

Online Motion Planning MA-INF 1314

Introduction

Elmar Langetepe
University of Bonn

Einordnung: Bewegungsplanung mobiler Agenten

- Lecture: Tuesday/Thursday 12:30 to 14:00
- Exercise groups: Starting next week 19/20th
Tuesday: 10-12 Wednesday: 14-16
- Sign in now
- Manuscript on the webpage
- Slides on the webpage
- Old manuscript (in german)?
- Exercises
- Today: Short intro, different topics, some examples

Motion planning categories

- Elektronische devices
- Mechanical devices
- Control/Process engineering
- Artificial Intelligence
- Softwareengineering
- ⋮
- Plans: **Algorithmic**
- Input: Geometry of the Environment

Geometric Algorithms

- Geometry of Agent and Environment
- Solve motion planning problem
- Main difference: **Online**/Offline
- Incomplete Information
- **Correctnes?** **Efficiency?**
- **Lower/Upper Bound**
- Structural Properties
- Groundtasks
- Algorithms paradigms
- Formal proofs

Example I: Simple polygon, exploration

- Simple room, no obstacles
- Agent starts at s , sees everything, returns to the start
- 360 degree vision
- Modell: Point-shaped agent, simple polygons, visibility polygon,
- Optimal route: Shortest Watchman Route
- Offline: Polygon is fully known
- Online: Union of visibility polygons
- Offline: Optimal algorithm
- Applets: Offline algorithm versus Online algorithm

Example I: Online/Offline

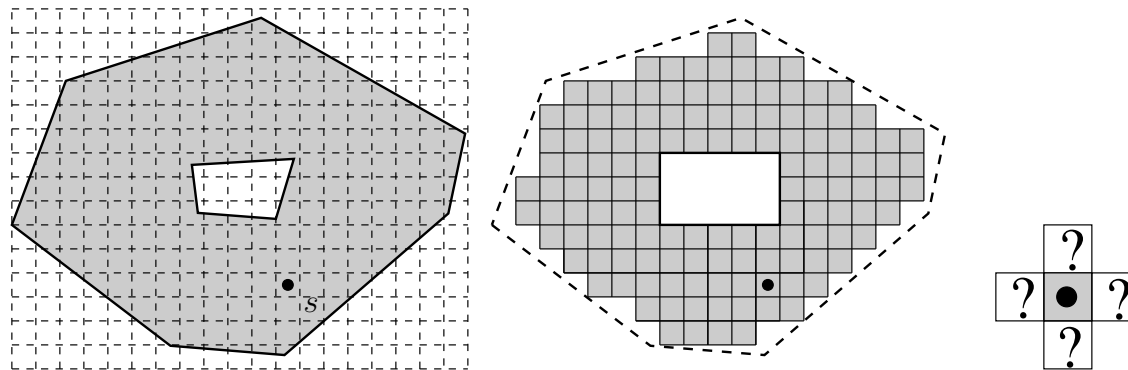
- Axisparallel polygon■
- **Offline Algorithm:** $O(n)$ Applet!■
- Online, not optimal!■
- **Greedy Online Strategy!**■
- Proof: L_1 -optimal!■
- $\sqrt{2}$ Approximation!■
- **Competitive Ratio**■

Online motion planning: Modell

- Searching for a goal
- Exploration of an environment
- Process task on (subset of) environment
- Escape from a labyrinth
- Continuous/discrete vision
- Touch sensor/compass
- Build a map
- History: Simple \Rightarrow complicated

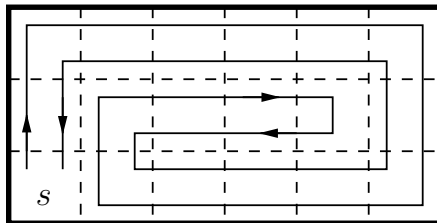
Example II: Exploration of grid-environments

- Grid world (cells), tool of cell-size, one-step from cell to cell
- Process any cell in the connected-component
- Touchsensor only detects neighboring cells
- Build a map, visit all cells, return to start
- Grid-explore applet

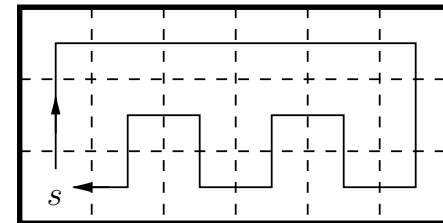


Example: Exploration of grid-environments

- Online DFS for cells \Rightarrow : **Strategy**
- Visit all connected cells: **Correctness**
- Number of steps: $2 \times (C - 1)$
- Optimum: At least C
- Ratio: smaller than 2 times the optimum, **Performance**
- No better strategy! **Lower Bound!!**
- Provoke detour! Proof of this later!!!



