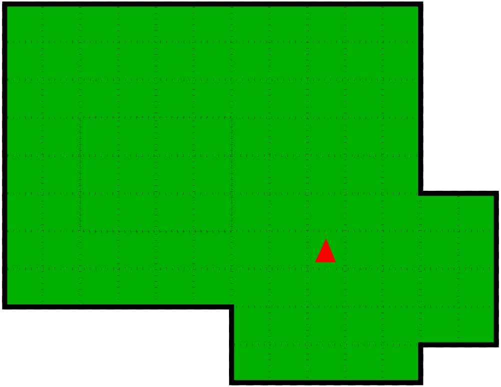
Grid polygon:

- 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



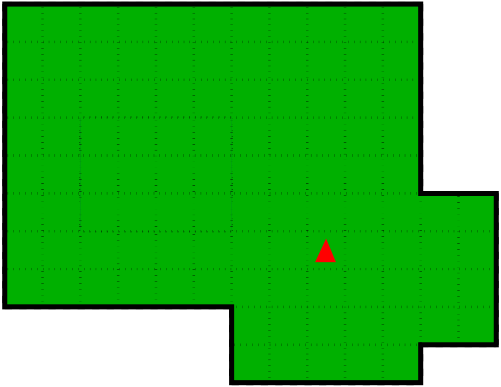
Grid polygon:

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

Robot

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

Environment and Robot



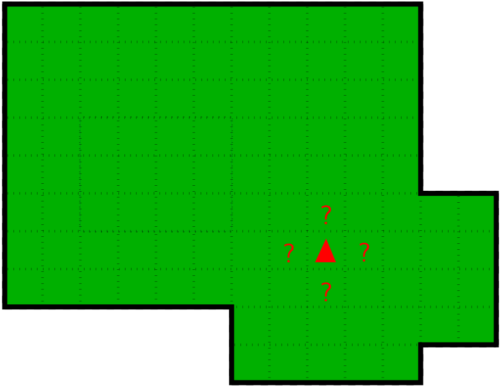
Grid polygon:

- 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



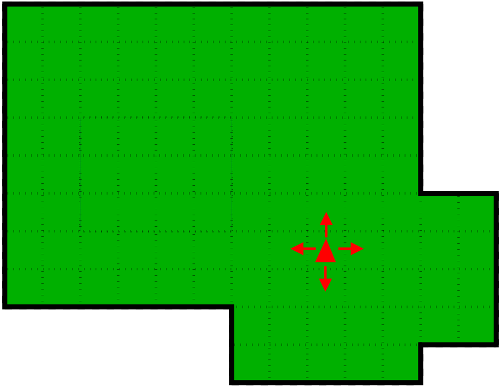
Grid polygon:

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

Robot

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

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

Previous Work

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

- With holes:
  NP-hard [Itai, Papadimitriou, Szwarcfiter; 1982]
  $\frac{53}{40}$-approximation [Arkin, Fekete, Mitchell; 2000]
- Without holes: complexity is unknown!
  $\frac{4}{3}$-approximation [Ntafos; 1992]
  $\frac{6}{5}$-approximation [Arkin, Fekete, Mitchell; 2000]

Online

- [Butler; 1998], [Gabriely, Rimon; 2000]
  [Bruckstein, Lindenbaum, Wagner; 2000]
- Survey on covering [Choset; 2001]

Previous Work

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

- With holes:
  NP-hard [Itai, Papadimitriou, Szwarcfiter; 1982]
  $\frac{53}{40}$-approximation [Arkin, Fekete, Mitchell; 2000]
- Without holes: complexity is unknown!
  $\frac{4}{3}$-approximation [Ntafos; 1992]
  $\frac{6}{5}$-approximation [Arkin, Fekete, Mitchell; 2000]

Online

- [Butler; 1998], [Gabriely, Rimon; 2000]
  [Bruckstein, Lindenbaum, Wagner; 2000]
- Survey on covering [Choset; 2001]

Previous Work

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

- With holes:
  NP-hard [Itai, Papadimitriou, Szwarcfiter; 1982]
  $\frac{53}{40}$-approximation [Arkin, Fekete, Mitchell; 2000]
- Without holes: complexity is unknown!
  $\frac{4}{3}$-approximation [Ntafos; 1992]
  $\frac{6}{5}$-approximation [Arkin, Fekete, Mitchell; 2000]

Online

- [Butler; 1998], [Gabriely, Rimon; 2000]
  [Bruckstein, Lindenbaum, Wagner; 2000]
- Survey on covering [Choset; 2001]

A Lower Bound

Theorem

No online exploration strategy achieves a competitive factor better than

$$\frac{7}{6}$$

in simple grid polygons.

Proof.

Adversary strategy. □

A Lower Bound

Theorem

No online exploration strategy achieves a competitive factor better than

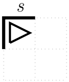
$$\frac{7}{6}$$

in simple grid polygons.

Proof.

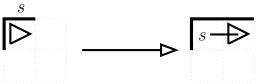
Adversary strategy. □

Proof: Lower Bound



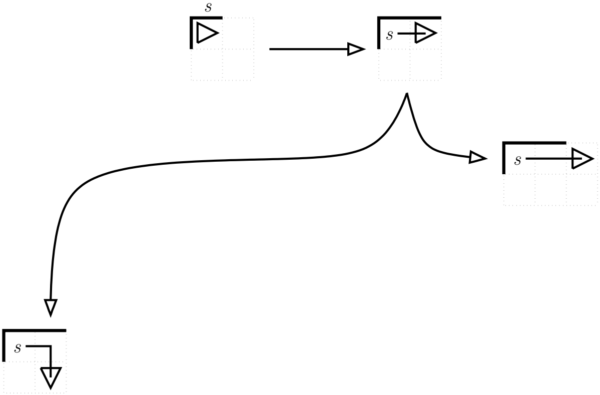
Proof: Lower Bound

W. l. o. g.: east



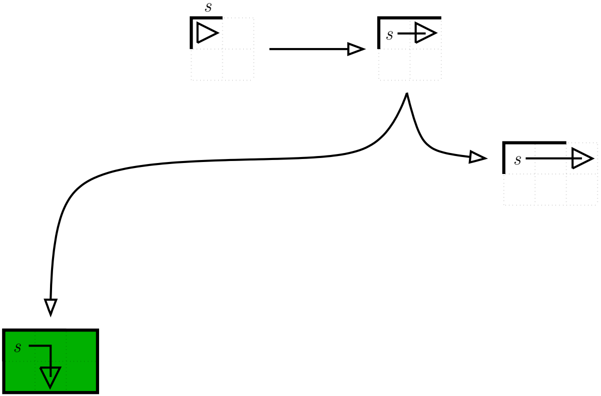
Proof: Lower Bound

South or east

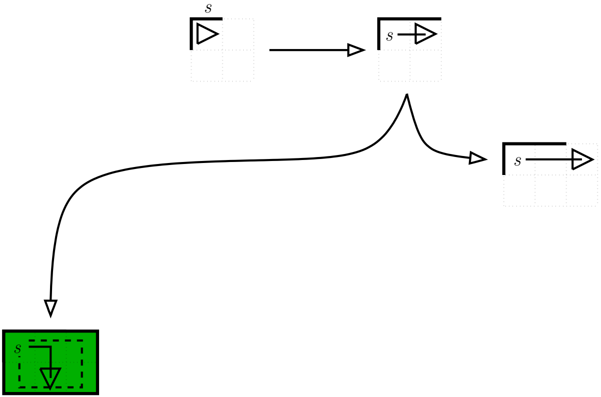


Proof: Lower Bound

Close polygon

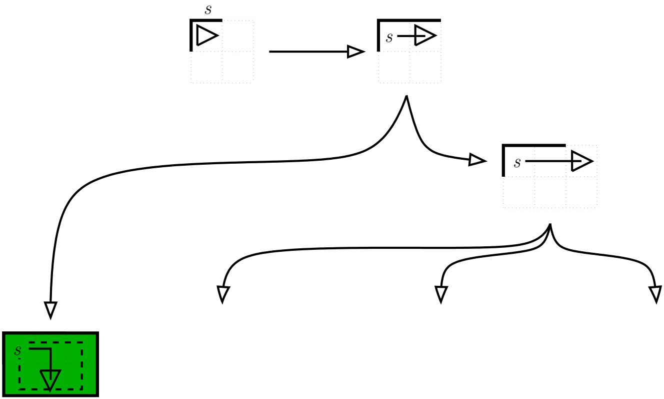


Online vs. optimal



8/6

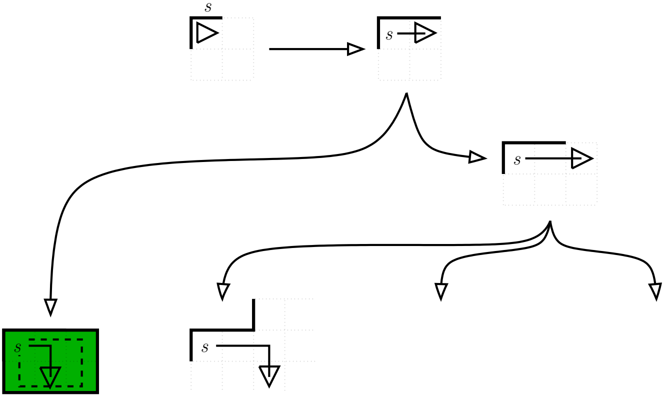
3 possibilities:



8/6

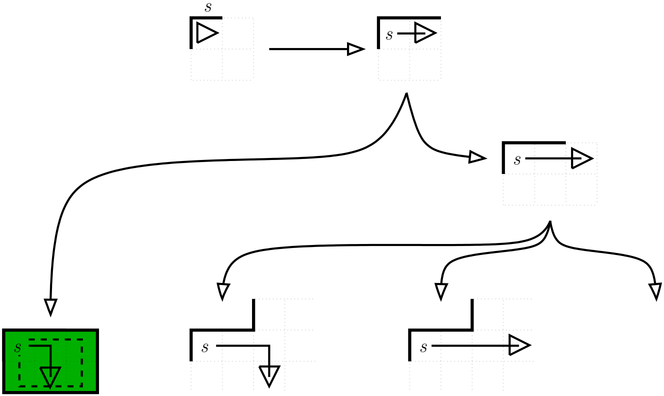
Proof: Lower Bound

3 possibilities: south,



8/6

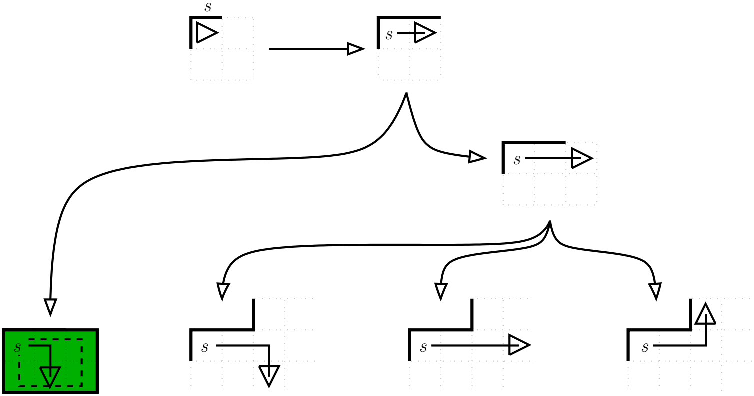
3 possibilities: south, east,



8/6

Proof: Lower Bound

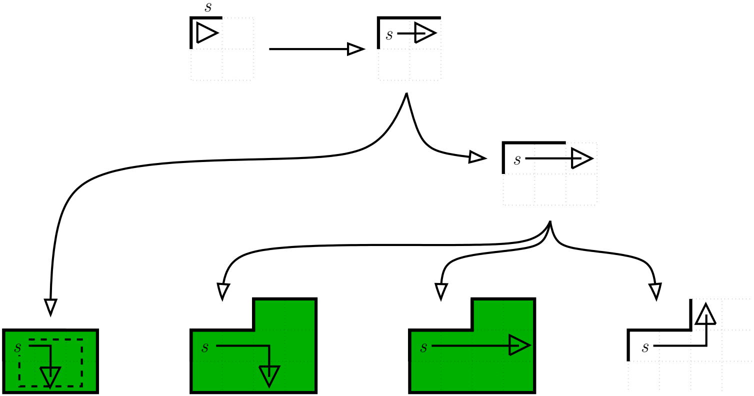
3 possibilities: south, east, north



8/6

Proof: Lower Bound

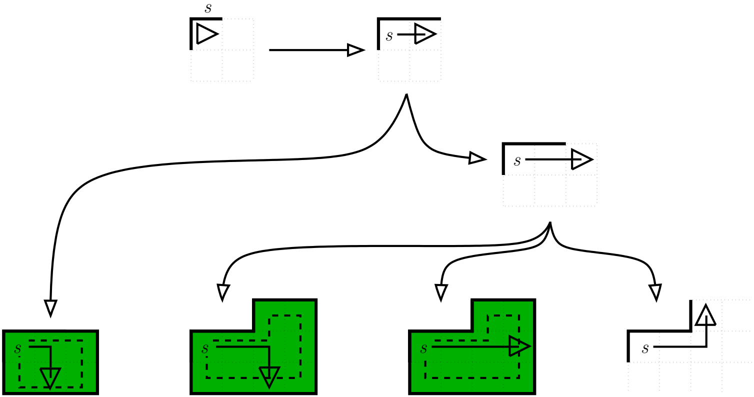
Close polygon



8/6

Proof: Lower Bound

Online vs. optimal



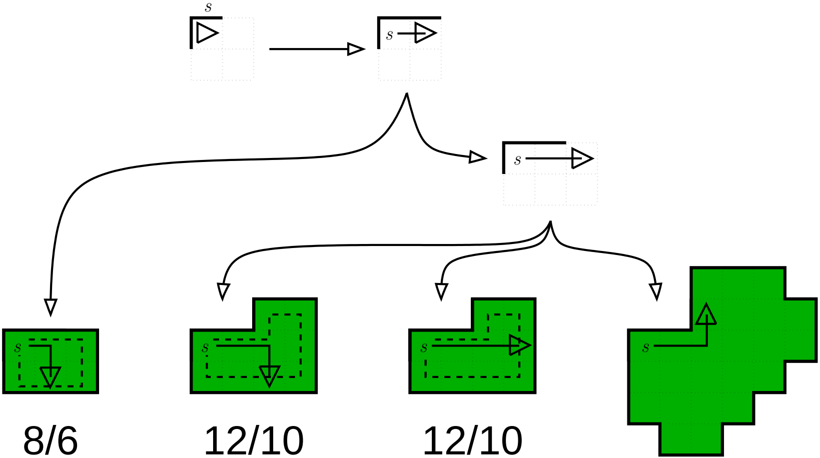
8/6

12/10

12/10

Proof: Lower Bound

Close polygon



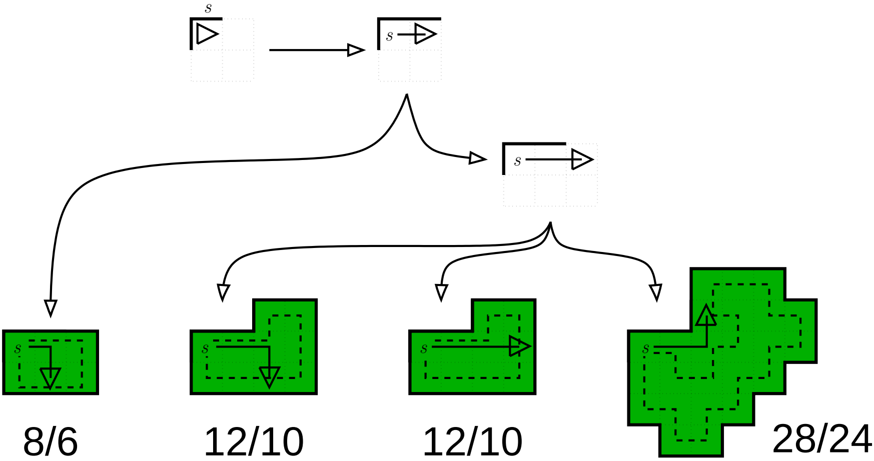
8/6

12/10

12/10

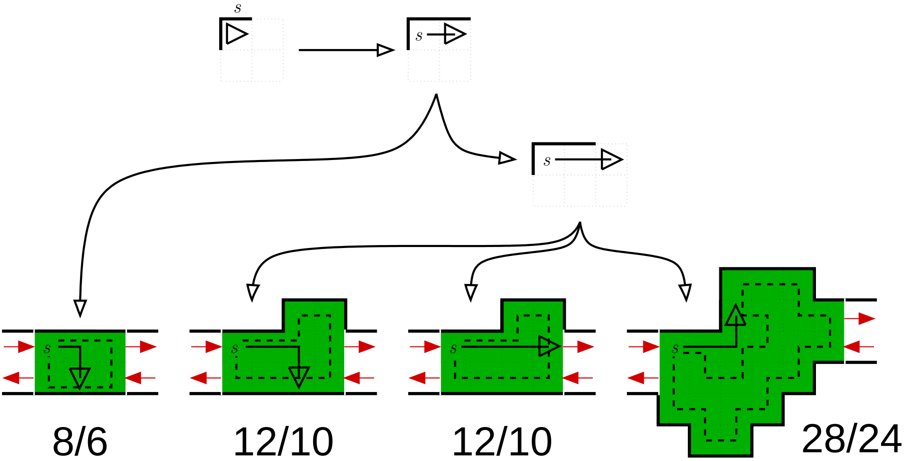
Proof: Lower Bound

Online vs. optimal

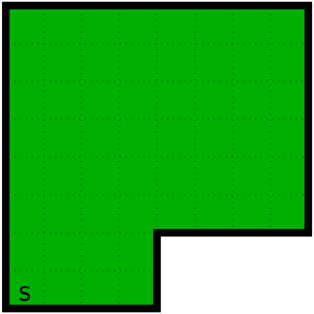


Proof: Lower Bound

Polygons of arbitrary size

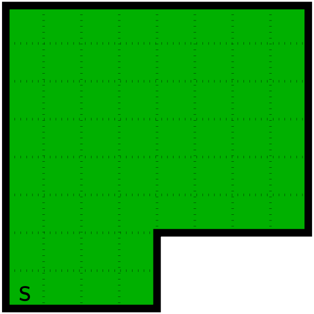


SmartDFS: An exploration strategy (1)



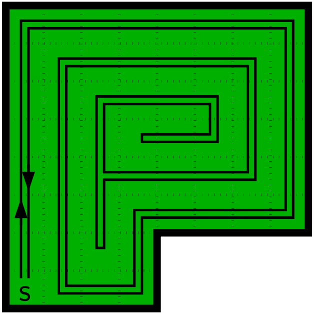
- First idea: Apply depth-first search (DFS)
- *Left-hand rule*: prefer step to the left over a straight step over a step to the right
- Visits *each* cell twice!

SmartDFS: An exploration strategy (1)



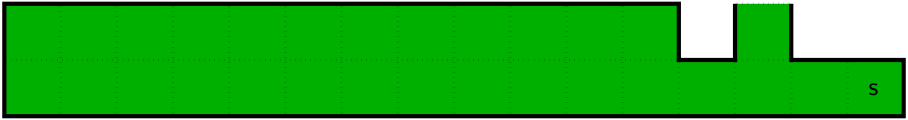
- First idea: Apply depth-first search (DFS)
- *Left-hand rule*: prefer step to the left over a straight step over a step to the right
- Visits *each* cell twice!

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!

SmartDFS: An exploration strategy (2)

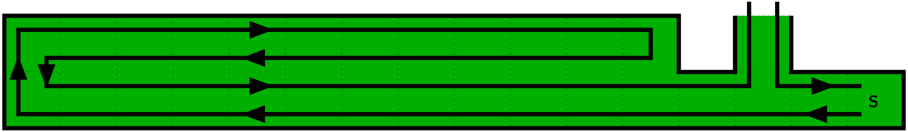


- DFS visits each cell twice
- More reasonable: Return directly to unvisited cell
- Improved DFS

Improvement 1

Return directly to those cells that have unexplored neighbors.

SmartDFS: An exploration strategy (2)

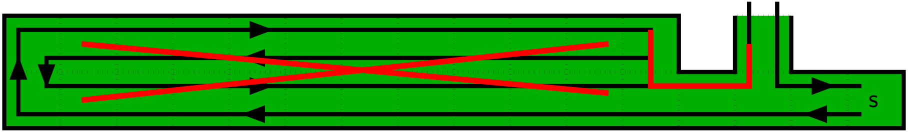


- DFS visits each cell twice
- More reasonable: Return directly to unvisited cell
- Improved DFS

Improvement 1

Return directly to those cells that have unexplored neighbors.

SmartDFS: An exploration strategy (2)

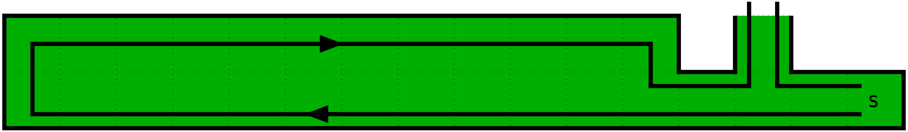


- DFS visits each cell twice
- More reasonable: Return directly to unvisited cell
- Improved DFS

Improvement 1

Return directly to those cells that have unexplored neighbors.

SmartDFS: An exploration strategy (2)

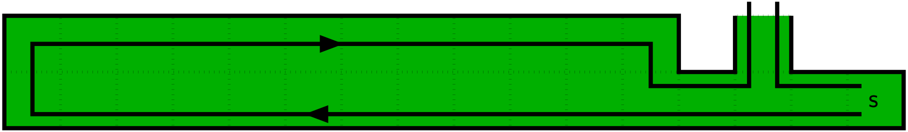


- DFS visits each cell twice
- More reasonable: Return directly to unvisited cell
- Improved DFS

Improvement 1

Return directly to those cells that have unexplored neighbors.

SmartDFS: An exploration strategy (2)

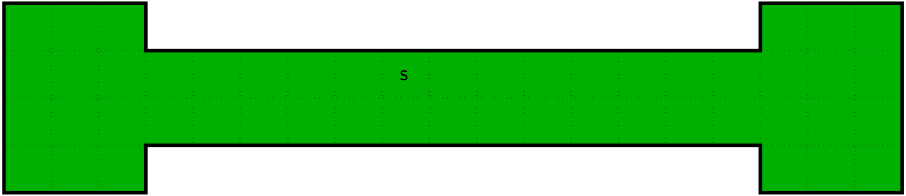


- DFS visits each cell twice
- More reasonable: Return directly to unvisited cell
- Improved DFS

Improvement 1

Return directly to those cells that have unexplored neighbors.

SmartDFS: An exploration strategy (3)

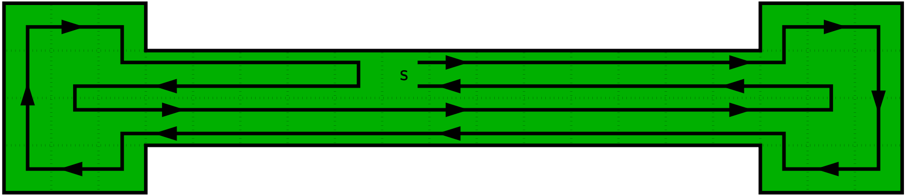


- DFS visits long corridor four times
- More reasonable: Visit right part immediately, continue with the corridor, visit left part, return to s
- Long corridor is traversed only two times!
- *Split cells*: Set of unvisited cells gets disconnected

Improvement 2

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

SmartDFS: An exploration strategy (3)

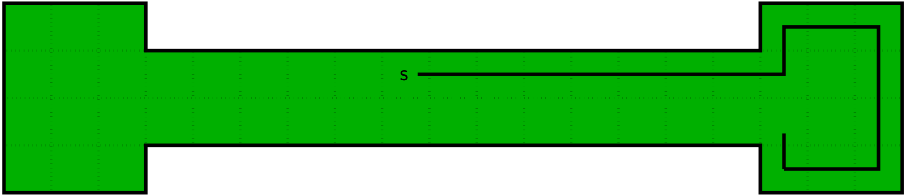


- DFS visits long corridor four times
- More reasonable: Visit right part immediately, continue with the corridor, visit left part, return to s
- Long corridor is traversed only two times!
- *Split cells*: Set of unvisited cells gets disconnected

Improvement 2

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

SmartDFS: An exploration strategy (3)

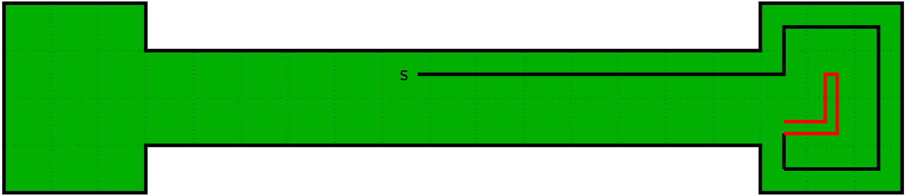


- DFS visits long corridor four times
- More reasonable: Visit right part immediately, continue with the corridor, visit left part, return to *s*
- Long corridor is traversed only two times!
- *Split cells*: Set of unvisited cells gets disconnected

Improvement 2

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

SmartDFS: An exploration strategy (3)

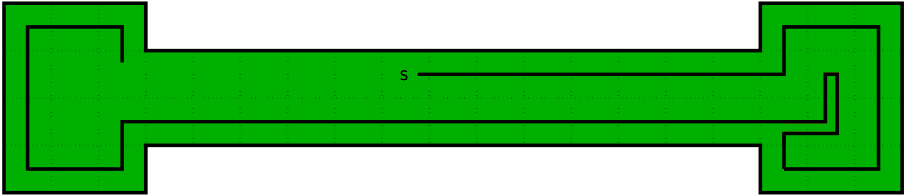


- DFS visits long corridor four times
- More reasonable: Visit right part immediately, continue with the corridor, visit left part, return to s
- Long corridor is traversed only two times!
- *Split cells*: Set of unvisited cells gets disconnected

Improvement 2

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

SmartDFS: An exploration strategy (3)

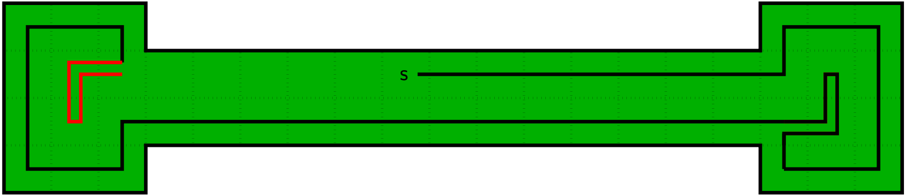


- DFS visits long corridor four times
- More reasonable: Visit right part immediately, continue with the corridor, visit left part, return to *s*
- Long corridor is traversed only two times!
- *Split cells*: Set of unvisited cells gets disconnected

Improvement 2

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

SmartDFS: An exploration strategy (3)

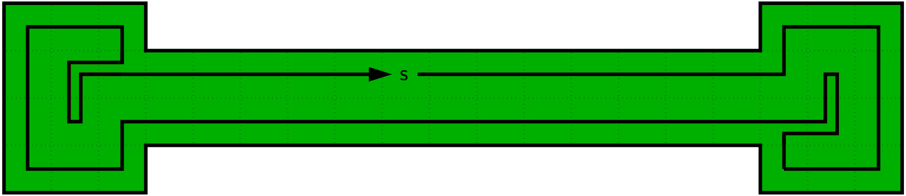


- DFS visits long corridor four times
- More reasonable: Visit right part immediately, continue with the corridor, visit left part, return to *s*
- Long corridor is traversed only two times!
- *Split cells*: Set of unvisited cells gets disconnected

Improvement 2

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

SmartDFS: An exploration strategy (3)

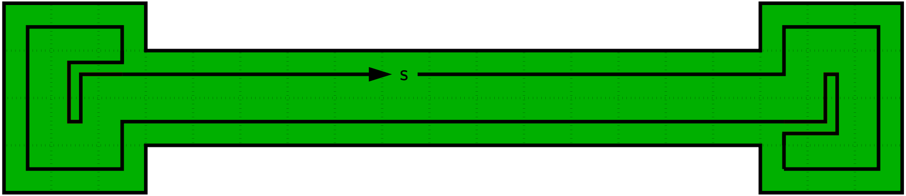


- DFS visits long corridor four times
- More reasonable: Visit right part immediately, continue with the corridor, visit left part, return to s
- Long corridor is traversed only two times!
- *Split cells*: Set of unvisited cells gets disconnected

