COSC76/276 Artificial Intelligence
Fall 2022
Informed Search & Deriving Heuristics

Soroush Vosoughi
Computer Science
Dartmouth College
Soroush@Dartmouth.edu

Reminders

- Thank you for participating in Slack and joining office hours!
  - Please continue to` feel free to reach out
- SA-2 due today Oct 1st 11:59pm ET (completion based)
- PA-1 due Sep 29 at 11:59pm ET (1 day blanket extension for everyone_
- PA-2 will be out today. Due Oct 7th at 11:59 ET.

Recap: Search problem

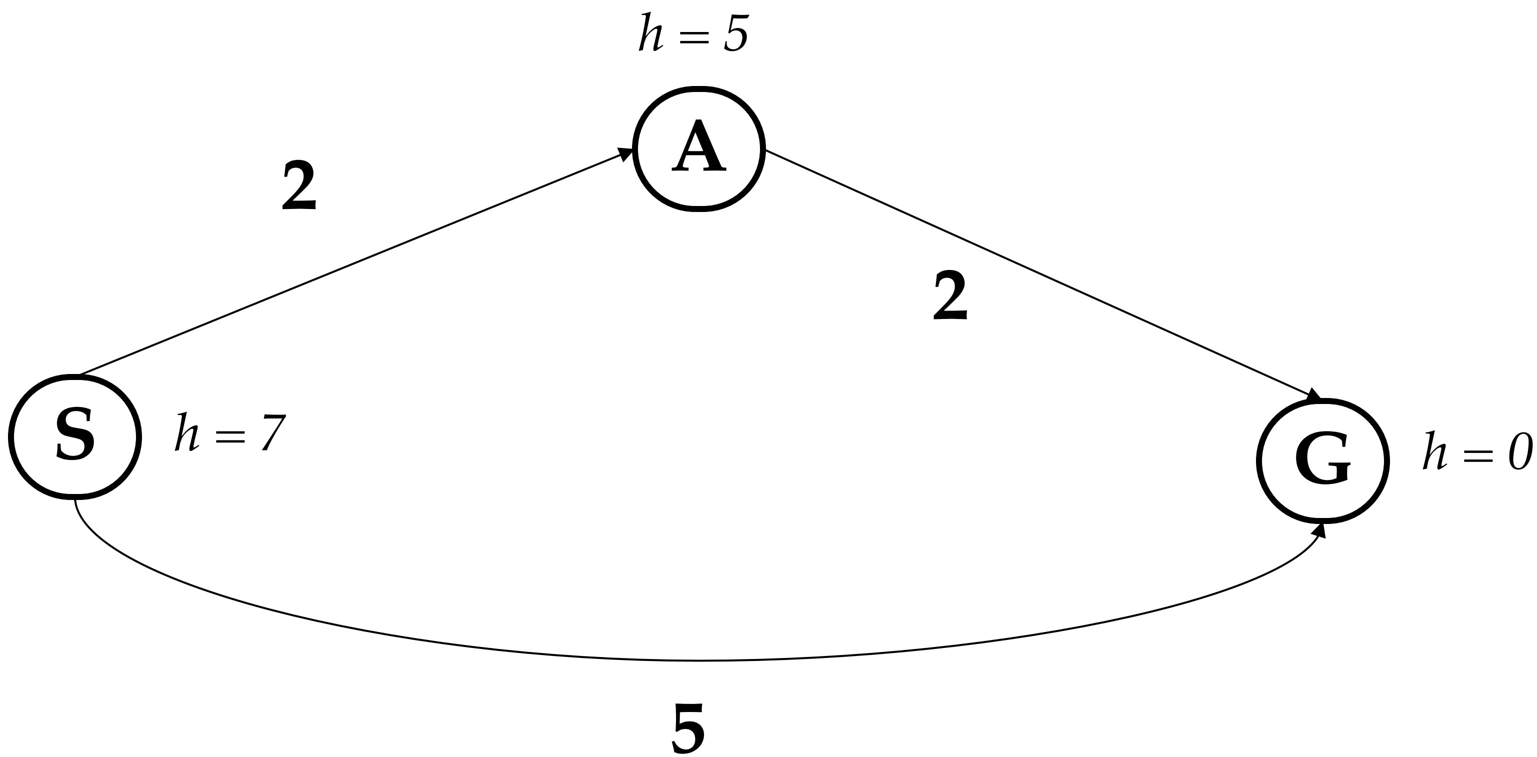
- Uninformed search
  - BFS, DFS, variants, and their properties
  - Do not consider any information about the goal
  - Uninformed search with cost: UCS
- Informed search
  - Heuristic
  - Greedy and A*

A* search

- Combine UCS and greedy
- Evaluation function (cost + heuristic)
  - $f(n) = g(n)+h(n)$

Goes to the goal but backtracks as needed.

Optimality of A*



- Which solution will be found by A*?
  - S,G, with cost 5 instead of S,A,G with cost 4, because of overestimated heuristic

Optimality of A* for tree-search

- A* *tree* search produces shortest paths if the heuristic is **optimistic** (also called **admissible**): it underestimates cost of path to goal from any node on the tree.

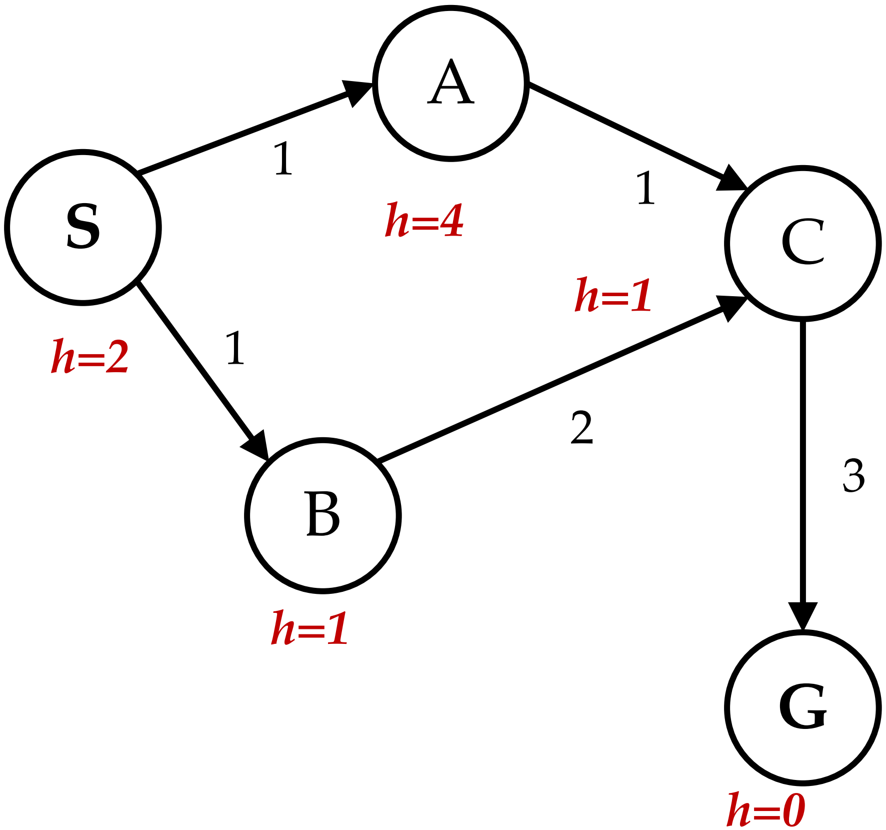
$$0 <= h(n) <= h^*(n)$$

where $h^*(n)$ is the true cost from node n.