DFS



Optimal

Competitive analysis, competitive ratio

Definition 1: Let Π be a problem class and S be a strategy, that solves any instance $P \in \Pi$.

Let $K_S(P)$ be the cost of S for solving P .

Let $K_{\text{OPT}}(P)$ be the cost of the optimal solution for P .

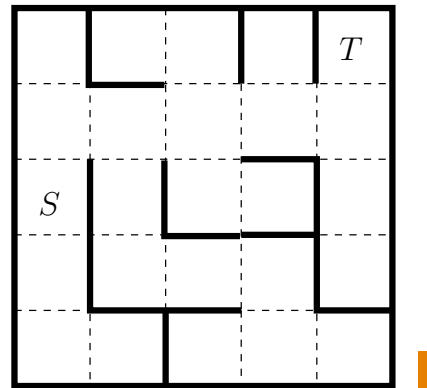
The strategy S is denoted to be **c -competitive**, if there are fixed constants $c, \alpha > 0$, so that for all $P \in \Pi$

$$K_S(P) \leq c \cdot K_{\text{OPT}}(P) + \alpha$$

holds.

Chap. 1: Labyrinths, Grids, Graphs

- Different 2D environments
- Tasks: Searching for a goal, escape, exploration
- Def: **Labyrinth** L intuitiv: Divide the plane by walls into corridors
- Def: **Grid-Labyrinth** grid/cell environment with walls on the edges
- History 1950: Shannon 5×5 Labyrinth with an elektr. Mouse



Chap. 1.1 Shannons Mouse Alg.

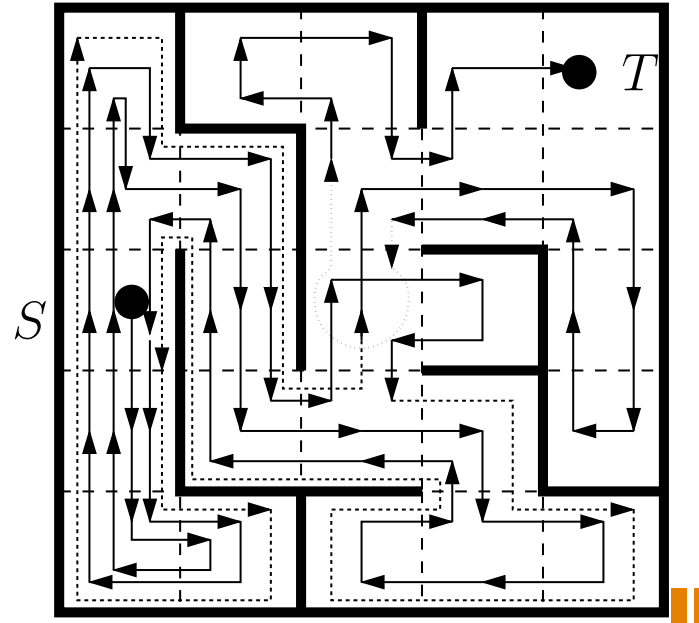
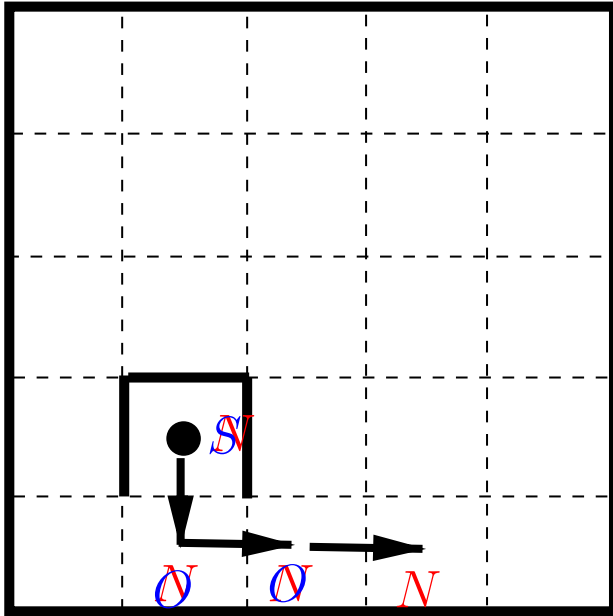
Historie 1950: Shannon 5×5 Labyrinth with an elektr. Mouse ■