Improvement 2

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

SmartDFS: An exploration strategy (3)

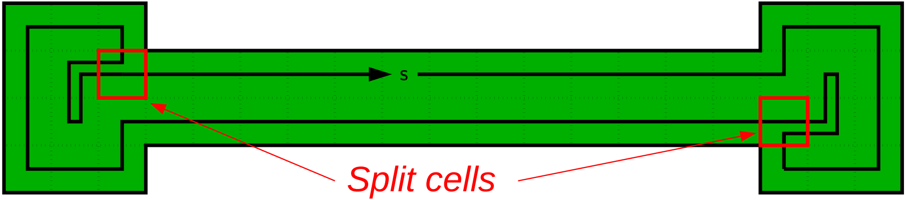


- DFS visits long corridor four times
- More reasonable: Visit right part immediately, continue with the corridor, visit left part, return to *s*
- Long corridor is traversed only two times!
- *Split cells*: Set of unvisited cells gets disconnected

Improvement 2

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

SmartDFS: An exploration strategy (3)

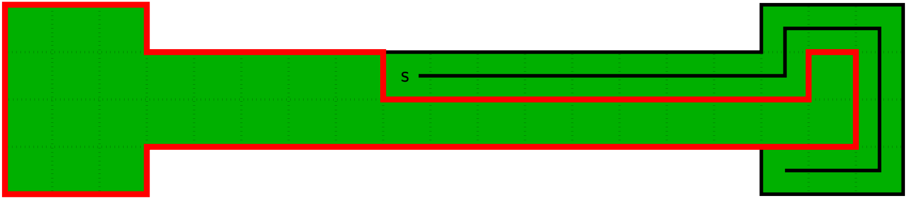


- DFS visits long corridor four times
- More reasonable: Visit right part immediately, continue with the corridor, visit left part, return to *s*
- Long corridor is traversed only two times!
- *Split cells*: Set of unvisited cells gets disconnected

Improvement 2

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

SmartDFS: An exploration strategy (3)

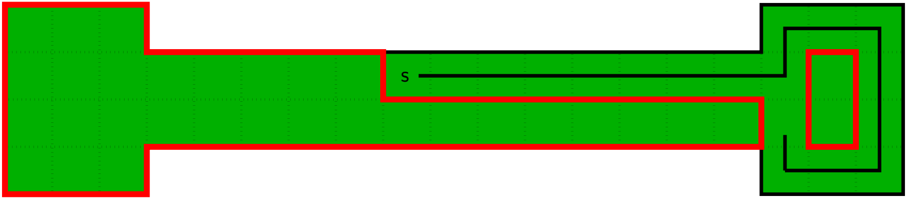


- DFS visits long corridor four times
- More reasonable: Visit right part immediately, continue with the corridor, visit left part, return to *s*
- Long corridor is traversed only two times!
- *Split cells*: Set of unvisited cells gets disconnected

Improvement 2

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

SmartDFS: An exploration strategy (3)

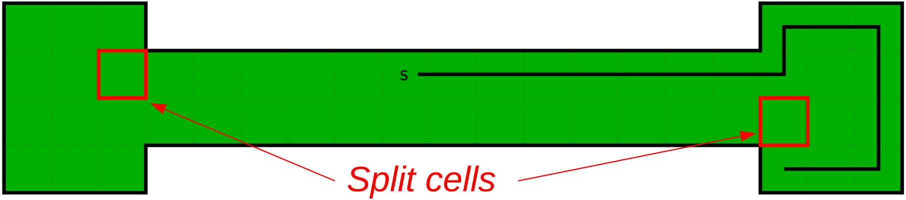


- DFS visits long corridor four times
- More reasonable: Visit right part immediately, continue with the corridor, visit left part, return to *s*
- Long corridor is traversed only two times!
- *Split cells*: Set of unvisited cells gets disconnected

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).

Performance of SmartDFS

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)

Theorem (Competitivity)

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

Performance of SmartDFS

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)

Theorem (Competitivity)

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

Java Applet

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

A Lower Bound

Theorem

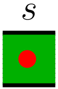
No online exploration strategy achieves a factor better than

$$2$$

in grid polygons with holes.

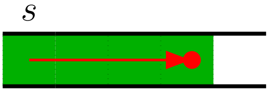
Proof: Lower Bound

- fix large Q, observe strategy's behaviour



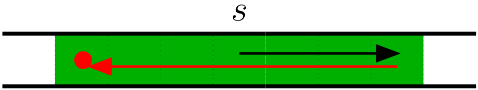
Proof: Lower Bound

- fix large Q, observe strategy's behaviour



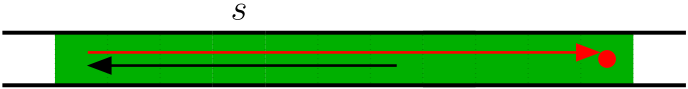
Proof: Lower Bound

- fix large Q, observe strategy's behaviour



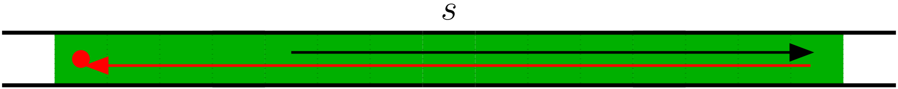
Proof: Lower Bound

- fix large Q, observe strategy's behaviour



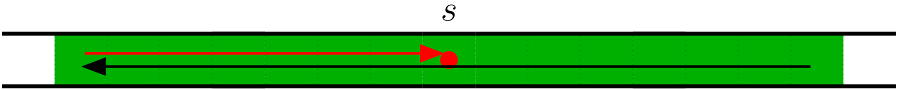
Proof: Lower Bound

- fix large Q, observe strategy's behaviour



Proof: Lower Bound

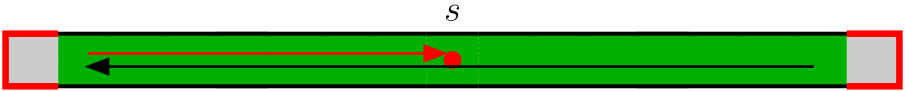
- fix large Q, observe strategy's behaviour



- Case 1: robot returns to s after $Q < S < 2Q$ steps
- $\rightarrow$ close corridor with one unexplored cell at each end
- Robot has walked at least $2R - 2$ steps
- Needs another $2R$ steps to explore the last two cells
- Optimal $2R$, $\frac{\text{Strat}}{\text{Opt}} \rightarrow 2$ for $Q \rightarrow \infty$

Proof: Lower Bound

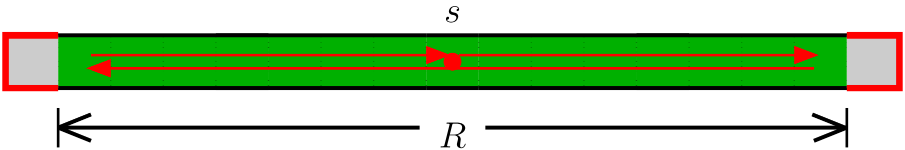
- fix large Q, observe strategy's behaviour



- Case 1: robot returns to s after $Q < S < 2Q$ steps
- $\rightarrow$ close corridor with one unexplored cell at each end
- Robot has walked at least $2R - 2$ steps
- Needs another $2R$ steps to explore the last two cells
- Optimal $2R$, $\frac{\text{Strat}}{\text{Opt}} \rightarrow 2$ for $Q \rightarrow \infty$

Proof: Lower Bound

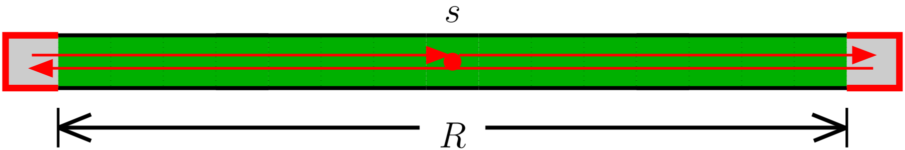
- fix large Q, observe strategy's behaviour



- Case 1: robot returns to s after $Q < S < 2Q$ steps
- $\rightarrow$ close corridor with one unexplored cell at each end
- Robot has walked at least $2R - 2$ steps
- Needs another $2R$ steps to explore the last two cells
- Optimal $2R$, $\frac{\text{Strat}}{\text{Opt}} \rightarrow 2$ for $Q \rightarrow \infty$

Proof: Lower Bound

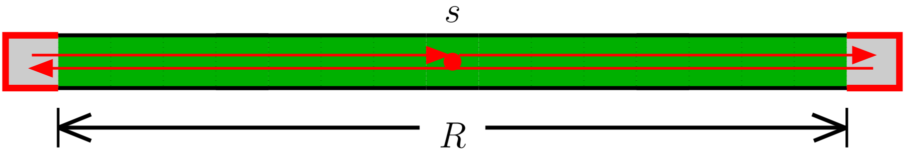
- fix large Q, observe strategy's behaviour



- Case 1: robot returns to s after $Q < S < 2Q$ steps
- $\rightarrow$ close corridor with one unexplored cell at each end
- Robot has walked at least $2R - 2$ steps
- Needs another $2R$ steps to explore the last two cells
- Optimal $2R$, $\frac{\text{Strat}}{\text{Opt}} \rightarrow 2$ for $Q \rightarrow \infty$

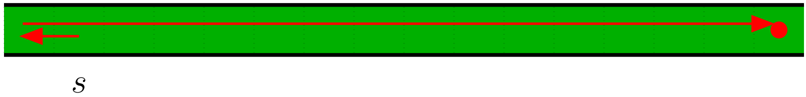
Proof: Lower Bound

- fix large Q, observe strategy's behaviour



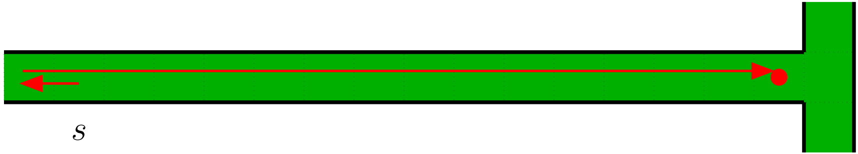
- Case 1: robot returns to s after $Q < S < 2Q$ steps
- $\rightarrow$ close corridor with one unexplored cell at each end
- Robot has walked at least $2R - 2$ steps
- Needs another $2R$ steps to explore the last two cells
- Optimal $2R$, $\frac{\text{Strat}}{\text{Opt}} \rightarrow 2$ for $Q \rightarrow \infty$

Proof: Lower Bound



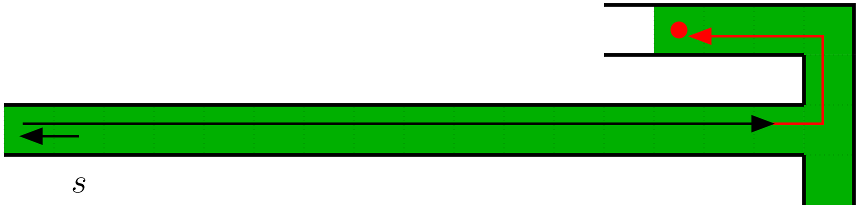
- Case 2: robot prefers on side of the corridor
- → Add a T-crossing, both corridors turn back
- Robot explored one corridor “up to s” → Close corridor
- Robot walked $\approx 2R + 2R'$, needs another $\approx 2R + 2R'$
- Optimal $2R + 2R'$, $\frac{\text{Strat}}{\text{Opt}} \to 2$ for $Q \to \infty$

Proof: Lower Bound



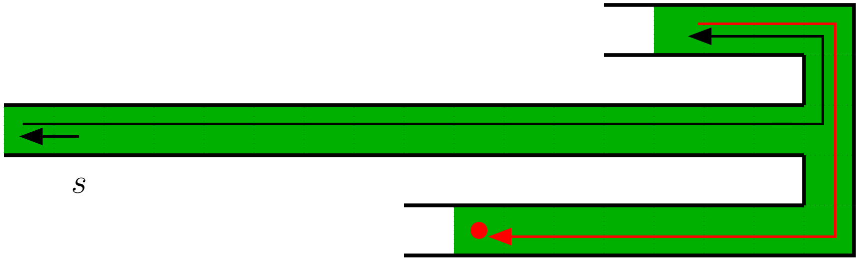
- Case 2: robot prefers on side of the corridor
- → Add a T-crossing, both corridors turn back
- Robot explored one corridor “up to s” → Close corridor
- Robot walked $\approx 2R + 2R'$, needs another $\approx 2R + 2R'$
- Optimal $2R + 2R'$, $\frac{\text{Strat}}{\text{Opt}} \to 2$ for $Q \to \infty$

Proof: Lower Bound



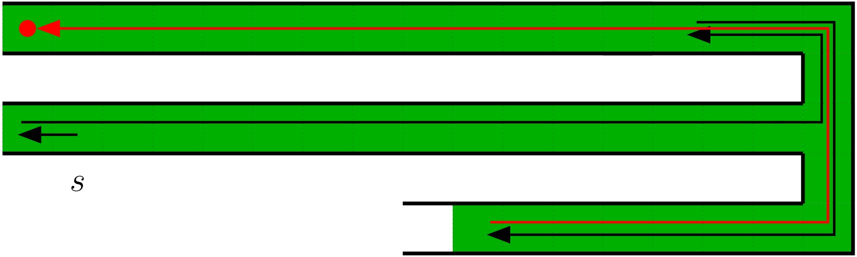
- Case 2: robot prefers on side of the corridor
- → Add a T-crossing, both corridors turn back
- Robot explored one corridor “up to s” → Close corridor
- Robot walked $\approx 2R + 2R'$, needs another $\approx 2R + 2R'$
- Optimal $2R + 2R'$, $\frac{\text{Strat}}{\text{Opt}} \to 2$ for $Q \to \infty$