Admissible (optimistic) heuristics slow down bad plans but never outweigh true costs

Can you think of an example?

Optimality of A* (graph-search)



- **Priority queue:**
- S2
- (pop S2, push B2, A5)
- B2 A5
- (pop B2, push C4)
- C4 A5
- (pop C4, push G6)
- A5, G6
- (pop A5, C3 not pushed as already explored)
- G6
- (pop G6)

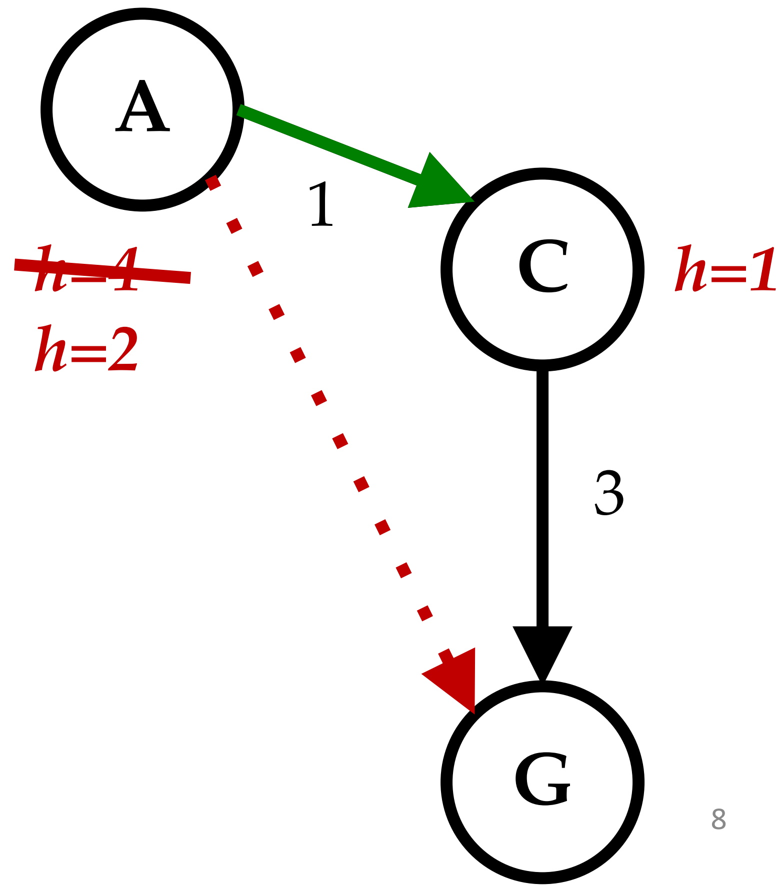
Solution: S,B,C,G with cost 6, instead of S,A,C,G with cost 5

```
for each child of current_state:
   if child not in explored:
     add child to explored
     pack child state into a node, with backpointer to current_node
     add the node to the frontier
   else if child is in frontier with higher f
     replace that frontier node with child node
```

Optimality of A* for graph-search

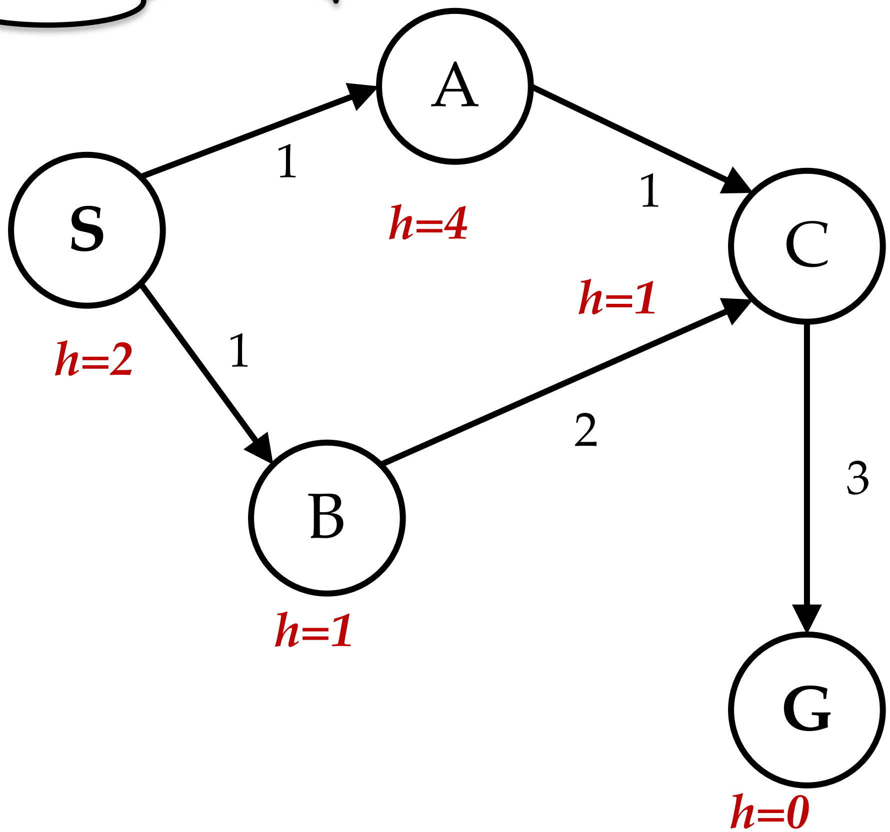
- A* search on a graph produces shortest paths if the heuristic is **consistent** (also called **monotone**).