- For any cell, there is a marker that stores the direction (N,E,S,W) in which the mouse left the cell at the last visit.
- Initialize any cell with marker N (north)
- While T has not been visited:
Search for the first free cell in clockwise order starting from the current marker direction.
Change the marker correspondingly and enter the corresponding cell.



Shannons Maus Alg.



Shannons Mouse: Correctness!

Theorem 1.1: Shannons Mouse Algorithm will always find the target in any grid labyrinth from any starting point, if a path from S to T exists.

Proof: ■

- Consider the strategy without the target ■
- Show: All reachable cells will be visited infinitely often ■
- **Formal proof:** Blackboard! ■

Efficient algorithm for graph-exploration

- Graph-exploration, visit all edges (and vertices) ■
- Vertex: Outgoing edges become visible! Build the full graph, return to the start! ■
- Visited edges can be located ■
- Strategy: Online-DFS for the edges, ■ visits any edges twice ■

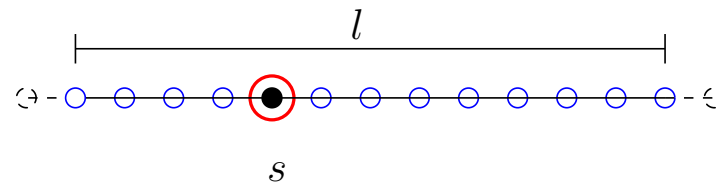
Theorem: Exploring an unknown graph requires roughly twice as many edge visits than the optimal exploration route for the known graph. DFS requires no more than twice as many edges. ■

Formal proof! Second part is already clear! ■ Lower bound by worst-case adversary strategy ■

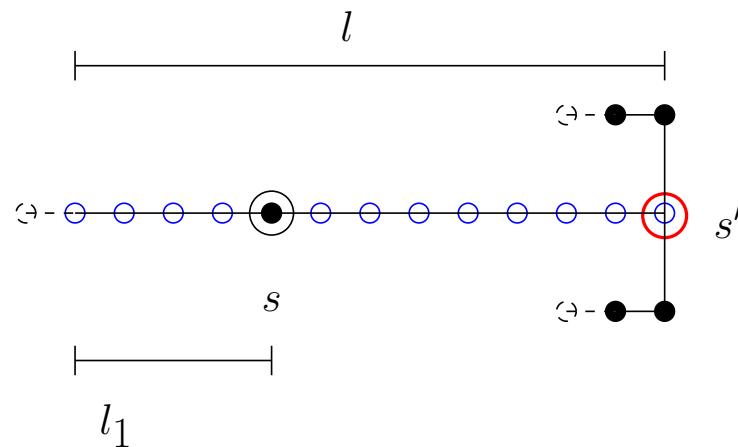
Graphexploration, Edge visits, Adversary

Adversary: $2 - \delta$ worse than the optimum

corridor, agent from s has explored l vertices

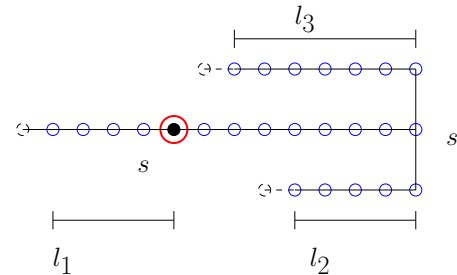


Now bifurcation at s'