Proof: Lower Bound



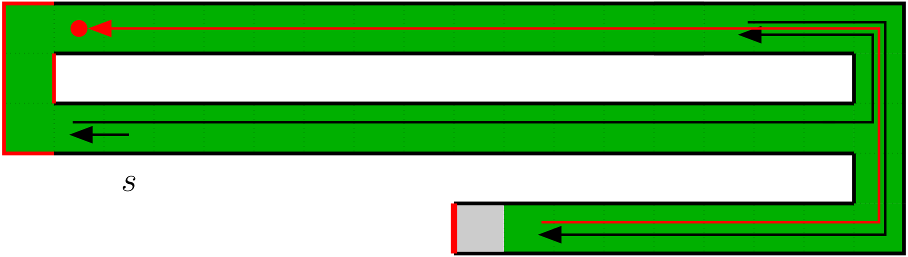
- Case 2: robot prefers on side of the corridor
- → Add a T-crossing, both corridors turn back
- Robot explored one corridor “up to s” → Close corridor
- Robot walked $\approx 2R + 2R'$, needs another $\approx 2R + 2R'$
- Optimal $2R + 2R'$, $\frac{\text{Strat}}{\text{Opt}} \to 2$ for $Q \to \infty$

Proof: Lower Bound



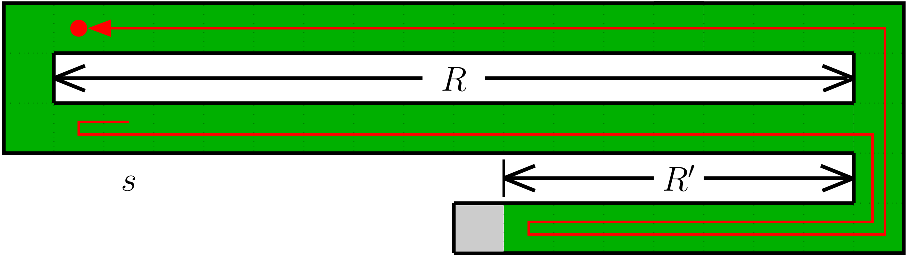
- Case 2: robot prefers on side of the corridor
- $\rightarrow$ Add a T-crossing, both corridors turn back
- Robot explored one corridor “up to s” $\rightarrow$ Close corridor
- Robot walked $\approx 2R + 2R'$, needs another $\approx 2R + 2R'$
- Optimal $2R + 2R'$, $\frac{\text{Strat}}{\text{Opt}} \rightarrow 2$ for $Q \rightarrow \infty$

Proof: Lower Bound



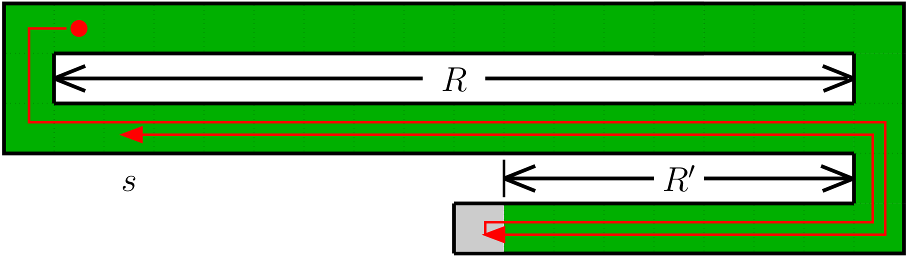
- Case 2: robot prefers on side of the corridor
- → Add a T-crossing, both corridors turn back
- Robot explored one corridor “up to *s*” → Close corridor
- Robot walked $\approx 2R + 2R'$, needs another $\approx 2R + 2R'$
- Optimal $2R + 2R'$, $\frac{\text{Strat}}{\text{Opt}} \to 2$ for $Q \to \infty$

Proof: Lower Bound



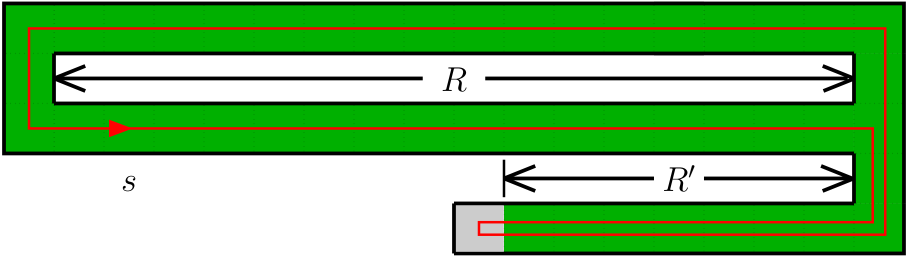
- Case 2: robot prefers on side of the corridor
- → Add a T-crossing, both corridors turn back
- Robot explored one corridor “up to *s*” → Close corridor
- Robot walked $\approx 2R + 2R'$, needs another $\approx 2R + 2R'$
- Optimal $2R + 2R'$, $\frac{\text{Strat}}{\text{Opt}} \to 2$ for $Q \to \infty$

Proof: Lower Bound



- Case 2: robot prefers on side of the corridor
- → Add a T-crossing, both corridors turn back
- Robot explored one corridor “up to s” → Close corridor
- Robot walked $\approx 2R + 2R'$, needs another $\approx 2R + 2R'$
- Optimal $2R + 2R'$, $\frac{\text{Strat}}{\text{Opt}} \to 2$ for $Q \to \infty$

Proof: Lower Bound



- Case 2: robot prefers on side of the corridor
- → Add a T-crossing, both corridors turn back
- Robot explored one corridor “up to s” → Close corridor
- Robot walked $\approx 2R + 2R'$, needs another $\approx 2R + 2R'$
- Optimal $2R + 2R'$, $\frac{\text{Strat}}{\text{Opt}} \rightarrow 2$ for $Q \rightarrow \infty$

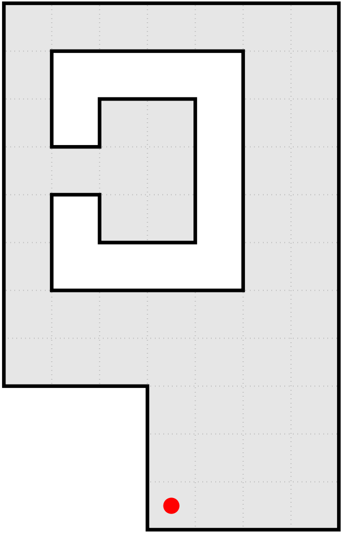
Strategy CellExplore

Forward mode:

- Proceed using left-hand rule
- *Reserve* cells right to (or on) the walked path
- If no forward step is possible: enter backward mode

Backward mode:

- Walk back on reserved cells
- If unexplored cell appears: enter forward mode



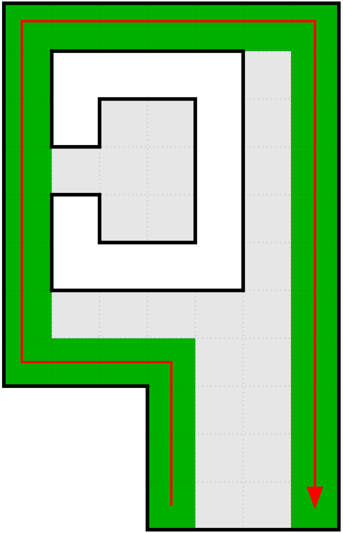
Strategy CellExplore

Forward mode:

- Proceed using left-hand rule
- *Reserve* cells right to (or on) the walked path
- If no forward step is possible: enter backward mode

Backward mode:

- Walk back on reserved cells
- If unexplored cell appears: enter forward mode



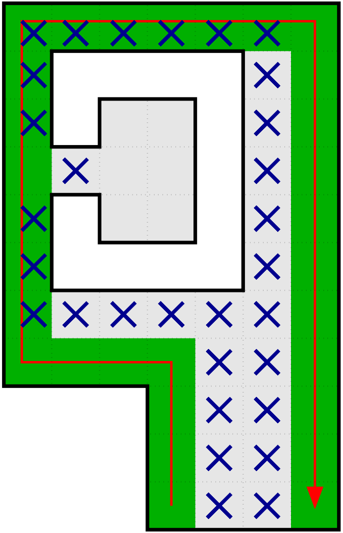
Strategy CellExplore

Forward mode:

- Proceed using left-hand rule
- *Reserve* cells right to (or on) the walked path
- If no forward step is possible: enter backward mode

Backward mode:

- Walk back on reserved cells
- If unexplored cell appears: enter forward mode



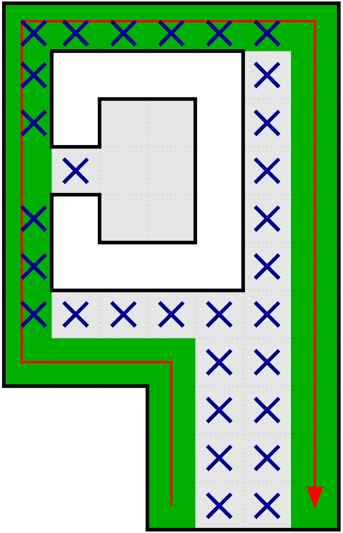
Strategy CellExplore

Forward mode:

- Proceed using left-hand rule
- *Reserve* cells right to (or on) the walked path
- If no forward step is possible: enter backward mode

Backward mode:

- Walk back on reserved cells
- If unexplored cell appears: enter forward mode



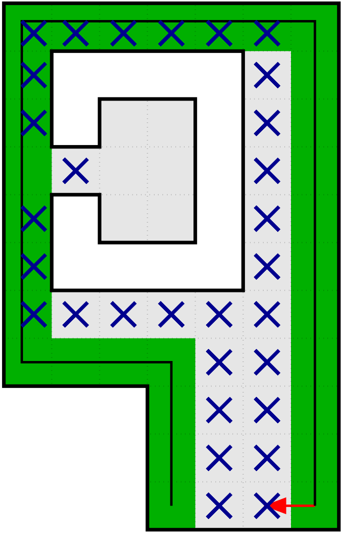
Strategy CellExplore

Forward mode:

- Proceed using left-hand rule
- *Reserve* cells right to (or on) the walked path
- If no forward step is possible: enter backward mode

Backward mode:

- Walk back on reserved cells
- If unexplored cell appears: enter forward mode



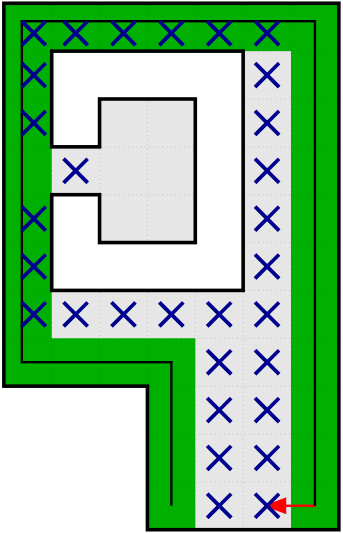
Strategy CellExplore

Forward mode:

- Proceed using left-hand rule
- *Reserve* cells right to (or on) the walked path
- If no forward step is possible: enter backward mode

Backward mode:

- Walk back on reserved cells
- If unexplored cell appears: enter forward mode



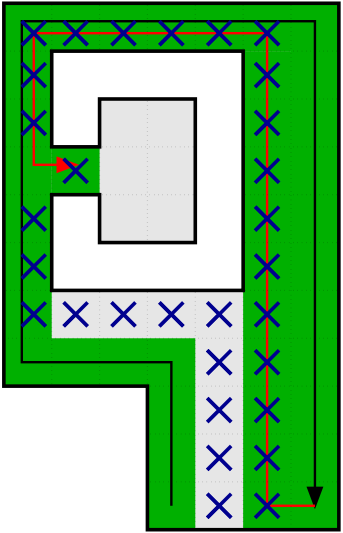
Strategy CellExplore

Forward mode:

- Proceed using left-hand rule
- *Reserve* cells right to (or on) the walked path
- If no forward step is possible: enter backward mode

Backward mode:

- Walk back on reserved cells
- If unexplored cell appears: enter forward mode



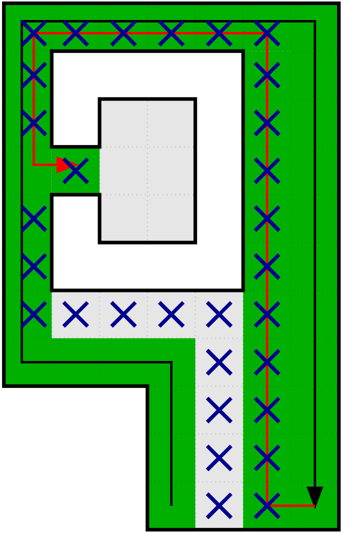
Strategy CellExplore

Forward mode:

- Proceed using left-hand rule
- *Reserve* cells right to (or on) the walked path
- If no forward step is possible: enter backward mode

Backward mode:

- Walk back on reserved cells
- If unexplored cell appears: enter forward mode



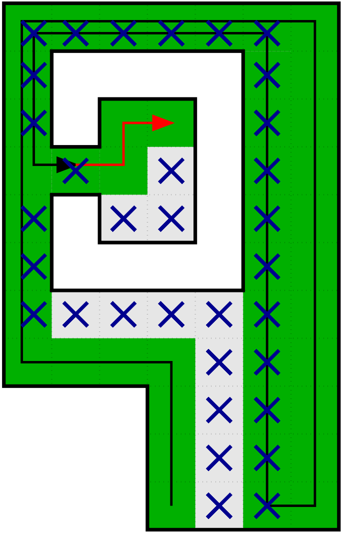
Strategy CellExplore

Forward mode:

- Proceed using left-hand rule
- *Reserve* cells right to (or on) the walked path
- If no forward step is possible: enter backward mode

Backward mode:

- Walk back on reserved cells
- If unexplored cell appears: enter forward mode



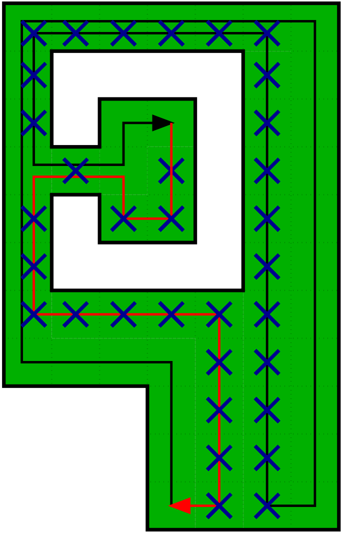
Strategy CellExplore

Forward mode:

- Proceed using left-hand rule
- *Reserve* cells right to (or on) the walked path
- If no forward step is possible: enter backward mode

Backward mode:

- Walk back on reserved cells
- If unexplored cell appears: enter forward mode



Performance of CellExplore

Theorem (Number of Steps)

CellExplore needs at most

$$C + \frac{1}{2}E + 3H + W - 2$$

steps to explore a polygon. This bound is tight.