$$h(n) <= c(n, n') + h(n')$$



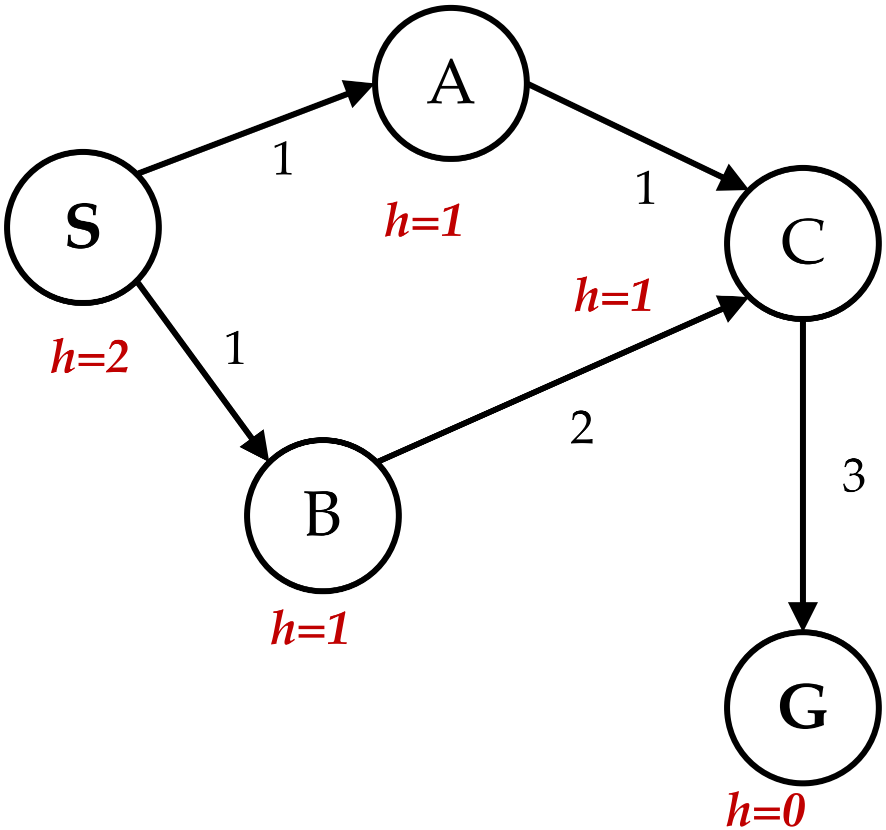
Discussion. Which heuristic value and how should be changed?

Optimality of A* (graph-search)




```
for each child of current_state:
   if child not in explored:
     add child to explored
     pack child state into a node, with backpointer to current_node
     add the node to the frontier
   else if child is in frontier with higher f
     replace that frontier node with child node
```

Optimality of A* (graph-search)



With consistent heuristic

- **Priority queue:**
- S2
- (pop S2, push B2, A2)
- B2 A2
- (pop B2, push C4)
- A2 C4
- (pop A2, push C3, C4 marked as to be removed)
- C3, C4
- (pop C3, push G5)
- G5
- (pop G5)

Solution: S,A,C,G with cost 5,

```
for each child of current_state:
   if child not in explored:
     add child to explored
     pack child state into a node, with backpointer to current_node
     add the node to the frontier
   else if child is in frontier with higher f
     replace that frontier node with child node
```

Properties of A*

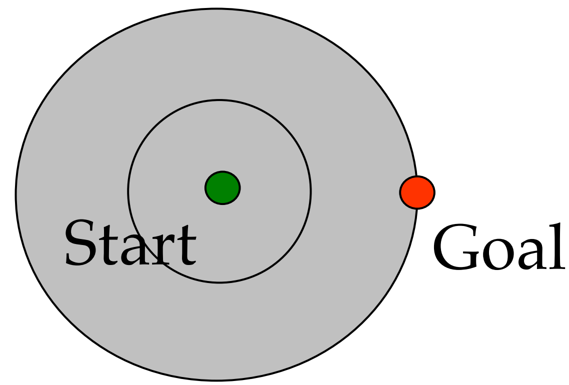
- Space - Keeps all nodes in memory
- Optimal – Yes*
- Complete – Yes
- Time – b^d Worst case scenario

Properties of A*

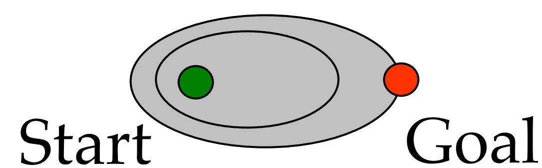
- Space - Keeps all nodes in memory
- Optimal – Yes*
- Complete – Yes
- Time – In practice, depends on goodness of *h.* A good heuristic reduces the effective branding factor: b times the relative error in ($(h^*-h)/h^*$)

Comparison with UCS and A*

- Uniform-cost expands equally in all "directions"



- A* expands mainly toward the goal, but with some limits to ensure optimality



A* applications

- Video games
- Pathing / routing problems
- Resource planning problems
- Robot motion planning
- …

Summary

- To include cost: Uniform cost search
  - Use priority queue
- To include information about the goal: use heuristic
  - Greedy search
  - A*: combines both UCS and greedy
- Heuristic should be admissible and consistent to guarantee optimality for tree and graph search, respectively

Next

- Deriving heuristics

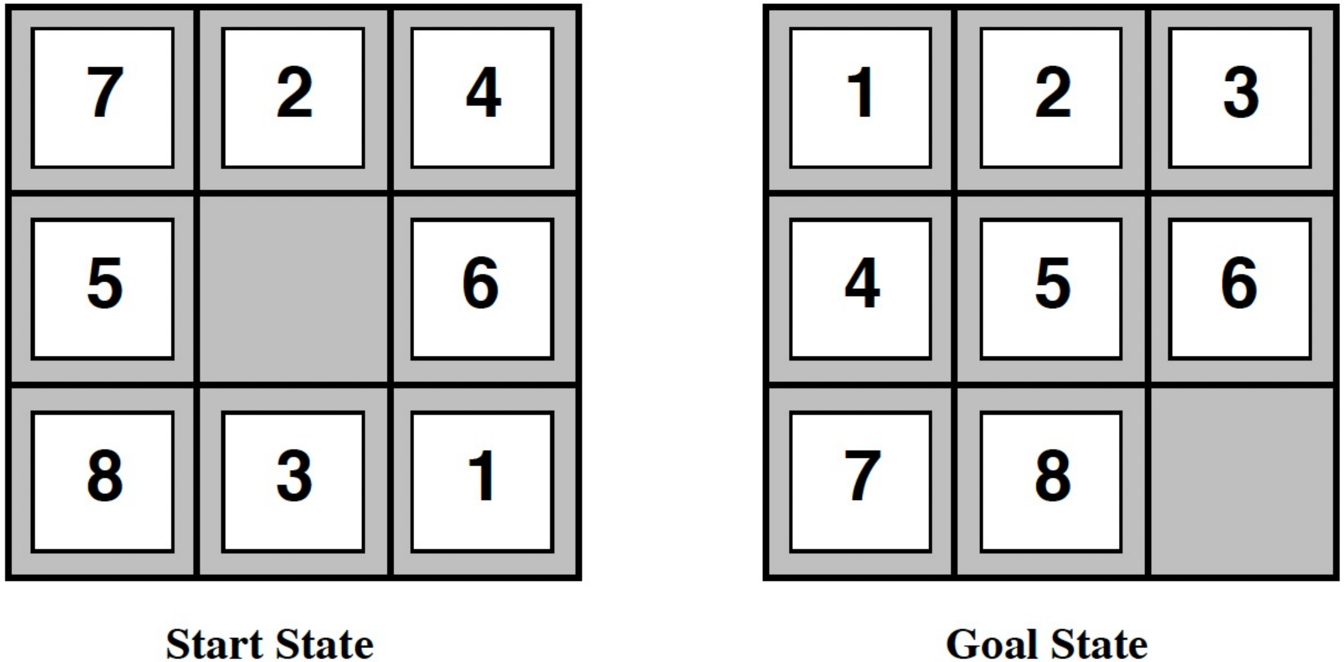
Deriving heuristics

- Most of the work in solving hard search problems optimally is in coming up with admissible heuristics

- Often, admissible heuristics are solutions to *relaxed problems,* where the problem has fewer restrictions.

