CS 440
Introduction to Artificial Intelligence

Lecture 2:

Intelligent Agents

Optimization Problems: Local Search Methods

23 January 2020

Webpage:

https://robotics.cs.rutgers.edu/cs-440-intro-to-artifical-intelligence-spring-2020/

Environment

How do you describe an AI problem?

- Environment
- Perceptions/Observations
- Actions
- Evaluation

Environment

- The world in which the problem takes place
- Could be a physical environment or a virtual environment

Examples

- Chess: The chess board
- Image recognition: The real objects
- Robotics: The physical world around the robot

Perception/Observation

How do you describe an AI problem?

- Environment
- Perceptions/Observations
- Actions
- Evaluation

Perceptions/Observations

- Inputs the agent receives

Examples

- Chess: The current state of the board
- Image recognition: An image file
- Robotics: Sensor readings

Convention

- Let **O** be the set of all possible observations
- Let **o**ϵO refer to an individual observation

Actions

How do you describe an AI problem?

- Environment
- Perceptions/Observations
- Actions
- Evaluation

Actions

- Operations the agent can perform

Examples

- Chess: A valid move
- Image recognition: A label for the image
- Robotics: Motor or actuator control

Convention

- Let **A** be the set of all possible actions
- Let **a**ϵ**A** refer to an individual action

Evaluation

How do you describe an AI problem?

- Environment
- Perceptions/Observations
- Actions
- Evaluation

Evaluation

- Feedback

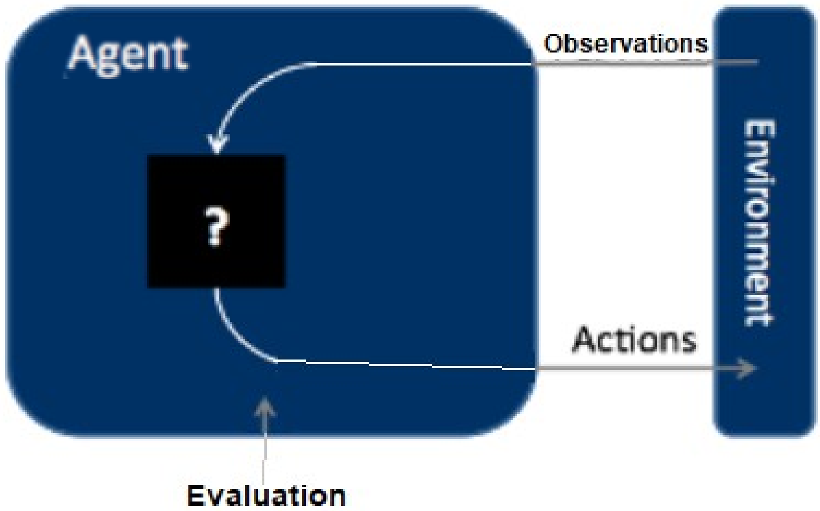
Examples

- Chess:
  - Positive evaluation if wins
  - Negative evaluation if looses
  - Neutral if draws
- Image recognition:
  - Positive evaluation if labels image correctly
- Robotics:
  - Positive evaluation if robot reaches some goal
  - Negative evaluation if robot collides with obstacle

AI Problem Definition

How do you describe an AI problem?

- Environment
- Perceptions/Observations
- Actions
- Evaluation



Select an action that yields best evaluation given the observations as well as any built-in knowledge the agent has about its environment.

State:

- Data structure representing environment
- Incorporate all relevant information

Examples:

- Chess: An 8x8 array
- Image recognition: An array of RGB values
- Robotics: A geometric representation of the environment

Convention:

- Let S be the set of all states
- Let $s \in S$ denote an individual state

Challenges

- States are problem specific
- Size needs to be reasonable
-

Hint: think about if you can represent problem using common data structure such as array or graph.

Reward

How do you describe an AI problem?

- Environment
- Perceptions/Observations
- Actions
- Evaluation

Evaluation

- Feedback

Examples

- Chess:
  - Positive evaluation if wins
  - Negative evaluation if looses
  - Neutral if draws
- Image recognition:
  - Positive evaluation if labels image correctly
- Robotics:
  - Positive evaluation if robot reaches some goal
  - Negative evaluation if robot collides with obstacle

Transition Function

Transition

- How do actions effect state?
- Transition function τ
- $s' = \tau(s,a)$

Examples

- Chess:
  - Move a piece
    - Move : Pawn d2 to d4
    - Update state by removing pawn from d2 and putting it in d4
- Robotics:
  - Movement of robot in environment
  - Change position of objects effected by movement

Sudoku

Discussion: How would we formulate game Sudoku as an AI problem ?

