Models and Algorithms for Online Exploration and Search

Tom Kamphans¹

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

April 04, 2006
Planning a path for an autonomous vehicle

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

Exploration:
Move around, until everything was 'seen'

Searching:
Move around, until target is found
Planning a path for an autonomous vehicle

Exploration:
Move around, until everything was 'seen'

Searching:
Move around, until target is found
Planning a path for an autonomous vehicle

Exploration:
Move around, until everything was 'seen'

Searching:
Move around, until target is found
Planning a path for an autonomous vehicle

Exploration:
Move around, until everything was 'seen'

Searching:
Move around, until target is found
Planning a path for an autonomous vehicle

Exploration:
Move around, until everything was 'seen'

Searching:
Move around, until target is found
Planning a path for an autonomous vehicle

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

Exploration:
Move around, until everything was 'seen'

Searching:
Move around, until target is found
Planning a path for an autonomous vehicle

Exploration:
Move around, until everything was 'seen'

Searching:
Move around, until target is found
Planning a path for an autonomous vehicle

Exploration:
Move around, until everything was 'seen'

Searching:
Move around, until target is found
’Real world’ $\rightarrow$ ’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: $\frac{|ONL|}{|OPT|}$
  - 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: $\frac{|ONL|}{|OPT|}$
  - Other ratios (search ratio)


Models

’Real world’ \(\rightarrow\) ’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’ $\rightarrow$ ’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: $\frac{|ONL|}{|OPT|}$
  - Other ratios (search ratio)
’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)
’Real world’ $\rightarrow$ ’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: $|\text{ONL}|/|\text{OPT}|$
  - Other ratios (search ratio)
’Real world’ $\rightarrow$ ’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)
’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)
’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)
’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)
’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)
'Real world' $\rightarrow$ '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: $\frac{|\text{ONL}|}{|\text{OPT}|}$
  - Other ratios (*search ratio*)
’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)
’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)
’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’ $\rightarrow$ ’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)
'Real world' $\rightarrow$ '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: $\frac{|\text{ONL}|}{|\text{OPT}|}$
  - Other ratios (*search ratio*)
’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)
’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)
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
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
Grid polygon:

- Environment is subdivided by an integer grid
- Simple $\Rightarrow$ No holes

Robot

- No vision
- Can sense 4 adjacent cells
- Can enter adjacent, free cell
Grid polygon:
- Environment is subdivided by an integer grid
- Simple $\Rightarrow$ No holes

Robot
- No vision
- Can sense 4 adjacent cells
- Can enter adjacent, free cell
Grid polygon:
- Environment is subdivided by an integer grid
- Simple $\Rightarrow$ No holes

Robot
- No vision
- Can sense 4 adjacent cells
- Can enter adjacent, free cell
Grid polygon:
- Environment is subdivided by an integer grid
- Simple $\Rightarrow$ 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 $\Rightarrow$ 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 $\Rightarrow$ No holes

Robot
- No vision
- Can sense 4 adjacent cells
- Can enter adjacent, *free* cell
Grid polygon:
- Environment is subdivided by an integer grid
- Simple $\Rightarrow$ No holes

Robot
- No vision
- Can sense 4 adjacent cells
- Can enter adjacent, free cell
Grid polygon:
- Environment is subdivided by an integer grid
- Simple $\Rightarrow$ 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 $\Rightarrow$ 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]
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

\[ s \rightarrow s \]
Proof: Lower Bound

South or east

Diagram showing movement from south to east.
Proof: Lower Bound

Close polygon

\[
\begin{align*}
\text{Step 1: } & \quad \text{Start point } s \\
\text{Step 2: } & \quad \text{Move to endpoint } s \\
\text{Step 3: } & \quad \text{Close the polygon}
\end{align*}
\]
Proof: Lower Bound

Online vs. optimal

8/6
Proof: Lower Bound

3 possibilities:

8/6
3 possibilities: south,
Proof: Lower Bound

3 possibilities: south, east,

\[ \frac{8}{6} \]
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

Tom Kamphans (Uni Bonn)  Models and Algos for Expl. and Search
Proof: Lower Bound

Close polygon

8/6

12/10

12/10

12/10
Proof: Lower Bound

Online vs. optimal

8/6
12/10
12/10
28/24
Proof: Lower Bound

Polygons of arbitrary size

8/6 → 12/10 → 12/10 → 28/24
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!
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!
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.
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.
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).
- 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).
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).
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).
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).
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).
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).
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).
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).
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).
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).
Theorem (Number of Steps)

\[ S \leq C + \frac{1}{2}E - 3 \] (tight!)