  -The relaxed problem may have better solutions as there might be shortcuts

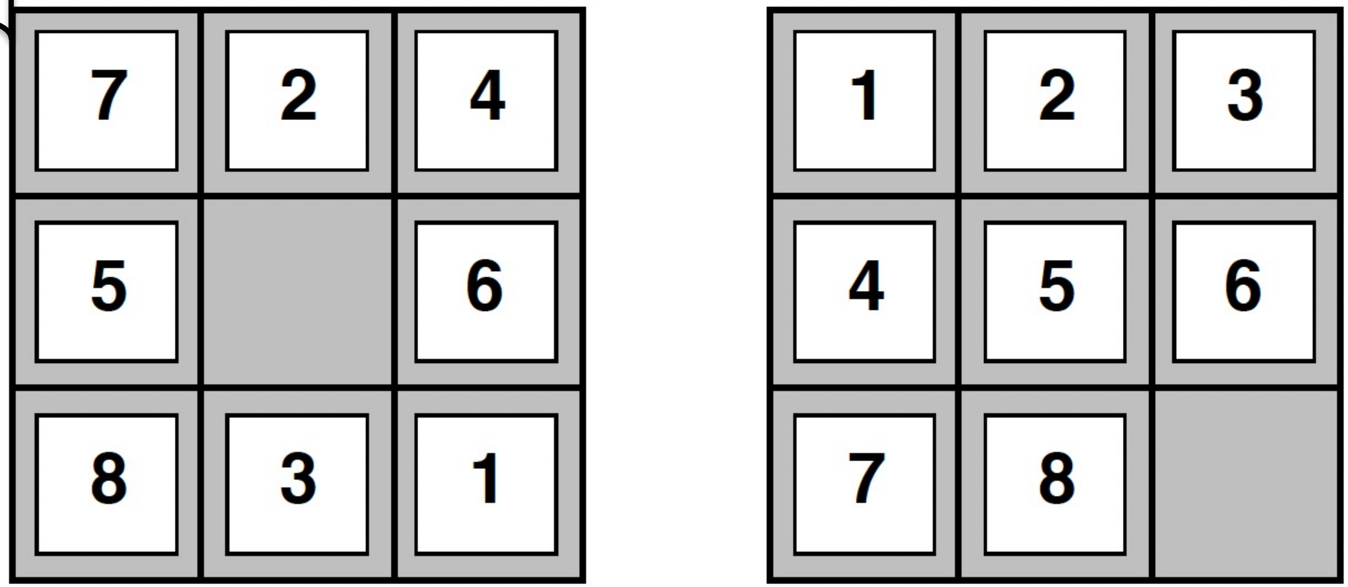
Example: 8-puzzle



- State?
- Action?
- Goal test?
- Action cost?

# of distinct states reachable?

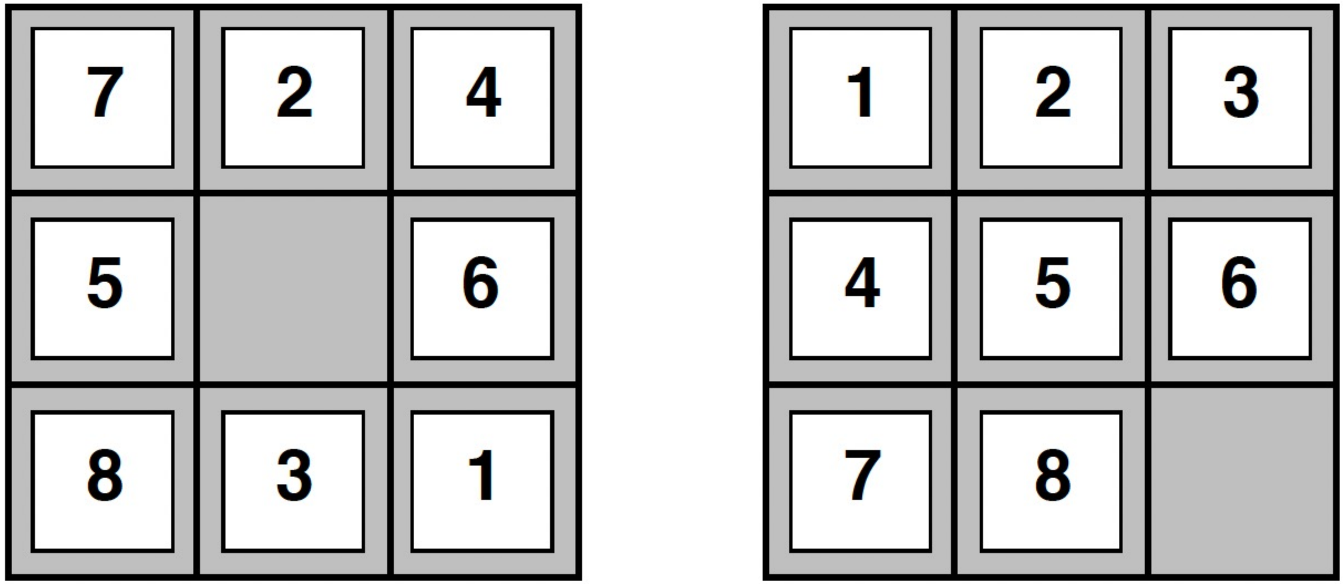
Example: 8-puzzle



Start State Goal State

- State: integer locations of tiles
- Action: move blank tile in an adjacent location in the grid
- Goal test: check with given goal state
- Action cost: 1 per move