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Environments and their Properties

<u>Fully observable vs. partially observable:</u>

- If the agent has always access to the complete state of the environment, then the environment is fully observable
- Partial observability due to: noisy, inaccurate sensors or sensor limitations

<u>Deterministic vs. stochastic:</u>

- If the next state of the world is completely determined by the current state and the agent's action, then the environment is deterministic.
- If the environment is deterministic except from the actions of other agents, then it is called strategic.

<u>Episodic vs. sequential:</u>

- In an episodic environment the agent's experience does not depend on the actions taken in previous episodes.
- In sequential environments, the current decision could affect all future decisions.

RUTGERS

Environments and their Properties

Fully observable vs. partially observable:

- If the agent has always access to the complete state of the environment, then the environment is fully observable
- Partial observability due to: noisy, inaccurate sensors or sensor limitations

Deterministic vs. stochastic:

- If the next state of the world is completely determined by the current state and the agent's action, then the environment is deterministic.
- If the environment is deterministic except from the actions of other agents, then it is called strategic.

Episodic vs. sequential:

- In an episodic environment the agent's experience does not depend on the actions taken in previous episodes.
- In sequential environments, the current decision could affect all future decisions.

Deterministic, Fully Observable

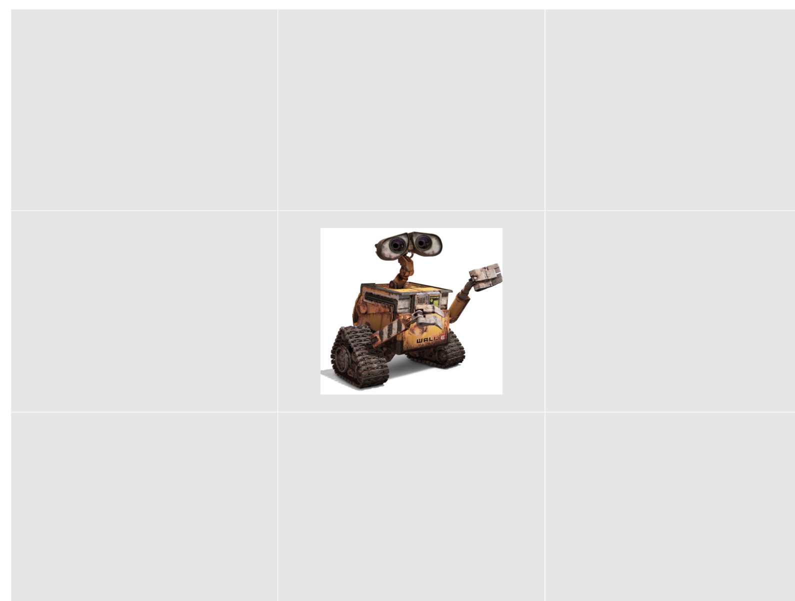
Given

- A fully observable environment
- Observations are state of environment
- A set of possible actions, A
- An evaluator
  - E.g. a set of goal states

Robot in Grid World

Given

- A fully observable environment
  - **A robot in a 3x3 grid world**
  - **State: 3x3 array with position of robot marked**
- Observations are state of environment
  - Cell robot is located in
- A set of possible actions, A
  - UP, DOWN, LEFT, RIGHT
- An evaluator
  - Goal cells