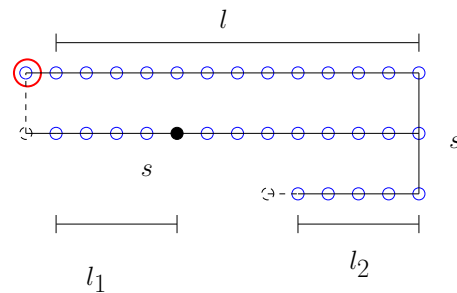


Graphexploration, Edge visits, Adversary

- Case 1: Agent returns to s



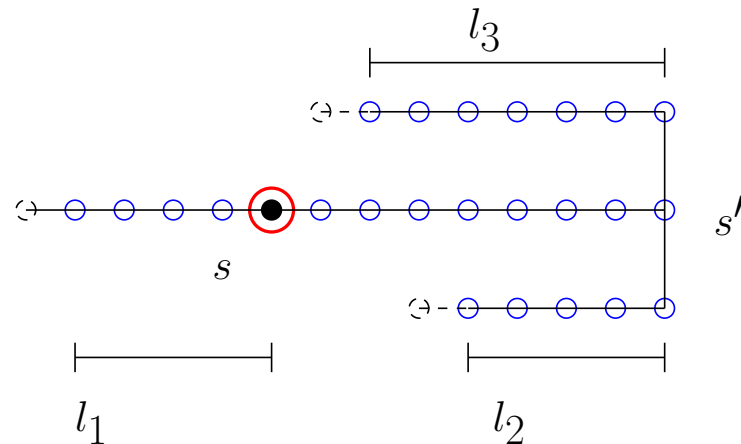
- Case 2: Agent reaches the end of the start corridor



- Close the corridors, present the scene

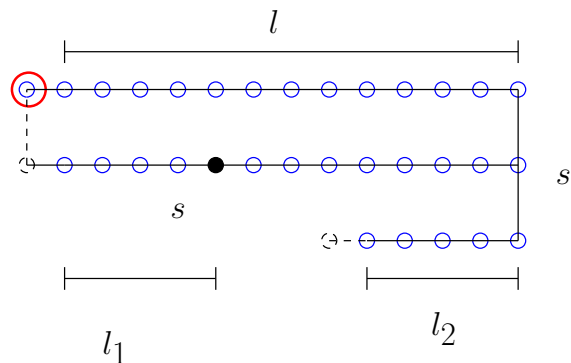
Adversary analysis: Case 1.

- Strategy S_{ROB} against S_{OPT} ■
- Case 1: Agent returns to s ■
- Up to now $|S_{\text{ROB}}| \geq 2l_1 + 2l_2 + 2l_3 + 2(l - l_1)$ ■ $= 2(l + l_2 + l_3)$ ■
- From now on: $2(l + l_2 + l_3) + 6$ ■
- Optimal: $|S_{\text{OPT}}| = 2(l + l_2 + l_3) + 6$ ■
- $|S_{\text{ROB}}| \geq (2 - \delta)|S_{\text{OPT}}|$ for $l \geq \lceil \frac{3}{\delta} \rceil$ ■



Adversary analysis: Case 2.

- Case 2: Agent reaches the end of start corridor
- Up to now: $|S_{\text{ROB}}| \geq 2l_1 + l - l_1 + 2l_2 + l + 1$
- From now on: $l + 1 + 2(l_2 + 1) + l - l_1$
- Sum: $|S_{\text{ROB}}| = 4(l + l_2) + 4$
- Optimal: $|S_{\text{OPT}}| = 2(l + 1 + l_2 + 1) = 2(l + l_2) + 4$
- $|S_{\text{ROB}}| \geq (2 - \delta)|S_{\text{OPT}}|$ für $l \geq \lceil \frac{3}{\delta} \rceil$



Remarks

Corollary : DFS for Online-Exploration of graphs is 2-competitive, there is no C -competitive strategy with $C < 2$.

- DFS is optimal
- Additive constant: Start situation
- Example: Goal very close
- Lower bounds only with arbitrary large examples

Remarks!

- Return to the target■
- No return: **Exercise**■
- Examples: Opt. Tour visits any edge twice■
- Visit only the vertices, DFS for vertices? **Exercise!**■