Example: 8-puzzle



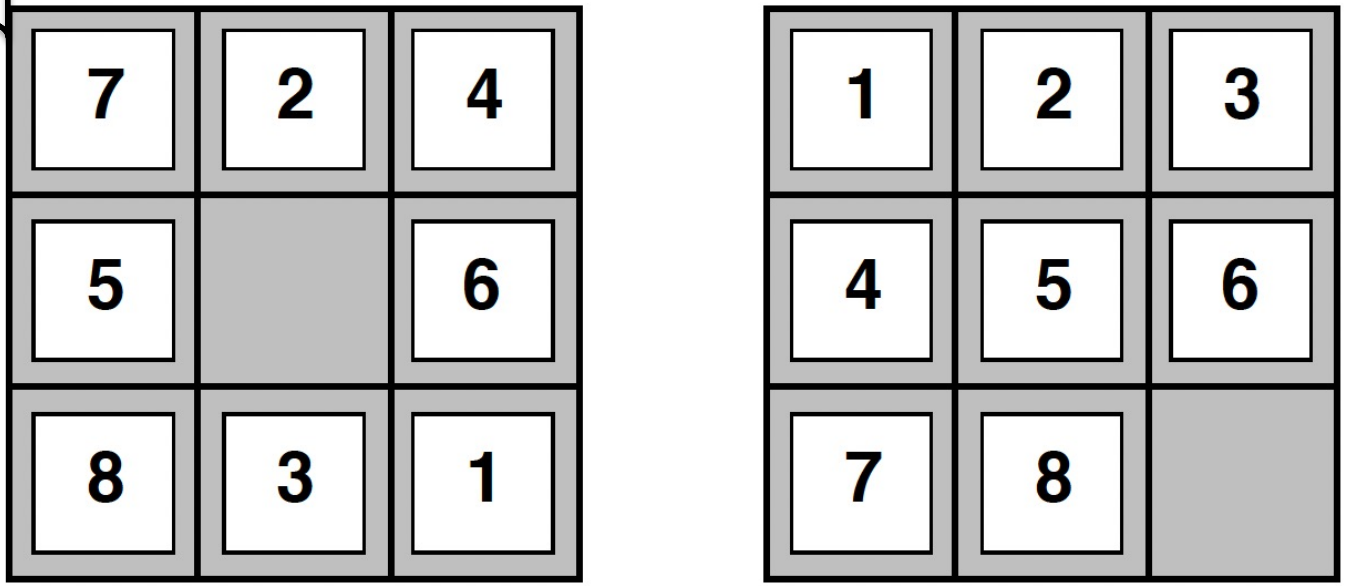
Start State	Goal State

- State: integer locations of tiles
- Action: move blank tile in an adjacent location in the grid
- Goal test: check with given goal state
- Action cost: 1 per move

- Average branching factor?

Can you think of a heuristic?

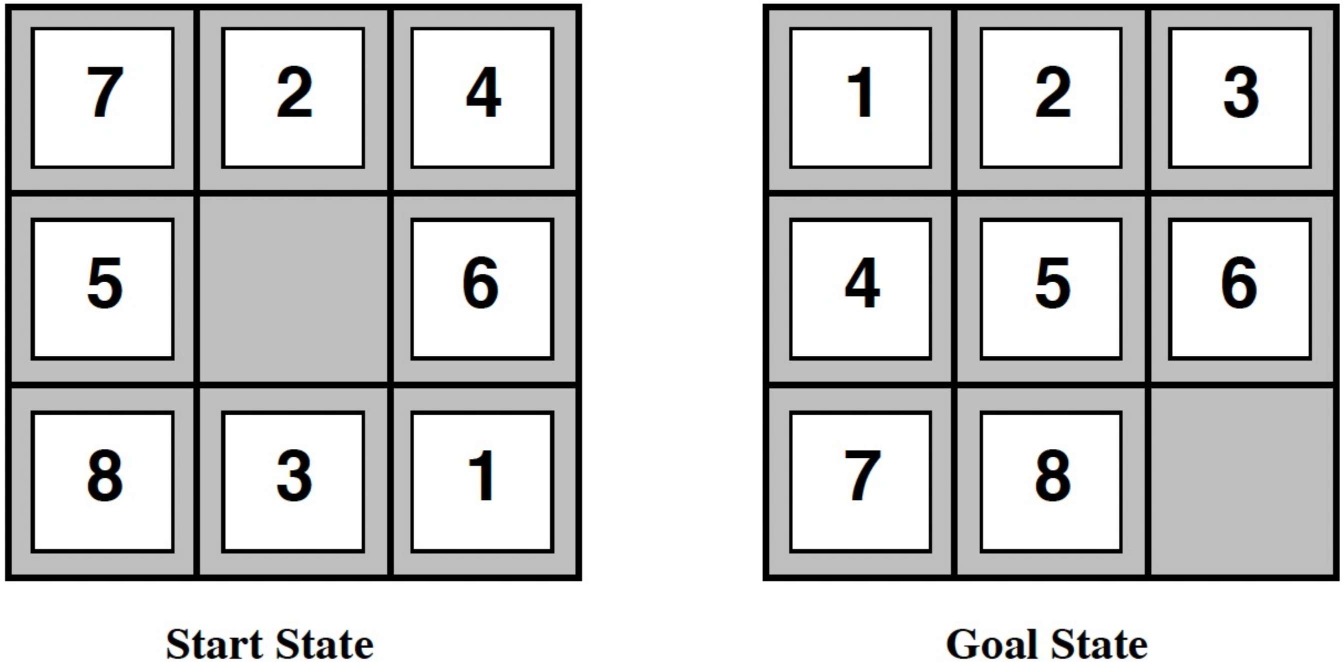
Example: 8-puzzle



Start State **Goal State**

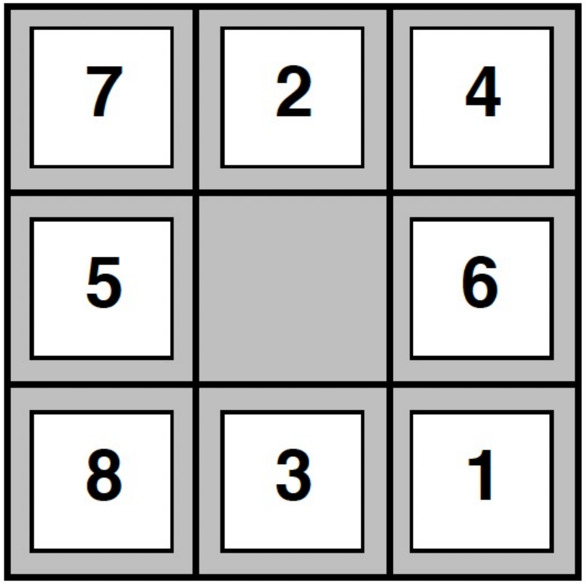
- Heuristics?

Example: 8-puzzle

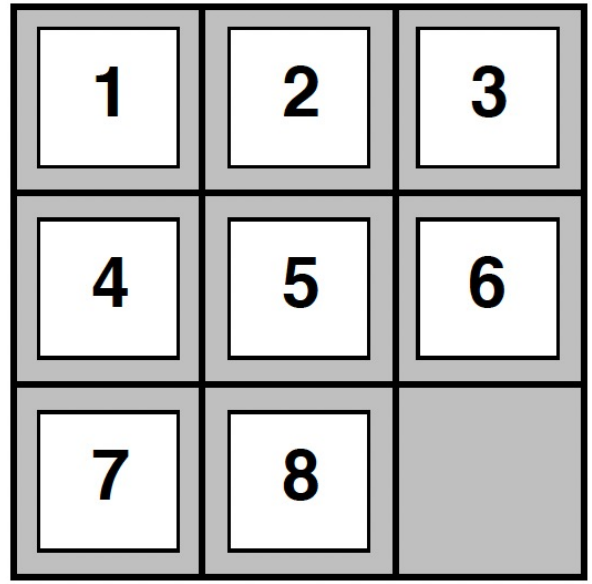


- h1(n) = number of misplaced tiles
  - A tile can “teleport” from square A to square B.

Example: 8-puzzle



Start State



Goal State

- h1(n) = number of misplaced tiles
  - A tile can “teleport” from square A to square B.
- h2(n) = total Manhattan distance
  - A tile can move from square A to square B if A is adjacent to B regardless of whether it is blank or not.