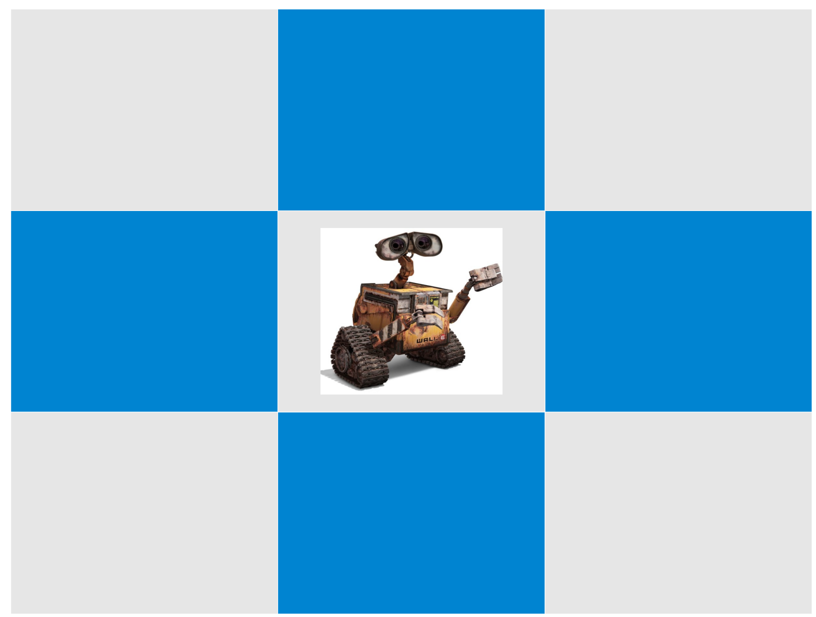


RUTGERS

Robot in Grid World

Given

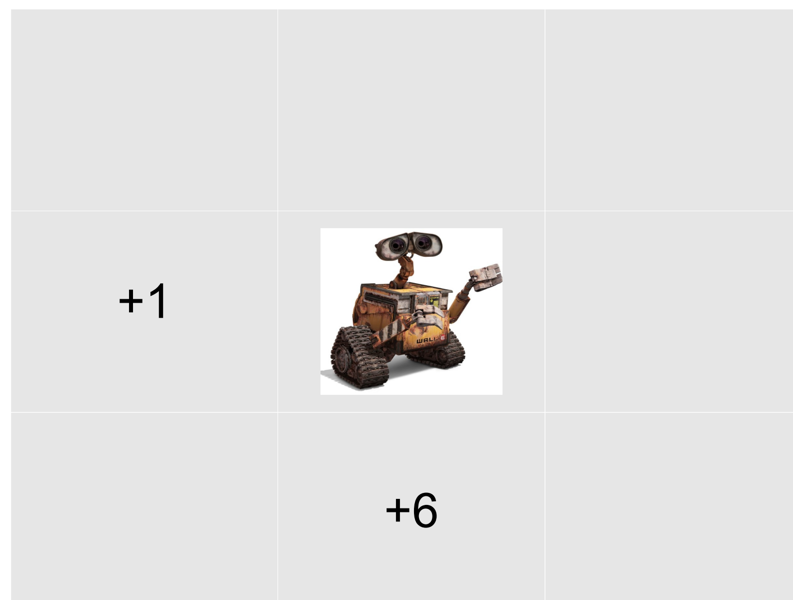
- A fully observable environment
  - A robot in a 3x3 grid world
  - State: 3x3 array with position of robot marked
- Observations are state of environment
  - Cell robot is located in
- A set of possible actions, A
  - **UP, DOWN, LEFT, RIGHT**
- An evaluator
  - Goal cells



Robot in Grid World

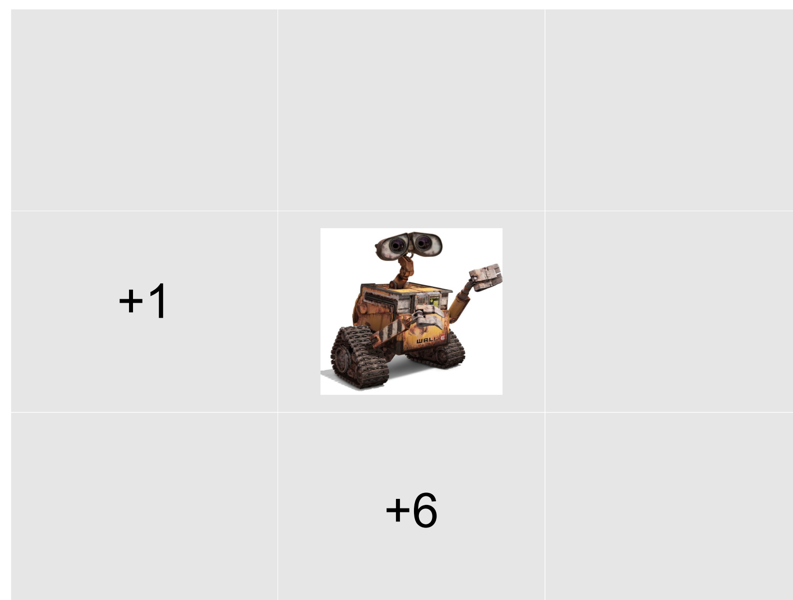
Given

- A fully observable environment
  - A robot in a 3x3 grid world
  - State: 3x3 array with position of robot marked
- Observations are state of environment
  - Cell robot is located in
- A set of possible actions, A
  - UP, DOWN, LEFT, RIGHT
- An evaluator
  - **Goal cells**



Greedy Solution

How do you select an action?