(C: #cells, E: #boundary edges, H: #holes, W: “sinuosity”)

Performance of CellExplore

Theorem (Number of Steps)

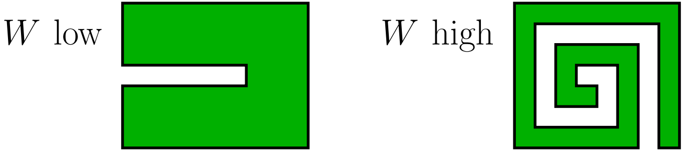
CellExplore needs at most

$$C + \frac{1}{2}E + 3H + W - 2$$

steps to explore a polygon. This bound is tight.

(C: #cells, E: #boundary edges, H: #holes, W: “sinuosity”)

W: distinguish between straight and winded polygons



Java Applet

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

Outline

1. Introduction

2. Exploring Grid Polygons
   - Introduction
   - Simple Grid Polygons
   - Grid Polygons with Holes

3. Search

Searching

- Search for a goal in a given environment, $\mathcal{E}$
- Quality measure?

Quality measure

- *Competitive ratio* for a strategy, $\mathcal{S}$:

$$C := \sup_{\mathcal{E}} \sup_{p \in \mathcal{E}} \frac{|\mathcal{S}(s,p)|}{|\mathrm{sp}(s,p)|}$$

- *Search ratio* for a strategy $\mathcal{S}$ in $\mathcal{E}$:

$$\mathrm{SR}(\mathcal{S},\mathcal{E}) := \sup_{p \in \mathcal{E}} \frac{|\mathcal{S}(s,p)|}{|\mathrm{sp}(s,p)|}$$

  (Koutsoupias et al.; 1996: offline search in graphs)

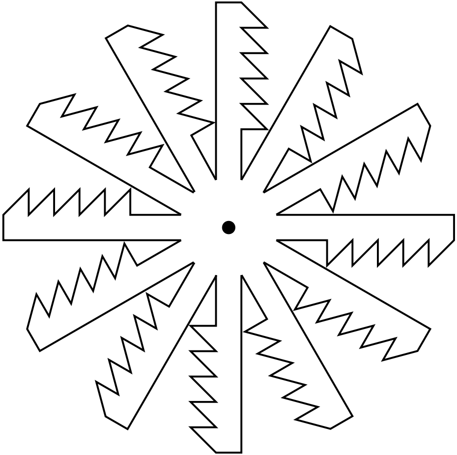
- *Optimal search ratio*: $\mathrm{SR}_{\mathrm{OPT}}(\mathcal{E}) := \inf_{\mathcal{S}} \mathrm{SR}(\mathcal{S},\mathcal{E})$

- Approximation: $\mathcal{S}$ *Search-competitive*

$$C_S := \sup_{\mathcal{E}} \frac{\mathrm{SR}(\mathcal{S},\mathcal{E})}{\mathrm{SR}_{\mathrm{OPT}}(\mathcal{E})}$$

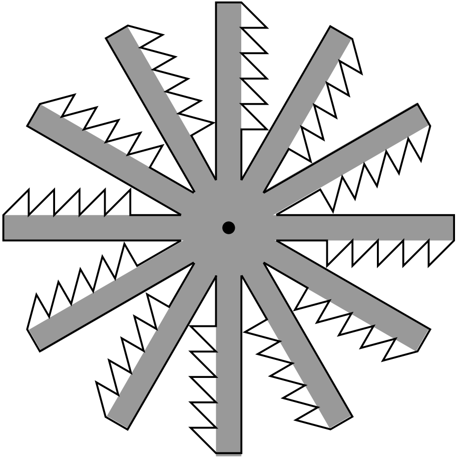
Competitive ratio

- Searching in a polygon
- Searcher has vision
- Adversary can force every strategy to explore every corridor
- Optimal path is very short
- ⇒ every strategy is 'bad' (i.e., not constant-competitive)



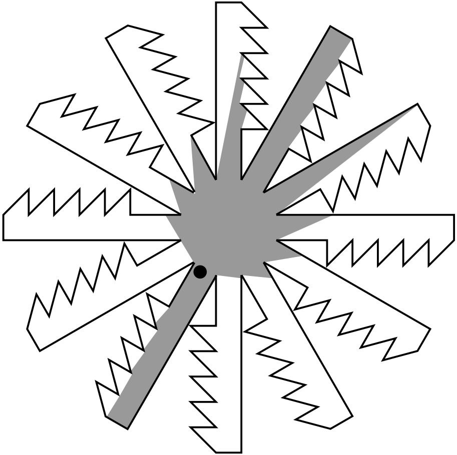
Competitive ratio

- Searching in a polygon
- Searcher has vision
- Adversary can force every strategy to explore every corridor
- Optimal path is very short
- ⇒ every strategy is 'bad' (i.e., not constant-competitive)



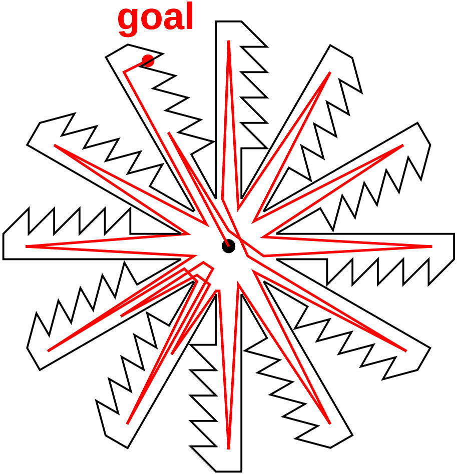
Competitive ratio

- Searching in a polygon
- Searcher has vision
- Adversary can force every strategy to explore every corridor
- Optimal path is very short
- ⇒ every strategy is 'bad' (i.e., not constant-competitive)



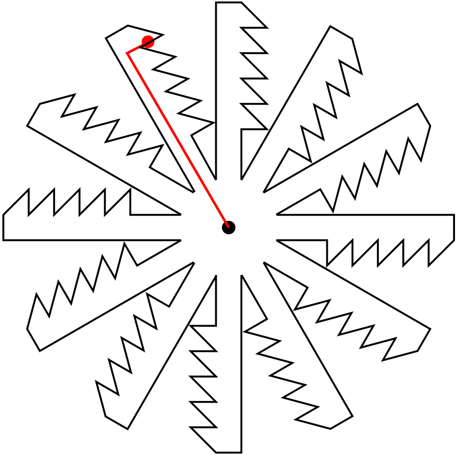
Competitive ratio

- Searching in a polygon
- Searcher has vision
- Adversary can force every strategy to explore every corridor
- Optimal path is very short
- ⇒ every strategy is 'bad' (i.e., not constant-competitive)



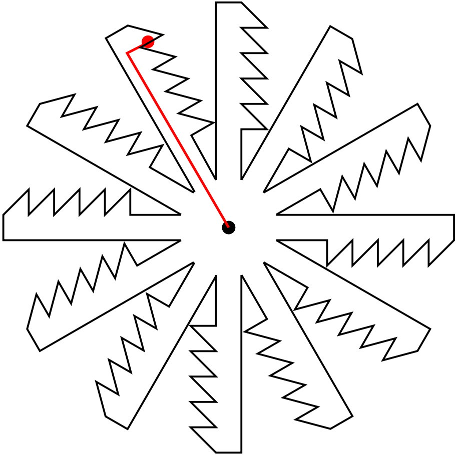
Competitive ratio

- Searching in a polygon
- Searcher has vision
- Adversary can force every strategy to explore every corridor
- Optimal path is very short
- ⇒ every strategy is ’bad’ (i.e., not constant-competitive)

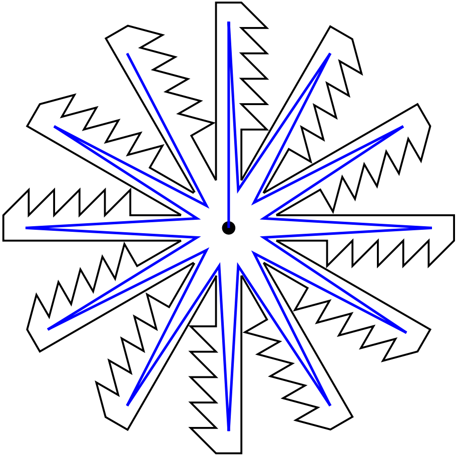


Competitive ratio

- Searching in a polygon
- Searcher has vision
- Adversary can force every strategy to explore every corridor
- Optimal path is very short
- ⇒ every strategy is ’bad’ (i.e., not constant-competitive)

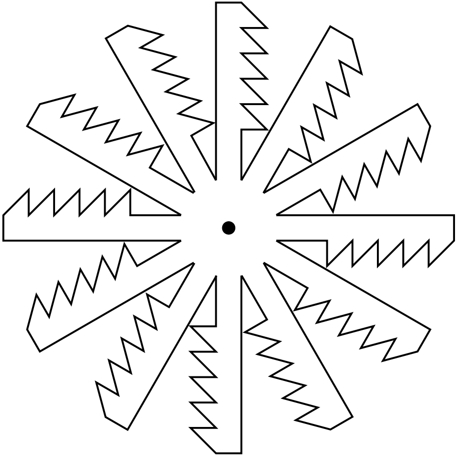


- Strat1: explore every corridor completely
- Strat2:
  visit corridors up to $d = 1$
  visit corridors up to $d = 2$
  visit corridors up to $d = 4$ etc.
- Strat2 seems to be 'better': visits points near to s earlier
- Can we measure this quality?



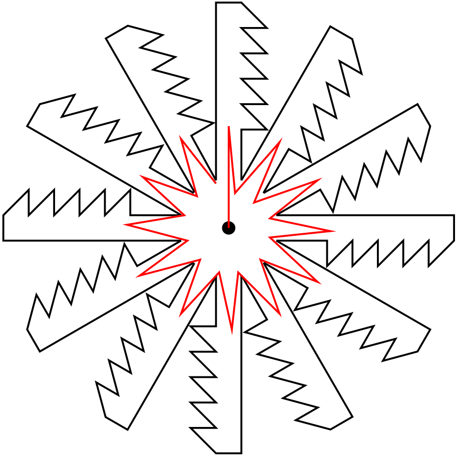
Competitive ratio

- Strat1: explore every corridor completely
- Strat2:
  visit corridors up to $d = 1$
  visit corridors up to $d = 2$
  visit corridors up to $d = 4$ etc.
- Strat2 seems to be 'better':
  visits points near to s earlier
- Can we measure this quality?



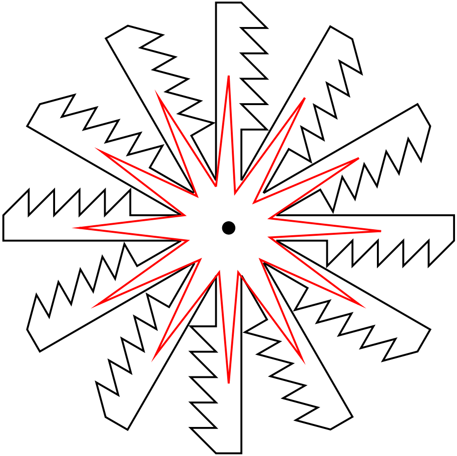
Competitive ratio

- Strat1: explore every corridor completely
- Strat2:
  visit corridors up to $d = 1$
  visit corridors up to $d = 2$
  visit corridors up to $d = 4$ etc.
- Strat2 seems to be 'better': visits points near to s earlier
- Can we measure this quality?



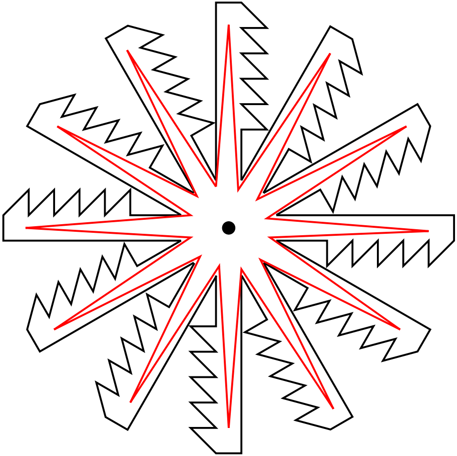
Competitive ratio

- Strat1: explore every corridor completely
- Strat2:
  visit corridors up to $d = 1$
  visit corridors up to $d = 2$
  visit corridors up to $d = 4$ etc.
- Strat2 seems to be 'better':
  visits points near to s earlier
- Can we measure this quality?



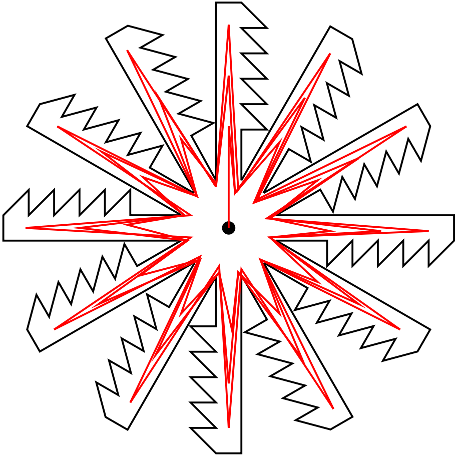
Competitive ratio

- Strat1: explore every corridor completely
- Strat2:
  visit corridors up to $d = 1$
  visit corridors up to $d = 2$
  visit corridors up to $d = 4$ etc.
- Strat2 seems to be 'better': visits points near to s earlier
- Can we measure this quality?