Example: 8-puzzle

7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

- h1(n) = 6
- h2(n) = 4+0+3+3+1+0+2+1

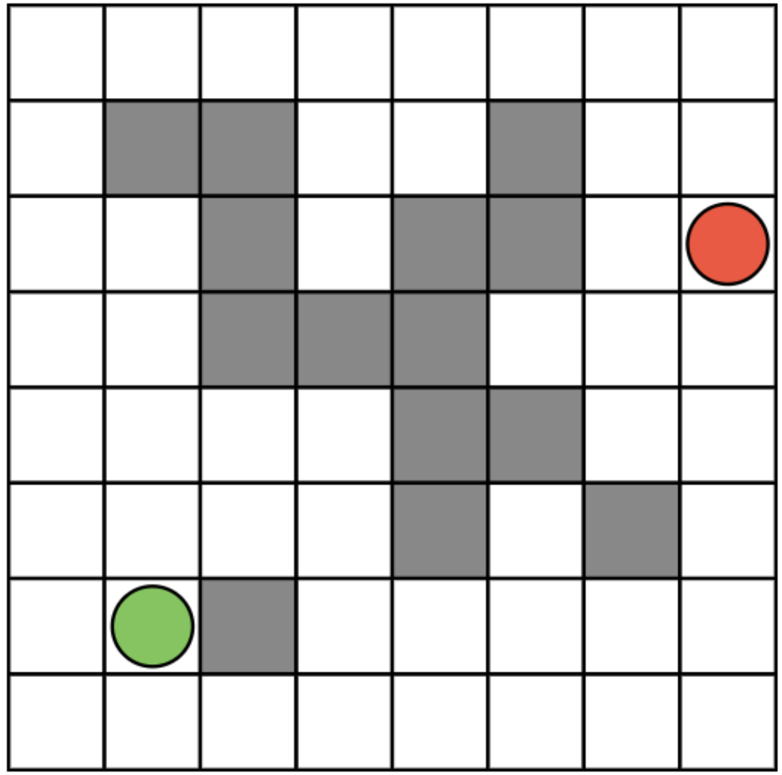
Dominance of heuristics

- If h2(n) >= h1(n) for all n (both admissible) then h2 dominates h1 and is better for search

- If the relation between heuristics is not known, given any admissible heuristics, take the max

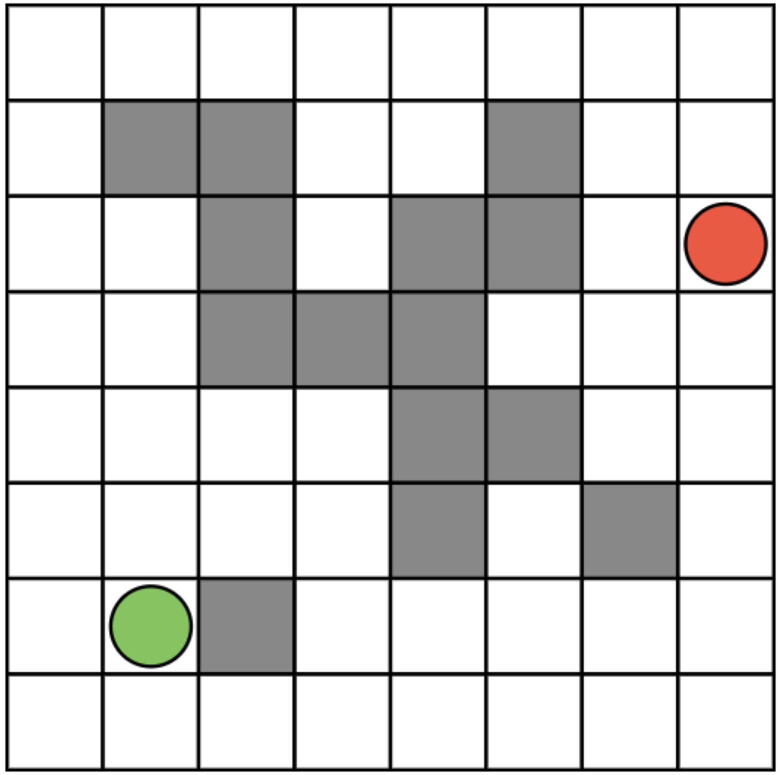
Heuristics? Write down on paper

PA2: Mazeworld problem



Robot can move N, S, E, W onto empty squares.

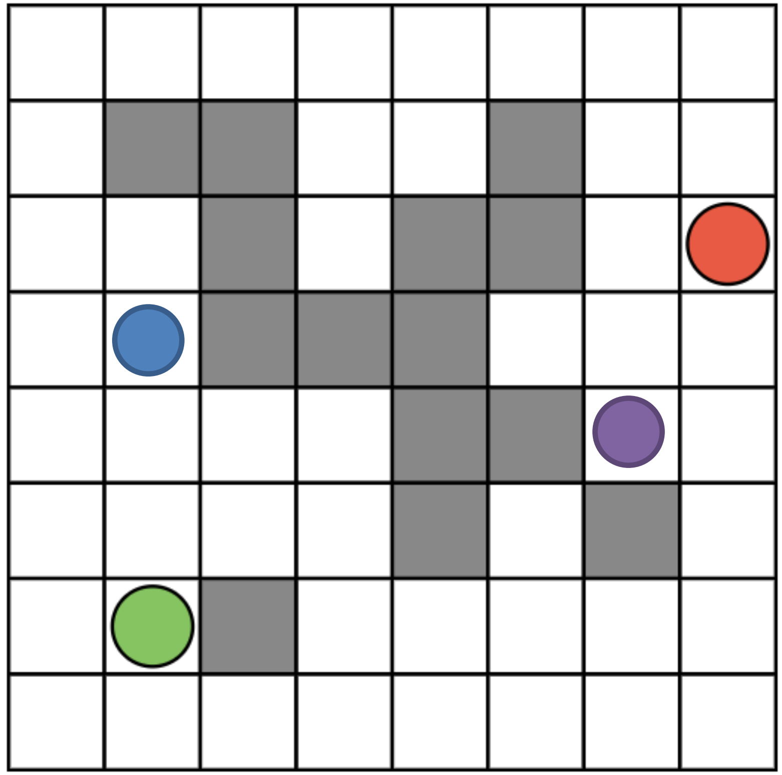
PA2: Mazeworld problem



Robot can move N, S, E, W onto empty squares.