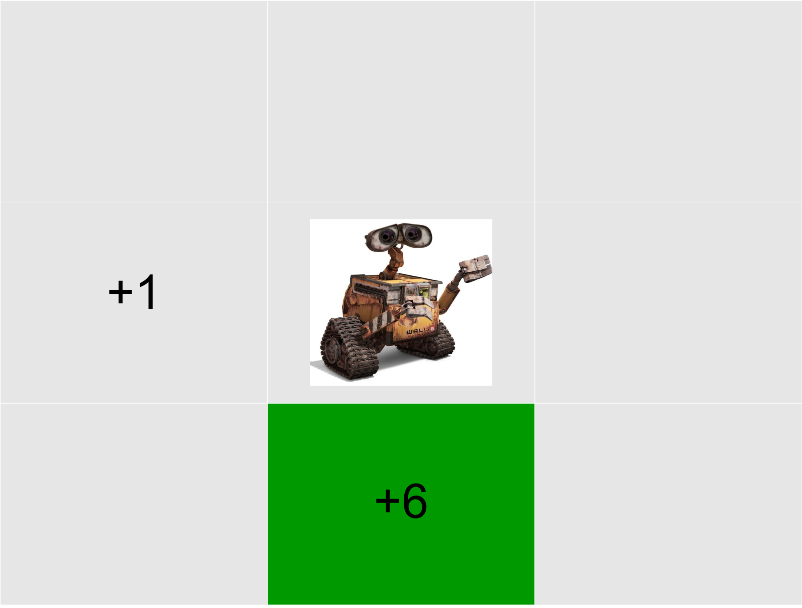
Greedy Solution

How do you select an action?

- Select action that gets you to state with highest reward

  Let s = current state
  For all a∊A
  $s' = \tau(s,a)$
  $r_a = r(s')$
  Select a with largest reward r_a

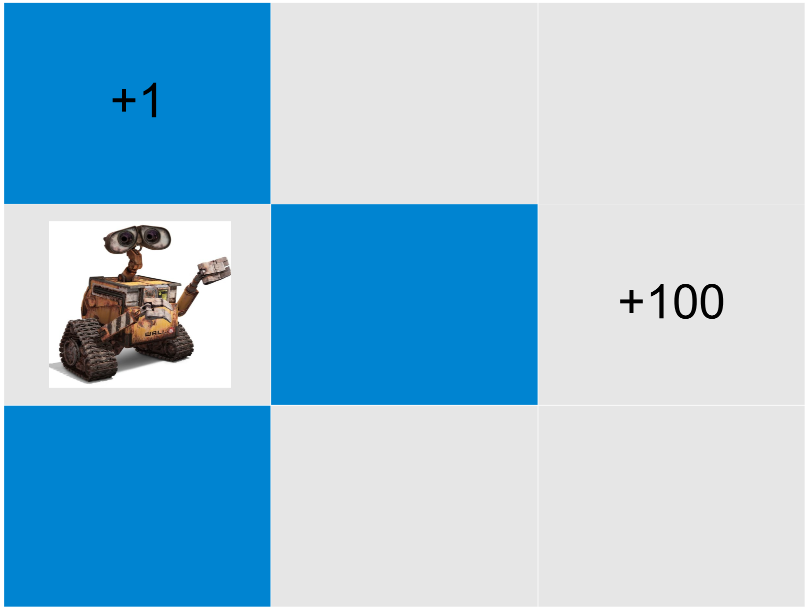
- Select action DOWN



Greedy Solution

Limitations

- Myopic
  - Only looks one move ahead
- We can do much better



Heuristic

- Estimate of utility of state
- Real value indicating utility of state
- Must be defined for all states in **S**

Convention

- h(s)

Examples

- Chess: Value of pieces captured/lost
- Robotics: Euclidean distance from goal
- What would be a good heuristic for Sudoku?

Hill Climbing

Can we combine greedy search with heuristics?

Can we combine greedy search with heuristics?