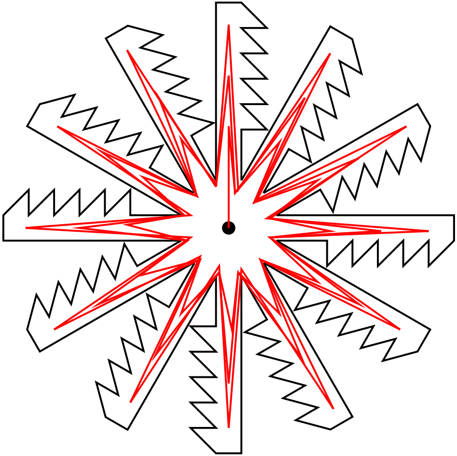
Competitive ratio

- Strat1: explore every corridor completely
- Strat2:
  visit corridors up to $d = 1$
  visit corridors up to $d = 2$
  visit corridors up to $d = 4$ etc.
- Strat2 seems to be 'better':
  visits points near to s earlier
- Can we measure this quality?



Competitive ratio

- Strat1: explore every corridor completely
- Strat2:
  visit corridors up to $d = 1$
  visit corridors up to $d = 2$
  visit corridors up to $d = 4$ etc.
- Strat2 seems to be 'better': visits points near to s earlier
- Can we measure this quality?



Quality measure

- *Competitive ratio* for a strategy, $\mathcal{S}$:

$$C := \sup_{\mathcal{E}} \sup_{p \in \mathcal{E}} \frac{|\mathcal{S}(s,p)|}{|\mathrm{sp}(s,p)|}$$

- *Search ratio* for a strategy $\mathcal{S}$ in $\mathcal{E}$:

$$\mathrm{SR}(\mathcal{S},\mathcal{E}) := \sup_{p \in \mathcal{E}} \frac{|\mathcal{S}(s,p)|}{|\mathrm{sp}(s,p)|}$$

  (Koutsoupias et al.; 1996: offline search in graphs)
- *Optimal search ratio*: $\mathrm{SR}_{\mathrm{OPT}}(\mathcal{E}) := \inf_{\mathcal{S}} \mathrm{SR}(\mathcal{S},\mathcal{E})$
- Approximation: $\mathcal{S}$ *Search-competitive*

$$C_S := \sup_{\mathcal{E}} \frac{\mathrm{SR}(\mathcal{S},\mathcal{E})}{\mathrm{SR}_{\mathrm{OPT}}(\mathcal{E})}$$

Quality measure

- *Competitive ratio* for a strategy, $\mathcal{S}$:

$$C := \sup_{\mathcal{E}} \sup_{p \in \mathcal{E}} \frac{|\mathcal{S}(s,p)|}{|\mathrm{sp}(s,p)|}$$

- *Search ratio* for a strategy $\mathcal{S}$ in $\mathcal{E}$:

$$\mathrm{SR}(\mathcal{S},\mathcal{E}) := \sup_{p \in \mathcal{E}} \frac{|\mathcal{S}(s,p)|}{|\mathrm{sp}(s,p)|}$$

(Koutsoupias et al.; 1996: offline search in graphs)

- *Optimal search ratio*: $\mathrm{SR}_{\mathrm{OPT}}(\mathcal{E}) := \inf_{\mathcal{S}} \mathrm{SR}(\mathcal{S},\mathcal{E})$
- Approximation: $\mathcal{S}$ *Search-competitive*

$$C_S := \sup_{\mathcal{E}} \frac{\mathrm{SR}(\mathcal{S},\mathcal{E})}{\mathrm{SR}_{\mathrm{OPT}}(\mathcal{E})}$$

Quality measure

- *Competitive ratio* for a strategy, $\mathcal{S}$:

$$C := \sup_{\mathcal{E}} \sup_{p \in \mathcal{E}} \frac{|\mathcal{S}(s,p)|}{|\mathrm{sp}(s,p)|}$$

- *Search ratio* for a strategy $\mathcal{S}$ in $\mathcal{E}$:

$$\mathrm{SR}(\mathcal{S},\mathcal{E}) := \sup_{p \in \mathcal{E}} \frac{|\mathcal{S}(s,p)|}{|\mathrm{sp}(s,p)|}$$

  (Koutsoupias et al.; 1996: offline search in graphs)

- *Optimal search ratio*: $\mathrm{SR}_{\mathrm{OPT}}(\mathcal{E}) := \inf_{\mathcal{S}} \mathrm{SR}(\mathcal{S},\mathcal{E})$

- Approximation: $\mathcal{S}$ *Search-competitive*

$$C_S := \sup_{\mathcal{E}} \frac{\mathrm{SR}(\mathcal{S},\mathcal{E})}{\mathrm{SR}_{\mathrm{OPT}}(\mathcal{E})}$$

Quality measure

- *Competitive ratio* for a strategy, $\mathcal{S}$:

$$C := \sup_{\mathcal{E}} \sup_{p \in \mathcal{E}} \frac{|\mathcal{S}(s,p)|}{|\mathrm{sp}(s,p)|}$$

- *Search ratio* for a strategy $\mathcal{S}$ in $\mathcal{E}$:

$$\mathrm{SR}(\mathcal{S},\mathcal{E}) := \sup_{p \in \mathcal{E}} \frac{|\mathcal{S}(s,p)|}{|\mathrm{sp}(s,p)|}$$

  (Koutsoupias et al.; 1996: offline search in graphs)

- *Optimal search ratio*: $\mathrm{SR}_{\mathrm{OPT}}(\mathcal{E}) := \inf_{\mathcal{S}} \mathrm{SR}(\mathcal{S},\mathcal{E})$
- Approximation: $\mathcal{S}$ *Search-competitive*

$$C_s := \sup_{\mathcal{E}} \frac{\mathrm{SR}(\mathcal{S},\mathcal{E})}{\mathrm{SR}_{\mathrm{OPT}}(\mathcal{E})}$$

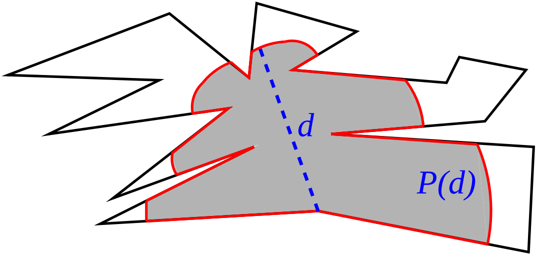
Depth-Restrictable Exploration

Definition

An exploration algorithm, *Expl*, for $\mathcal{E}$ is **depth restrictable**:

- $\mathit{Expl}(d)$: explore $\mathcal{E}$ only up to depth $d \geq 1$
- $\mathit{Expl}(d)$ is C-competitive, i.e., $\exists\, C \geq 1, \beta > 0 :\ \forall \mathcal{E}$:

$$|\mathit{Expl}(d)| \leq C \cdot |\mathit{Expl}_{\text{opt}}(\beta \cdot d)|\,.$$



Approximation Framework

Approximation Strategy

Use **Doubling paradigm**: call $Expl\,(2^i)$, $i = 1, 2, 3, \ldots$.

Theorem

Let $\mathcal{E}$ be an environment fulfilling $\forall p \in \mathcal{E} : |sp(s,p)| = |sp(p,s)|$, Expl be a C-competitive, depth-restrictable exploration algorithm for $\mathcal{E}$.

Searching with $Expl\,(2^i)$, $i = 1, 2, 3, \ldots$ yields a

- *$4\beta C$–search-competitive strategy (blind agent)*
- *$8\beta C$–search-competitive strategy (agent has vision)*

(β: enlargement factor for depth restriction)

Approximation Framework

Approximation Strategy

Use **Doubling paradigm**: call $\mathit{Expl}\,(2^i)$, $i = 1, 2, 3, \ldots$.

Theorem

Let $\mathcal{E}$ be an environment fulfilling $\forall p \in \mathcal{E} : |sp(s,p)| = |sp(p,s)|$, Expl be a C-competitive, depth-restrictable exploration algorithm for $\mathcal{E}$.

Searching with $\mathit{Expl}\,(2^i)$, $i = 1, 2, 3, \ldots$ yields a

- *$4\beta C$–search-competitive strategy (blind agent)*
- *$8\beta C$–search-competitive strategy (agent has vision)*

(β: enlargement factor for depth restriction)

Searching in Simple Polygons

- Shortest Watchman Route (Dror et al., 2003)
  $\Rightarrow$ offline 8–search-competitive strategy
- $\sqrt{2}$-competitive exploration for rectilinear polygons (Deng et al., 1991)
  $\Rightarrow$ $8\sqrt{2}$–search-competitive online strategy for rectilinear polygons
- 26.5-competitive exploration strategy *PolyExplore* (Hoffmann et al., 1998)
  $\Rightarrow$ 212–search-competitive online strategy for simple polygons

Searching in Simple Polygons

- Shortest Watchman Route (Dror et al., 2003)
  $\Rightarrow$ offline 8–search-competitive strategy
- $\sqrt{2}$-competitive exploration for rectilinear polygons (Deng et al., 1991)
  $\Rightarrow$ $8\sqrt{2}$–search-competitive online strategy for rectilinear polygons
- 26.5-competitive exploration strategy *PolyExplore* (Hoffmann et al., 1998)
  $\Rightarrow$ 212–search-competitive online strategy for simple polygons

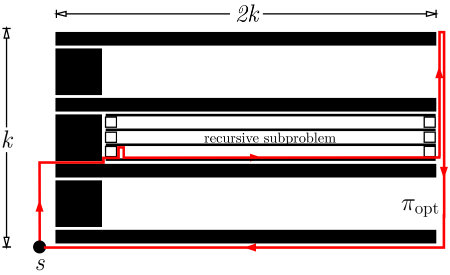
Searching in Simple Polygons

- Shortest Watchman Route (Dror et al., 2003)
  $\Rightarrow$ offline 8–search-competitive strategy
- $\sqrt{2}$-competitive exploration for rectilinear polygons (Deng et al., 1991)
  $\Rightarrow$ $8\sqrt{2}$–search-competitive online strategy for rectilinear polygons
- 26.5-competitive exploration strategy *PolyExplore* (Hoffmann et al., 1998)
  $\Rightarrow$ 212–search-competitive online strategy for simple polygons

Searching in Polygons with Holes

- No $O(1)$-competitive exploration for polygons with holes (Albers et al., 1999)
- Optimal exploration path has already bad search ratio
- Enlarge environment
- Optimal exploration path has constant search ratio
- Any online path still has search ratio $\Omega(k)$

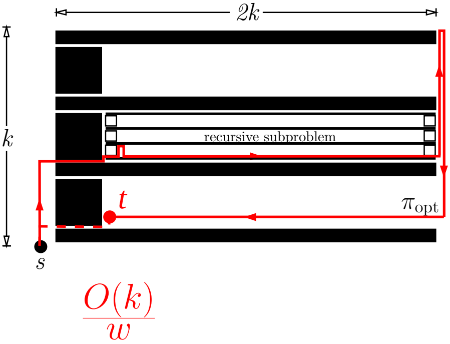
$\Rightarrow$ No search-competitive strategy



Searching in Polygons with Holes

- No $O(1)$-competitive exploration for polygons with holes (Albers et al., 1999)
- Optimal exploration path has already bad search ratio
- Enlarge environment
- Optimal exploration path has constant search ratio
- Any online path still has search ratio $\Omega(k)$

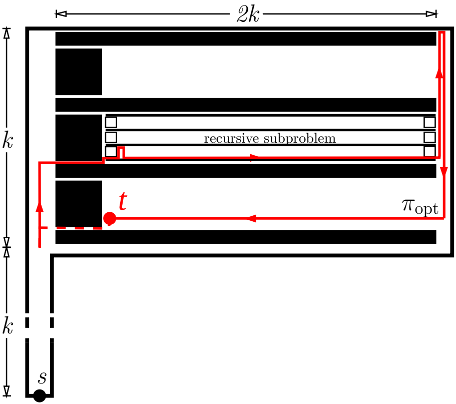
$\Rightarrow$ No search-competitive strategy



Searching in Polygons with Holes

- No $O(1)$-competitive exploration for polygons with holes (Albers et al., 1999)
- Optimal exploration path has already bad search ratio
- Enlarge environment
- Optimal exploration path has constant search ratio
- Any online path still has search ratio $\Omega(k)$

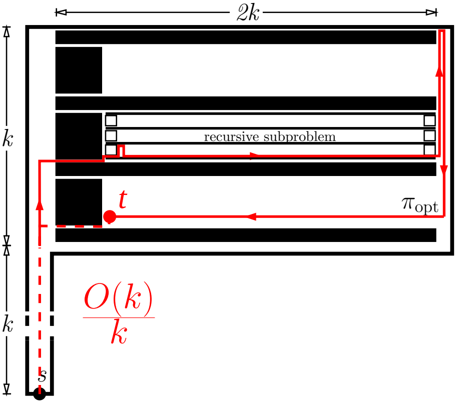
$\Rightarrow$ No search-competitive strategy



Searching in Polygons with Holes

- No $O(1)$-competitive exploration for polygons with holes (Albers et al., 1999)
- Optimal exploration path has already bad search ratio
- Enlarge environment
- Optimal exploration path has constant search ratio
- Any online path still has search ratio $\Omega(k)$

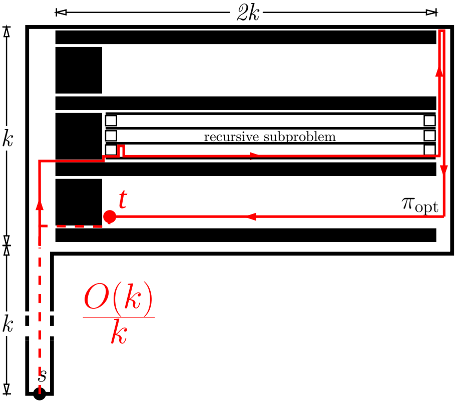
$\Rightarrow$ No search-competitive strategy



Searching in Polygons with Holes

- No $O(1)$-competitive exploration for polygons with holes (Albers et al., 1999)
- Optimal exploration path has already bad search ratio
- Enlarge environment
- Optimal exploration path has constant search ratio
- Any online path still has search ratio $\Omega(k)$