- h1(n) = Manhattan distance
- h2(n) = Euclidean distance

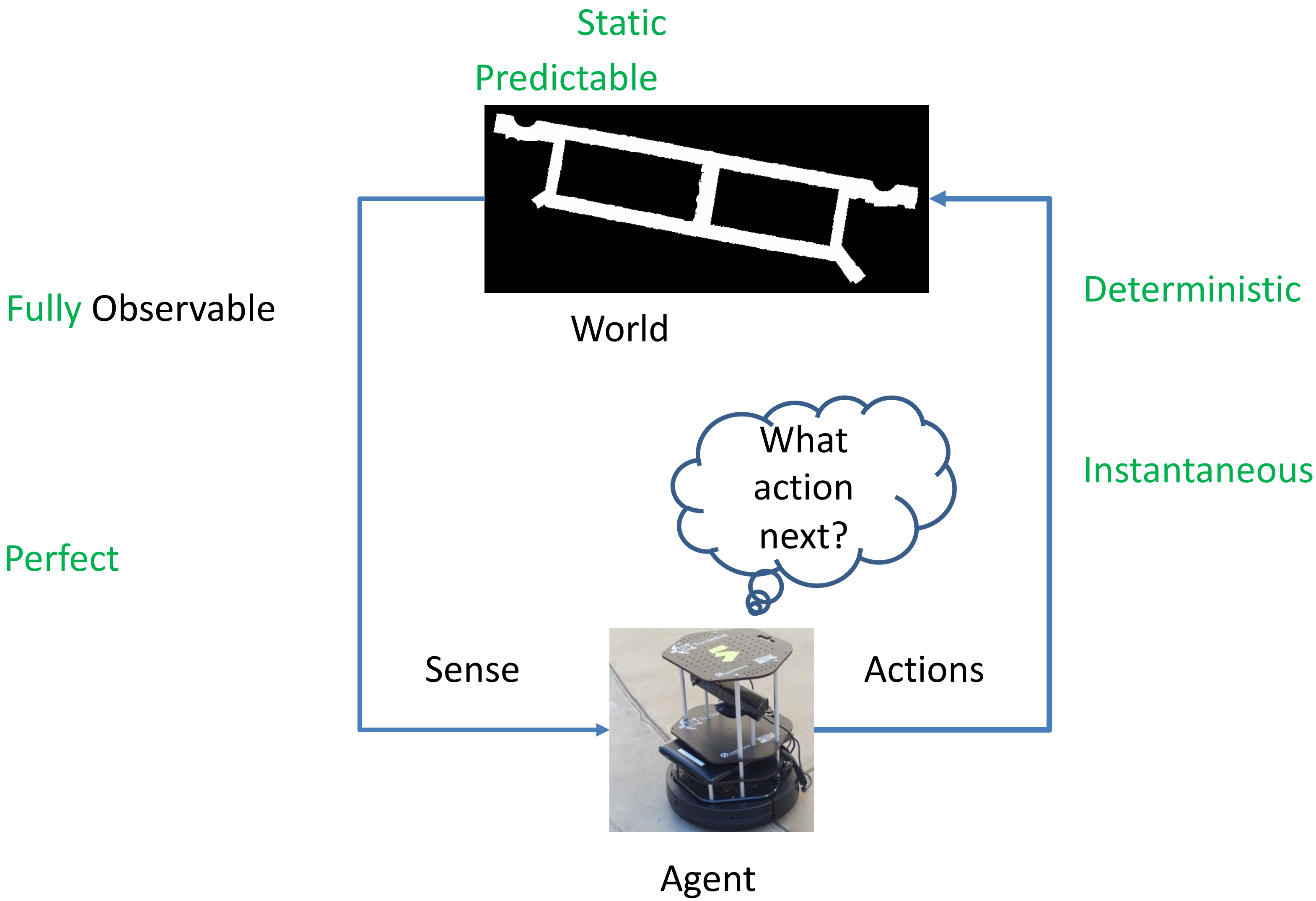
PA2: Mazeworld problem



Robots can move N, S, E, W onto empty squares.

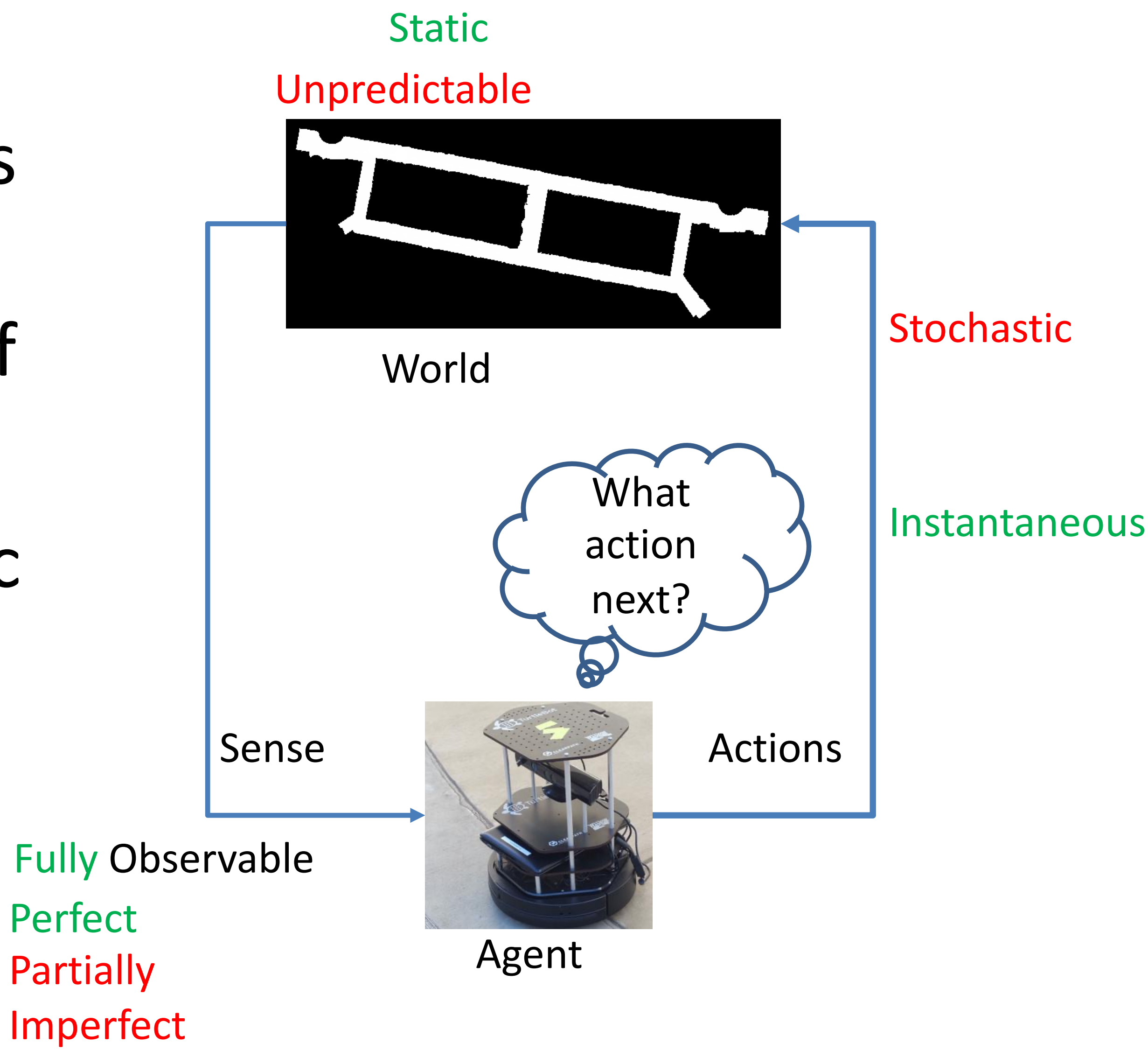
Non-deterministic actions and partial observations