(S: #Steps from cell to cell, C: #cells, E: #boundary edges)

Theorem (Competitivity)

SmartDFS is \( \frac{4}{3} \) competitive (i.e., \( S_{\text{SmartDFS}} \leq \frac{4}{3} \cdot S_{\text{Optimal}} \))
**Theorem (Number of Steps)**

\[ S \leq C + \frac{1}{2}E - 3 \]  
(tight!)

(S: #Steps from cell to cell, C: #cells, E: #boundary edges)

**Theorem (Competitiveness)**

SmartDFS is \( \frac{4}{3} \) competitive (i.e., \( S_{\text{SmartDFS}} \leq \frac{4}{3} \cdot S_{\text{Optimal}} \))
http://www.geometrylab.de/Gridrobot/
Theorem

No online exploration strategy achieves a factor better than $\frac{1}{2}$
in grid polygons with holes.
Proof: Lower Bound

- fix large $Q$, observe strategy’s behaviour

\[ S \]

Case 1: robot returns to $s$ after $Q < S < 2Q$ steps
→ 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 $2R$, Strat Opt $→ 2$ for $Q → ∞$
Proof: Lower Bound

- fix large $Q$, observe strategy’s behaviour

![Diagram of corridor with robot returning to $s$ after $Q < S < 2Q$ steps.]

- $s$

- Robot has walked at least $2R - 2$ steps.
- Needs another $2R$ steps to explore the last two cells.
Proof: Lower Bound

- fix large $Q$, observe strategy’s behaviour

$$S$$

Case 1: robot returns to $s$ after $Q < S < 2Q$ steps → 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 $2R$, $Strat_{Opt} \rightarrow 2$ for $Q \rightarrow \infty$
Proof: Lower Bound

- fix large $Q$, observe strategy’s behaviour

$$S$$
Proof: Lower Bound

- fix large $Q$, observe strategy’s behaviour

$S$

Robot has walked at least $2R - 2$ steps.
Needs another $2R$ steps to explore the last two cells.
Optimal $2R$, Strat 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
- 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

![Diagram of corridor with robot movement](image)

- Case 1: robot returns to $s$ after $Q < S < 2Q$ steps
  - 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

$$s$$

$$R$$

- Case 1: robot returns to $s$ after $Q < S < 2Q$ steps
- 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
- → 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 $2R$, $\frac{\text{Strat}_{\text{Opt}}}{\text{Opt}} \rightarrow 2$ for $Q \rightarrow \infty$
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 $Q \rightarrow \infty$
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 $Q \rightarrow \infty$
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 $Q \rightarrow \infty$
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 $Q \rightarrow \infty$
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 $Q \rightarrow \infty$
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 $2R + 2R'$, \[
\frac{\text{Strat}}{\text{Opt}} \rightarrow 2 \quad \text{for} \quad Q \rightarrow \infty
\]
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 $Q \rightarrow \infty$
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 $Q \rightarrow \infty$
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 $Q \rightarrow \infty$
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
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
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
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
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
http://www.geometrylab.de/Gridrobot/
1 Introduction

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

3 Search
Search for a goal in a given environment, $\mathcal{E}$
Quality measure?
**Competitive ratio** for a strategy, $S$:

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

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

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

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

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

**Approximation**: $S$ Search-competitive

$$C_s := \sup_{\mathcal{E}} \frac{SR(S, \mathcal{E})}{SR_{OPT}(\mathcal{E})}$$
- Searching in a polygon
  - Searcher has vision
  - Adversary can force every strategy to explore every corridor
  - Optimal path is very short
  - $\Rightarrow$ every strategy is ’bad’ (i.e., not constant-competitive)
Searching in a polygon
Searcher has vision
Adversary can force every strategy to explore every corridor
Optimal path is very short
$\Rightarrow$ 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
  - $\Rightarrow$ every strategy is 'bad'
    (i.e., not constant-competitive)
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)
- Searching in a polygon
- Searcher has vision
- Adversary can force every strategy to explore every corridor
- Optimal path is very short

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

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

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

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

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

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

- **Approximation**: $S$ Search-competitive

$$C_s := \sup_{\mathcal{E}} \frac{SR(S, \mathcal{E})}{SR_{OPT}(\mathcal{E})}$$
**Quality measure**

- **Competitive ratio** for a strategy, $S$:

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

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

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

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

- **Optimal search ratio**: $SR_{OPT}(\mathcal{E}) := \inf_S SR(S, \mathcal{E})$

- **Approximation**: $S$ Search-competitive

  \[
  C_s := \sup_{\mathcal{E}} \frac{SR(S, \mathcal{E})}{SR_{OPT}(\mathcal{E})}
  \]
Quality measure

- **Competitive ratio** for a strategy, $S$:

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

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

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

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

- **Optimal search ratio**: $SR_{OPT}(\mathcal{E}) := \inf_S SR(S, \mathcal{E})$}

- Approximation: $S$ **Search-competitive**

$$C_s := \sup_{\mathcal{E}} \frac{SR(S, \mathcal{E})}{SR_{OPT}(\mathcal{E})}$$
Quality measure

- **Competitive ratio** for a strategy, $S$:

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

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

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

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

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

- **Approximation**: $S$ **Search-competitive**

$$C_s := \sup_{\mathcal{E}} \frac{SR(S, \mathcal{E})}{SR_{\text{OPT}}(\mathcal{E})}$$
Depth-Restrictable Exploration

Definition

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

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

\[
|\text{Expl}(d)| \leq C \cdot |\text{Expl}_{\text{opt}}(\beta \cdot d)|.
\]
Approximation Framework

Approximation Strategy

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

Theorem

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

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

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

($\beta$: enlargement factor for depth restriction)
Approximation Framework

Approximation Strategy

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

Theorem

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

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

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

($\beta$: enlargement factor for depth restriction)
Shortest Watchman Route (Dror et al., 2003) ⇒ offline $8$–search-competitive strategy

$\sqrt{2}$-competitive exploration for rectilinear polygons (Deng et al., 1991) ⇒ $8\sqrt{2}$–search-competitive online strategy for rectilinear polygons

26.5-competitive exploration strategy PolyExplore (Hoffmann et al., 1998) ⇒ 212–search-competitive online strategy for simple polygons
Searching in Simple Polygons

- Shortest Watchman Route (Dror et al., 2003)
  ⇒ offline $8$–search-competitive strategy

- $\sqrt{2}$-competitive exploration for rectilinear polygons (Deng et al., 1991)
  ⇒ $8\sqrt{2}$–search-competitive online strategy for rectilinear polygons

- 26.5-competitive exploration strategy PolyExplore (Hoffmann et al., 1998)
  ⇒ 212–search-competitive online strategy for simple polygons
Searching in Simple Polygons

- Shortest Watchman Route (Dror et al., 2003) ⇒ offline 8–search-competitive strategy

- $\sqrt{2}$-competitive exploration for rectilinear polygons (Deng et al., 1991) ⇒ $8\sqrt{2}$–search-competitive online strategy for rectilinear polygons

- 26.5-competitive exploration strategy PolyExplore (Hoffmann et al., 1998) ⇒ 212–search-competitive online strategy for simple polygons
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
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 $\Omega(k)$
  \[ \Rightarrow \text{No search-competitive strategy} \]
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
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
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
- 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
Theorem

If for a given type of environments

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

⇒ there is no search-competitive strategy.
Close relation

- \exists \text{ constant-competitive, depth-restrictable exploration strategy}
  \Rightarrow \exists \text{ search-competitive strategy}

- \not\exists \text{ constant-competitive exploration strategy, but } \exists \text{'extendable’ lower bound}
  \Rightarrow \not\exists \text{ search-competitive strategy}

Open question

\exists \text{ search-competitive strategy}

\iff \exists \text{ constant-competitive exploration strategy (for environments fulfilling } \forall p \in \mathcal{E} : |sp(s, p)| = |sp(p, s)|\)
### Close relation

- ∃ constant-competitive, depth-restrictable exploration strategy
  ⇒ ∃ search-competitive strategy
- ∄ constant-competitive exploration strategy, but ∃ 'extendable' lower bound
  ⇒ ∄ search-competitive strategy

### Open question

∃ search-competitive strategy

? ⇔ ∃ constant-competitive exploration strategy

(for environments fulfilling ∀p ∈ E : |sp(s, p)| = |sp(p, s)|)
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/
Onl. exploration of grid polygons

Simple polygons
- Lower bound: \( \frac{7}{6} \)
- Expl. strategy \textit{SmartDFS}
- \( S \leq C + \frac{1}{2}E - 3 \)
- \( \frac{4}{3} \)-competitive

Grid polygons with holes
- Lower bound: 2
- Expl. strategy \textit{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/
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/
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 with holes
- Lower bound for polygons with holes
- Relation between exploration and searching

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

Tom Kamphans (Uni Bonn)
A Problem with SmartDFS

Split cell?\[⇒\] No local criterion for detecting split cells!
A Problem with SmartDFS

Split cell?

Split cell?

No local criterion for detecting split cells!
A Problem with SmartDFS

Split cell?

No local criterion for detecting split cells!
A Problem with SmartDFS

Split cell!

no split cell

⇒ No local criterion for detecting split cells!
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
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
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
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
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
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”)
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”)
A search algorithm $S$ is called $C$-competitive, if $\exists A$, so that for every environment:

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

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

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