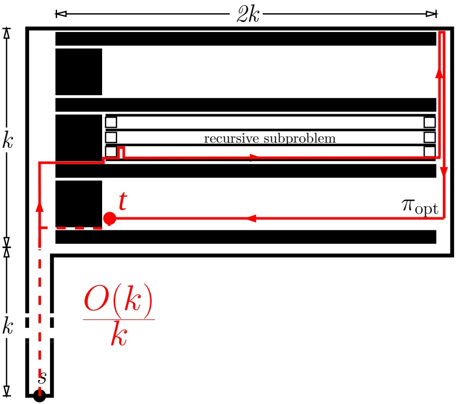
$\Rightarrow$ No search-competitive strategy



Searching in Polygons with Holes

- No $O(1)$-competitive exploration for polygons with holes (Albers et al., 1999)
- Optimal exploration path has already bad search ratio
- Enlarge environment
- Optimal exploration path has constant search ratio
- Any online path still has search ratio $\Omega(k)$

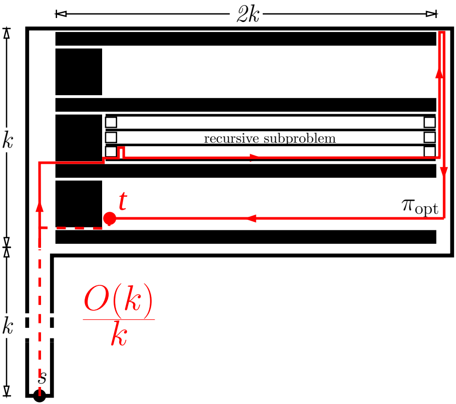
$\Rightarrow$ No search-competitive strategy



Searching in Polygons with Holes

- No $O(1)$-competitive exploration for polygons with holes (Albers et al., 1999)
- Optimal exploration path has already bad search ratio
- Enlarge environment
- Optimal exploration path has constant search ratio
- Any online path still has search ratio $\Omega(k)$

$\Rightarrow$ No search-competitive strategy



General Lower Bound

Theorem

If for a given type of environments

- *there is no constant-competitive exploration strategy*
- *the lower-bound scene can be enlarged*

$\Rightarrow$ *there is no search-competitive strategy.*

Relation Between Searching and Exploring

Close relation

- $\exists$ constant-competitive, depth-restrictable exploration strategy
  $\Rightarrow \exists$ search-competitive strategy
- $\nexists$ constant-competitive exploration strategy,
  but $\exists$ 'extendable' lower bound
  $\Rightarrow \nexists$ search-competitive strategy

Open question

$\exists$ search-competitive strategy
$\overset{?}{\Longleftrightarrow} \exists$ constant-competitive exploration strategy
(for environments fulfilling $\forall p \in \mathcal{E} : |\mathrm{sp}(s, p)| = |\mathrm{sp}(p, s)|$)

Relation Between Searching and Exploring

Close relation

- $\exists$ constant-competitive, depth-restrictable exploration strategy
  $\Rightarrow \exists$ search-competitive strategy
- $\nexists$ constant-competitive exploration strategy,
  but $\exists$ 'extendable' lower bound
  $\Rightarrow \nexists$ search-competitive strategy

Open question

$\exists$ search-competitive strategy

$\overset{?}{\Longleftrightarrow} \exists$ constant-competitive exploration strategy

(for environments fulfilling $\forall p \in \mathcal{E} : |\mathrm{sp}(s,p)| = |\mathrm{sp}(p,s)|$)

Summary

Onl. exploration of grid polygons

- Simple polygons
  - Lower bound: $\frac{7}{6}$
  - Expl. strategy *SmartDFS*
  - $S \leq C + \frac{1}{2}E - 3$
  - $\frac{4}{3}$-competitive
- Grid polygons with holes
  - Lower bound: 2
  - Expl. strategy *CellExplore*
  - $S \leq C + \frac{1}{2}E + 3H + W - 2$

Searching

- Quality measure: search ratio
- Approximation framework
- Applied to simple polygons
- Lower bound for polygons with holes
- Relation between exploration and searching

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

Summary

Onl. exploration of grid polygons

- Simple polygons
  - Lower bound: $\frac{7}{6}$
  - Expl. strategy *SmartDFS*
  - $S \leq C + \frac{1}{2}E - 3$
  - $\frac{4}{3}$-competitive
- Grid polygons with holes
  - Lower bound: 2
  - Expl. strategy *CellExplore*
  - $S \leq C + \frac{1}{2}E + 3H + W - 2$

Searching

- Quality measure: search ratio
- Approximation framework
- Applied to simple polygons
- Lower bound for polygons with holes
- Relation between exploration and searching

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

Summary

Onl. exploration of grid polygons

- Simple polygons
  - Lower bound: $\frac{7}{6}$
  - Expl. strategy *SmartDFS*
  - $S \leq C + \frac{1}{2}E - 3$
  - $\frac{4}{3}$-competitive
- Grid polygons with holes
  - Lower bound: 2
  - Expl. strategy *CellExplore*
  - $S \leq C + \frac{1}{2}E + 3H + W - 2$

Searching

- Quality measure: search ratio
- Approximation framework
- Applied to simple polygons
- Lower bound for polygons with holes
- Relation between exploration and searching

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

Summary

Onl. exploration of grid polygons

- Simple polygons
  - Lower bound: $\frac{7}{6}$
  - Expl. strategy *SmartDFS*
  - $S \leq C + \frac{1}{2}E - 3$
  - $\frac{4}{3}$-competitive
- Grid polygons with holes
  - Lower bound: 2
  - Expl. strategy *CellExplore*
  - $S \leq C + \frac{1}{2}E + 3H + W - 2$

Searching

- Quality measure: search ratio
- Approximation framework
- Applied to simple polygons
- Lower bound for polygons with holes
- Relation between exploration and searching

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

Summary

Onl. exploration of grid polygons

- Simple polygons
  - Lower bound: $\frac{7}{6}$
  - Expl. strategy *SmartDFS*
  - $S \leq C + \frac{1}{2}E - 3$
  - $\frac{4}{3}$-competitive
- Grid polygons with holes
  - Lower bound: 2
  - Expl. strategy *CellExplore*
  - $S \leq C + \frac{1}{2}E + 3H + W - 2$

Searching

- Quality measure: search ratio
- Approximation framework
- Applied to simple polygons
- Lower bound for polygons with holes
- Relation between exploration and searching

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

Summary

Onl. exploration of grid polygons

- Simple polygons
  - Lower bound: $\frac{7}{6}$
  - Expl. strategy *SmartDFS*
  - $S \leq C + \frac{1}{2}E - 3$
  - $\frac{4}{3}$-competitive
- Grid polygons with holes
  - Lower bound: 2
  - Expl. strategy *CellExplore*
  - $S \leq C + \frac{1}{2}E + 3H + W - 2$

Searching

- Quality measure: search ratio
- Approximation framework
- Applied to simple polygons
- Lower bound for polygons with holes
- Relation between exploration and searching

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

Summary

Onl. exploration of grid polygons

- Simple polygons
  - Lower bound: $\frac{7}{6}$
  - Expl. strategy *SmartDFS*
  - $S \leq C + \frac{1}{2}E - 3$
  - $\frac{4}{3}$-competitive
- Grid polygons with holes
  - Lower bound: 2
  - Expl. strategy *CellExplore*
  - $S \leq C + \frac{1}{2}E + 3H + W - 2$

Searching

- Quality measure: search ratio
- Approximation framework
- Applied to simple polygons
- Lower bound for polygons with holes
- Relation between exploration and searching

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

Summary

Onl. exploration of grid polygons

- Simple polygons
  - Lower bound: $\frac{7}{6}$
  - Expl. strategy *SmartDFS*
  - $S \leq C + \frac{1}{2}E - 3$
  - $\frac{4}{3}$-competitive
- Grid polygons with holes
  - Lower bound: 2
  - Expl. strategy *CellExplore*
  - $S \leq C + \frac{1}{2}E + 3H + W - 2$

Searching

- Quality measure: search ratio
- Approximation framework
- Applied to simple polygons
- Lower bound for polygons with holes
- Relation between exploration and searching

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

Summary

Onl. exploration of grid polygons

- Simple polygons
  - Lower bound: $\frac{7}{6}$
  - Expl. strategy *SmartDFS*
  - $S \leq C + \frac{1}{2}E - 3$
  - $\frac{4}{3}$-competitive
- Grid polygons with holes
  - Lower bound: 2
  - Expl. strategy *CellExplore*
  - $S \leq C + \frac{1}{2}E + 3H + W - 2$

Searching

- Quality measure: search ratio
- Approximation framework
- Applied to simple polygons
- Lower bound for polygons with holes
- Relation between exploration and searching

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

Summary

Onl. exploration of grid polygons

- Simple polygons
  - Lower bound: $\frac{7}{6}$
  - Expl. strategy *SmartDFS*
  - $S \leq C + \frac{1}{2}E - 3$
  - $\frac{4}{3}$-competitive
- Grid polygons with holes
  - Lower bound: 2
  - Expl. strategy *CellExplore*
  - $S \leq C + \frac{1}{2}E + 3H + W - 2$

Searching

- Quality measure: search ratio
- Approximation framework
- Applied to simple polygons
- Lower bound for polygons with holes
- Relation between exploration and searching

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

Summary

Onl. exploration of grid polygons

- Simple polygons
  - Lower bound: $\frac{7}{6}$
  - Expl. strategy *SmartDFS*
  - $S \leq C + \frac{1}{2}E - 3$
  - $\frac{4}{3}$-competitive
- Grid polygons with holes
  - Lower bound: 2
  - Expl. strategy *CellExplore*
  - $S \leq C + \frac{1}{2}E + 3H + W - 2$

Searching

- Quality measure: search ratio
- Approximation framework
- Applied to simple polygons
- Lower bound for polygons with holes
- Relation between exploration and searching

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

Summary

Onl. exploration of grid polygons

- Simple polygons
  - Lower bound: $\frac{7}{6}$
  - Expl. strategy *SmartDFS*
  - $S \leq C + \frac{1}{2}E - 3$
  - $\frac{4}{3}$-competitive
- Grid polygons with holes
  - Lower bound: 2
  - Expl. strategy *CellExplore*
  - $S \leq C + \frac{1}{2}E + 3H + W - 2$

Searching

- Quality measure: search ratio
- Approximation framework
- Applied to simple polygons
- Lower bound for polygons with holes
- Relation between exploration and searching

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

Summary

Onl. exploration of grid polygons

- Simple polygons
  - Lower bound: $\frac{7}{6}$
  - Expl. strategy *SmartDFS*
  - $S \leq C + \frac{1}{2}E - 3$
  - $\frac{4}{3}$-competitive
- Grid polygons with holes
  - Lower bound: 2
  - Expl. strategy *CellExplore*
  - $S \leq C + \frac{1}{2}E + 3H + W - 2$

Searching

- Quality measure: search ratio
- Approximation framework
- Applied to simple polygons
- Lower bound for polygons with holes
- Relation between exploration and searching

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

Summary

Onl. exploration of grid polygons

- Simple polygons
  - Lower bound: $\frac{7}{6}$
  - Expl. strategy *SmartDFS*
  - $S \leq C + \frac{1}{2}E - 3$
  - $\frac{4}{3}$-competitive
- Grid polygons with holes
  - Lower bound: 2
  - Expl. strategy *CellExplore*
  - $S \leq C + \frac{1}{2}E + 3H + W - 2$

Searching

- Quality measure: search ratio
- Approximation framework
- Applied to simple polygons
- Lower bound for polygons with holes
- Relation between exploration and searching

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

Summary

Onl. exploration of grid polygons

- Simple polygons
  - Lower bound: $\frac{7}{6}$
  - Expl. strategy *SmartDFS*
  - $S \leq C + \frac{1}{2}E - 3$
  - $\frac{4}{3}$-competitive
- Grid polygons with holes
  - Lower bound: 2
  - Expl. strategy *CellExplore*
  - $S \leq C + \frac{1}{2}E + 3H + W - 2$

Searching

- Quality measure: search ratio
- Approximation framework
- Applied to simple polygons
- Lower bound for polygons with holes
- Relation between exploration and searching

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

Summary

Onl. exploration of grid polygons

- Simple polygons
  - Lower bound: $\frac{7}{6}$
  - Expl. strategy *SmartDFS*
  - $S \leq C + \frac{1}{2}E - 3$
  - $\frac{4}{3}$-competitive
- Grid polygons with holes
  - Lower bound: 2
  - Expl. strategy *CellExplore*
  - $S \leq C + \frac{1}{2}E + 3H + W - 2$

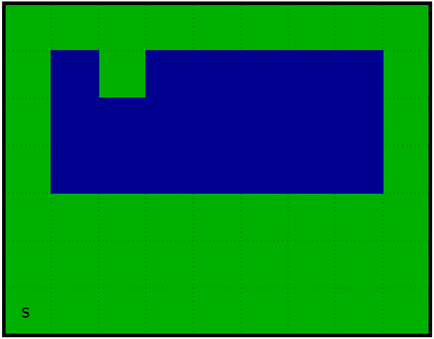
Searching

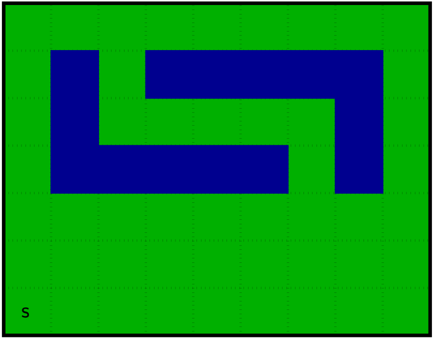
- Quality measure: search ratio
- Approximation framework
- Applied to simple polygons
- Lower bound for polygons with holes
- Relation between exploration and searching

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

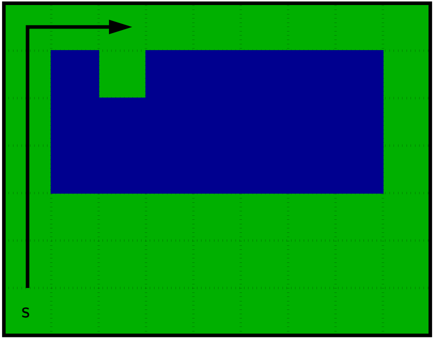
Thank you!