Hill climbing

Always select action that leads to state with best heuristic

Let s = current state

  For all aϵA

    s' = τ(s,a)

    h_a = h(s')

  Select a with largest reward h_a

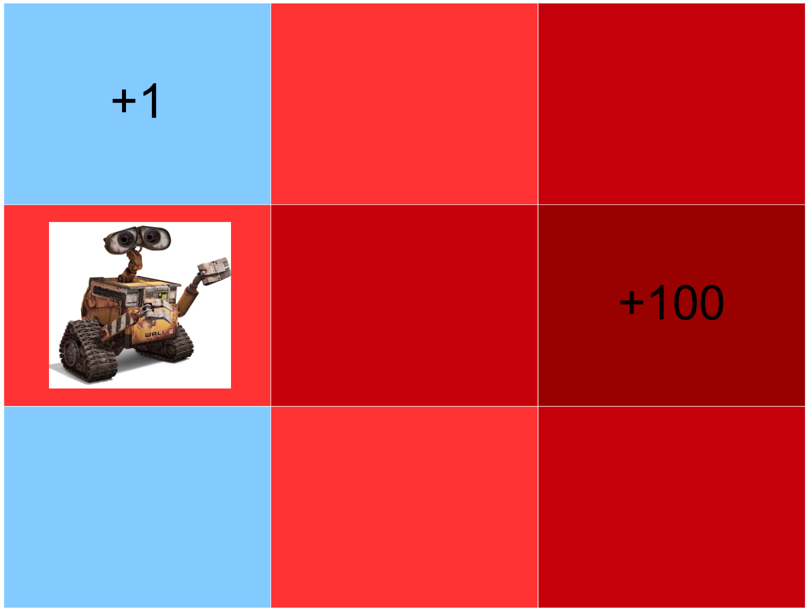
Hill Climbing

Can we combine greedy search with heuristics?

Hill climbing

Always select action that leads to state with best heuristic

Let s = current state
    For all $a \epsilon A$
        $s' = \tau(s,a)$
        $h_a = h(s')$
    Select a with largest reward h_a



Hill Climbing

Can we combine greedy search with heuristics?