Classical Planning view



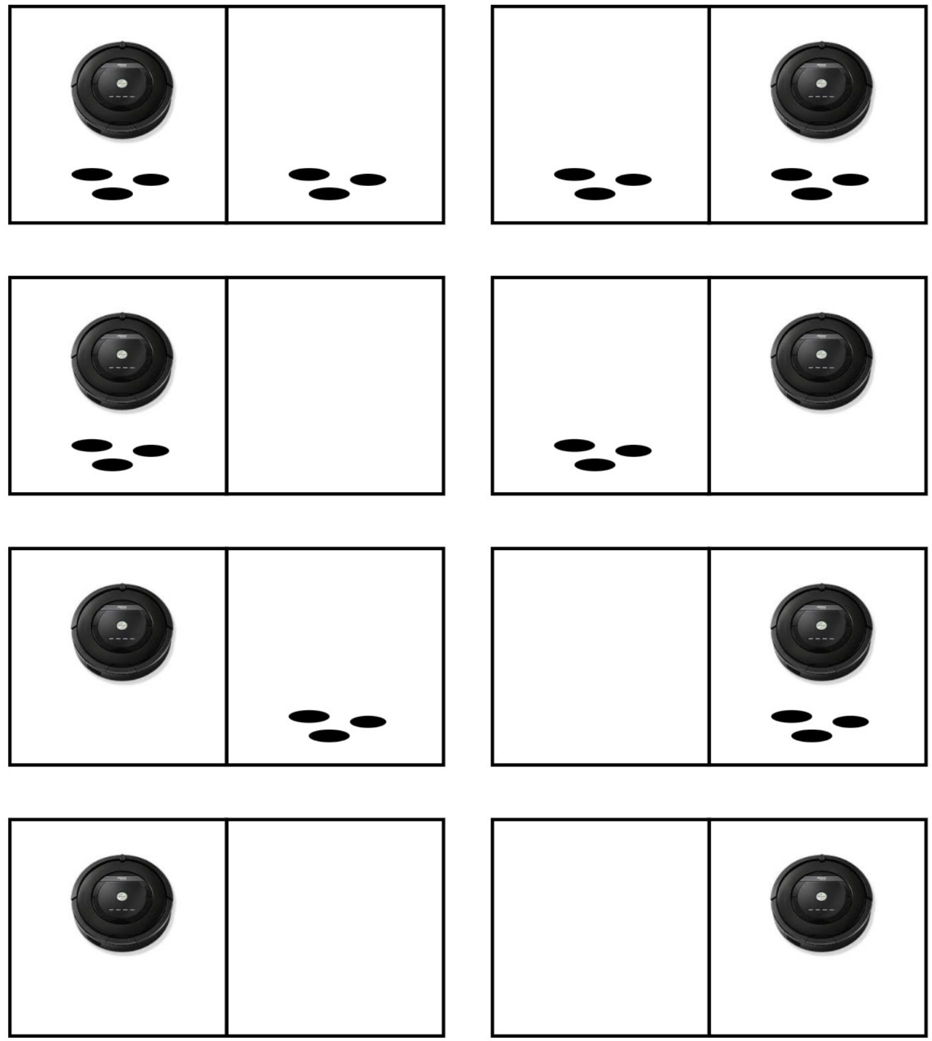
Planning views

- Different planning views which involve different set of techniques
- E.g., Stochastic planning



Vacuum world

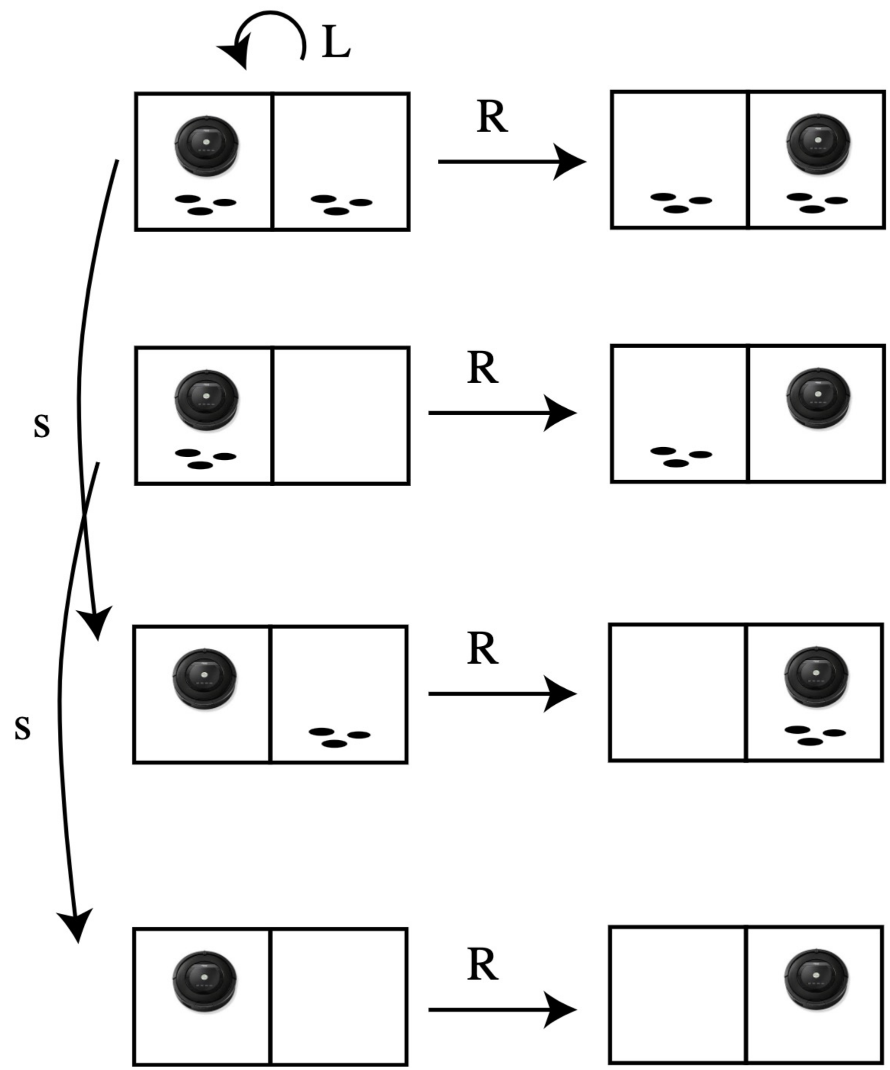
- Clean 2 rooms



Vacuum world: State space

- L go left
- R go right
- S sweep

(not all edges are marked)



Non-deterministic actions

Sweep action:

- If dirty square, then the action cleans the square and sometimes cleans up dirt in an adjacent square
- If applied to a clean square, it might deposit dirt