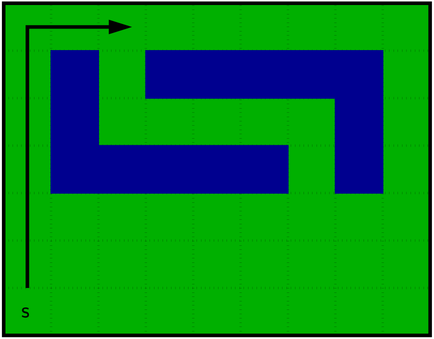
A Problem with SmartDFS



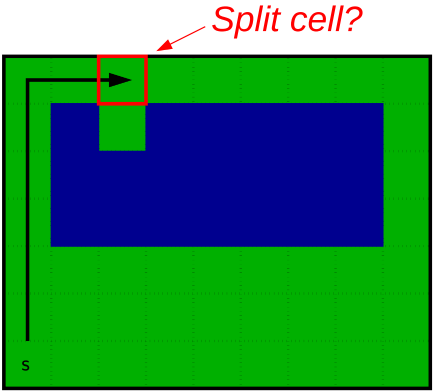


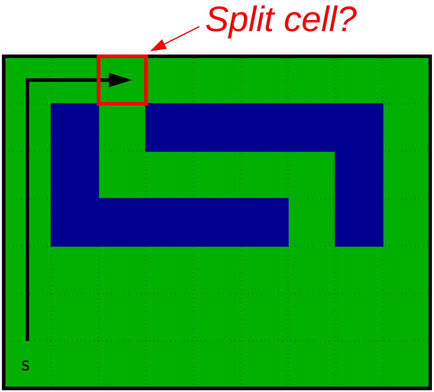
A Problem with SmartDFS



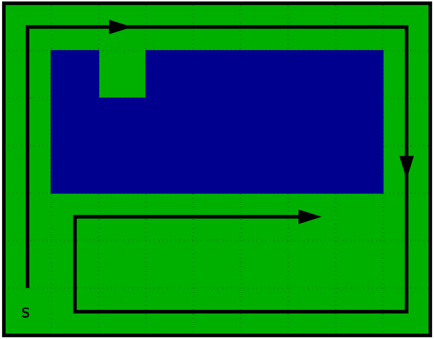


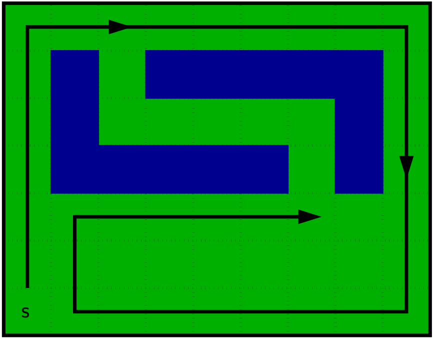
A Problem with SmartDFS



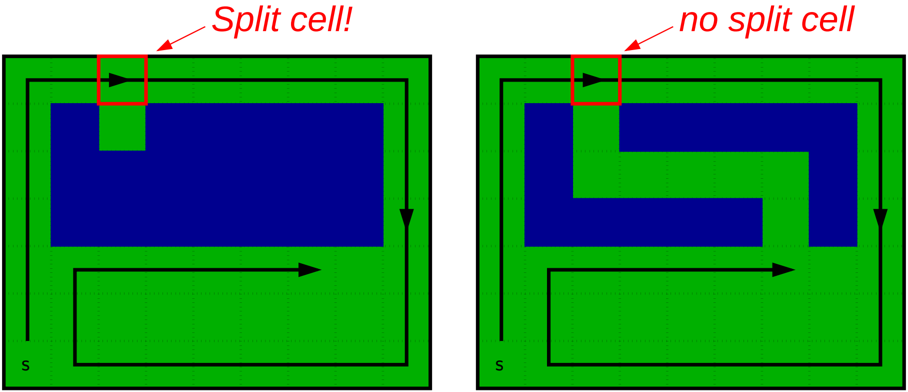


A Problem with SmartDFS





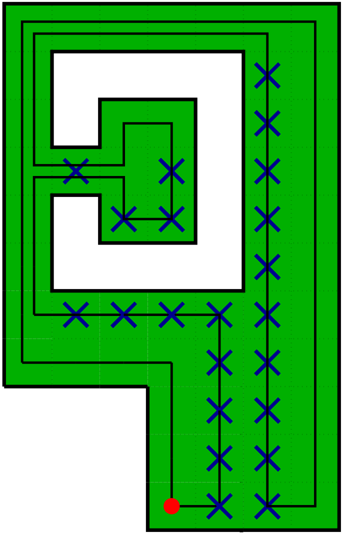
A Problem with SmartDFS



$\Longrightarrow$ No *local* criterion for detecting split cells!

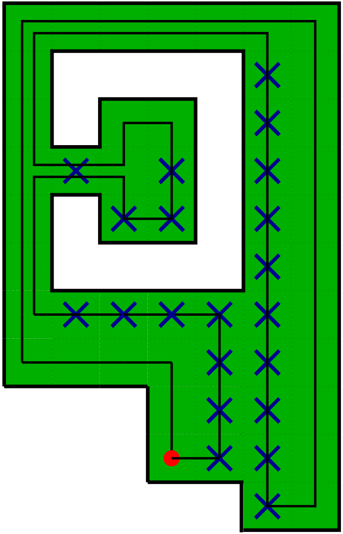
Analyzing technique

- Successively remove start cell and cells reserved in the first step
- Observe the balance of cells, edges, and steps
- Global arguments: charge holes and curves



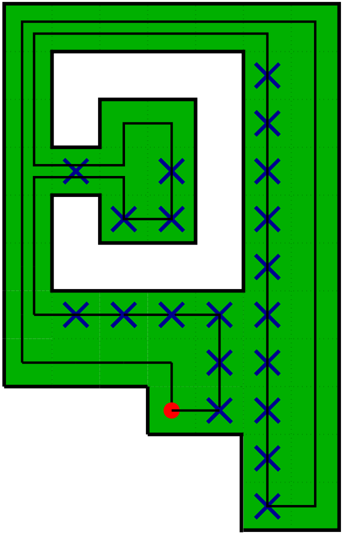
Analyzing technique

- Successively remove start cell and cells reserved in the first step
- Observe the balance of cells, edges, and steps
- Global arguments: charge holes and curves



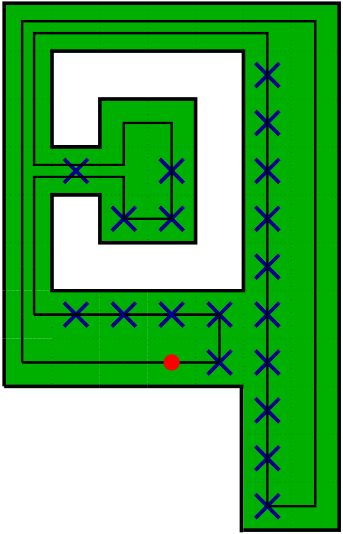
Analyzing technique

- Successively remove start cell and cells reserved in the first step
- Observe the balance of cells, edges, and steps
- Global arguments: charge holes and curves



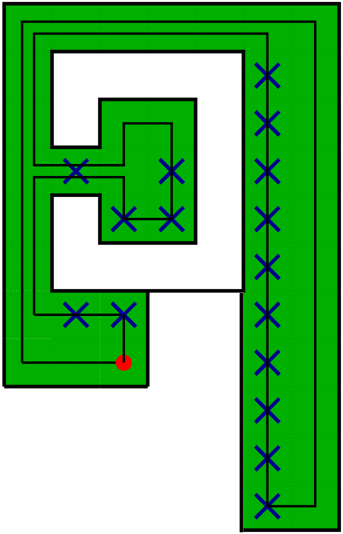
Analyzing technique

- Successively remove start cell and cells reserved in the first step
- Observe the balance of cells, edges, and steps
- Global arguments: charge holes and curves



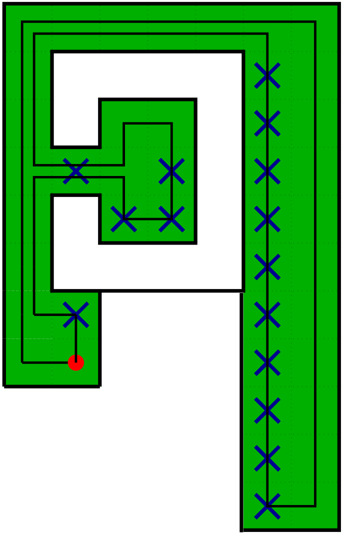
Analyzing technique

- Successively remove start cell and cells reserved in the first step
- Observe the balance of cells, edges, and steps
- Global arguments: charge holes and curves



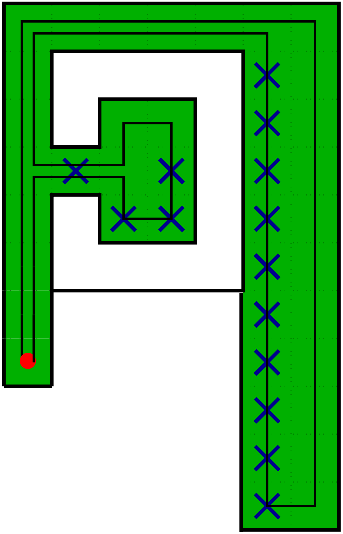
Analyzing technique

- Successively remove start cell and cells reserved in the first step
- Observe the balance of cells, edges, and steps
- Global arguments: charge holes and curves



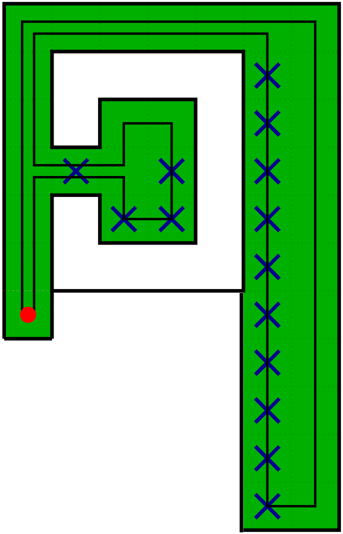
Analyzing technique

- Successively remove start cell and cells reserved in the first step
- Observe the balance of cells, edges, and steps
- Global arguments: charge holes and curves



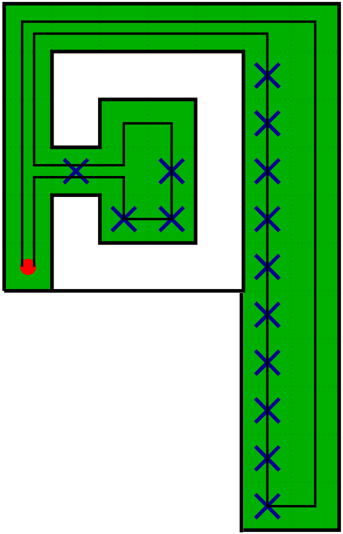
Analyzing technique

- Successively remove start cell and cells reserved in the first step
- Observe the balance of cells, edges, and steps
- Global arguments: charge holes and curves



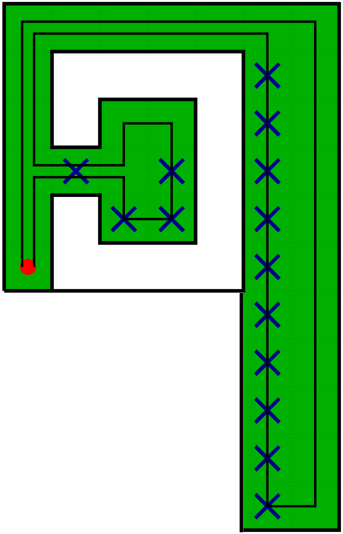
Analyzing technique

- Successively remove start cell and cells reserved in the first step
- Observe the balance of cells, edges, and steps
- Global arguments: charge holes and curves



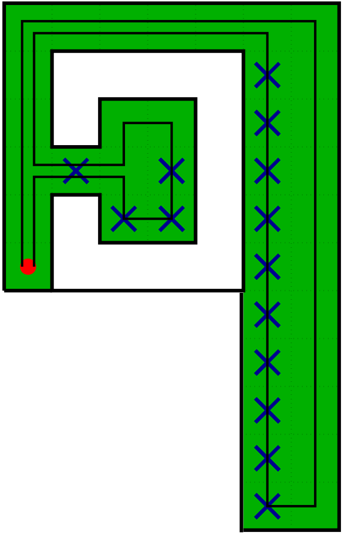
Analyzing technique

- Successively remove start cell and cells reserved in the first step
- Observe the balance of cells, edges, and steps
- Global arguments:
  charge holes and curves



Analyzing technique

- Successively remove start cell and cells reserved in the first step
- Observe the balance of cells, edges, and steps
- Global arguments: charge holes and curves



Performance of CellExplore

Theorem (Number of Steps)

CellExplore needs at most

$$C + \frac{1}{2}E + 3H + W - 2$$

steps to explore a polygon. This bound is tight.

(C: #cells, E: #boundary edges, H: #holes, W: “sinuosity”)

Performance of CellExplore

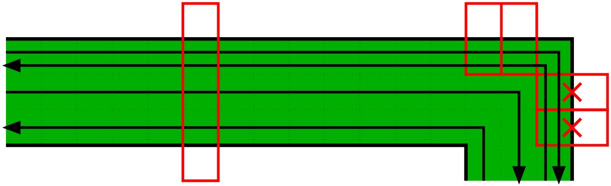
Theorem (Number of Steps)

CellExplore needs at most

$$C + \frac{1}{2}E + 3H + W - 2$$

steps to explore a polygon. This bound is tight.

(C: #cells, E: #boundary edges, H: #holes, W: “sinuosity”)



Quality measure

- A search algorithm $\mathcal{S}$ is called C-competitive, if $\exists A$, so that for every environment:

$$|\mathcal{S}| \leq C \cdot |\mathrm{OPT}| + A$$

- A search algorithm $\mathcal{S}$ is called C–search competitive, if $\exists A$, so that for every environment $\mathcal{E}$:

$$\mathrm{SR}(\mathcal{S}, \mathcal{E}) \leq C \cdot \mathrm{SR}_{\mathrm{OPT}}(\mathcal{E}) + A$$