Hill climbing

Always select action that leads to state with best heuristic

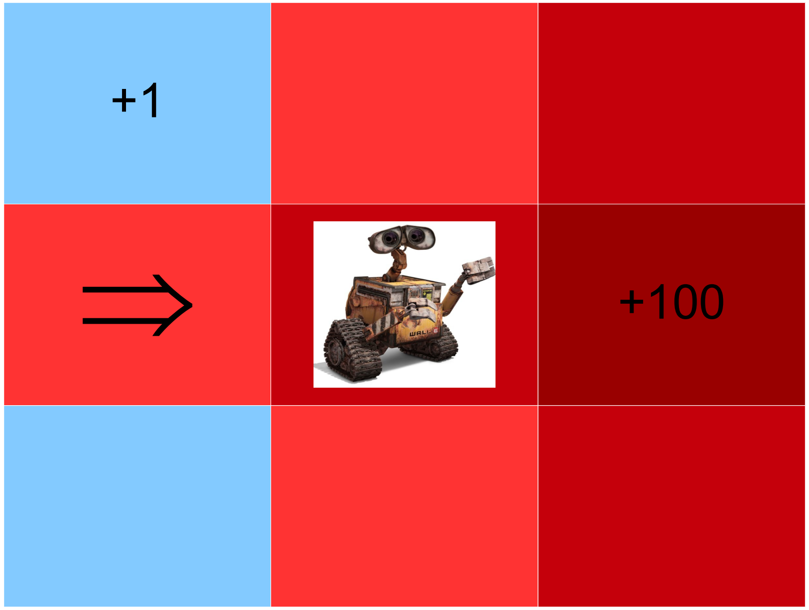
Let s = current statE

For all $a \epsilon A$

$s' = \tau(s,a)$

$h_a = h(s')$

Select a with largest reward h_a



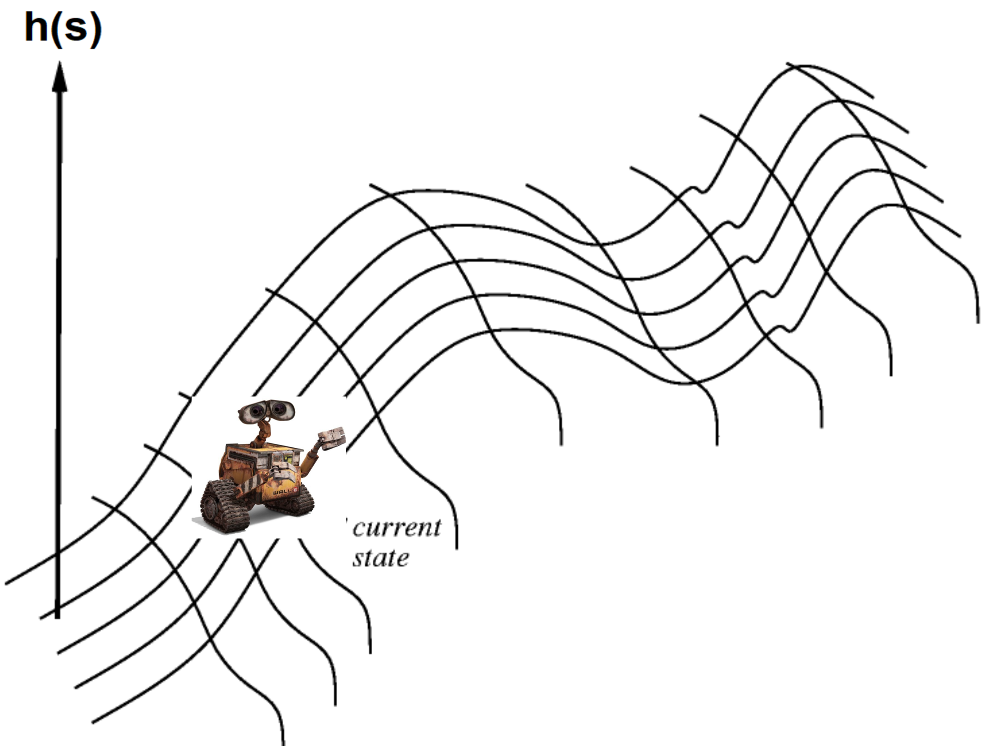
Can we apply hill climbing to continuous state spaces?

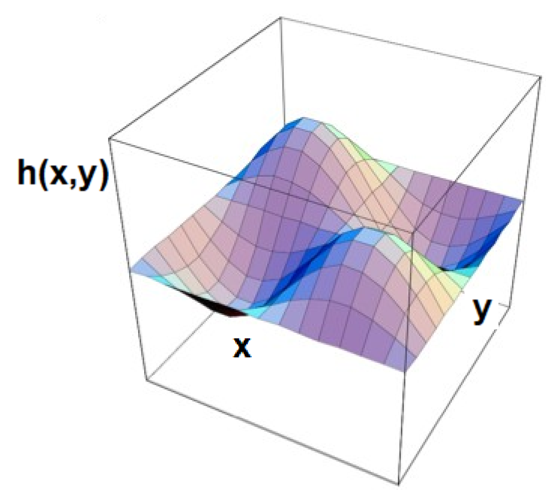
Example 2D Euclidean space

Gradient ascent / descent

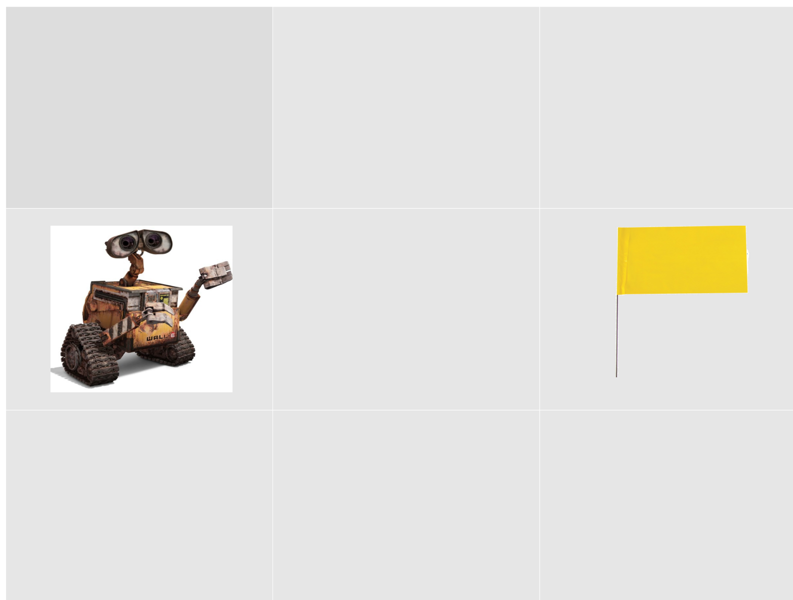
Move in direction of gradient

$\nabla h(s)$

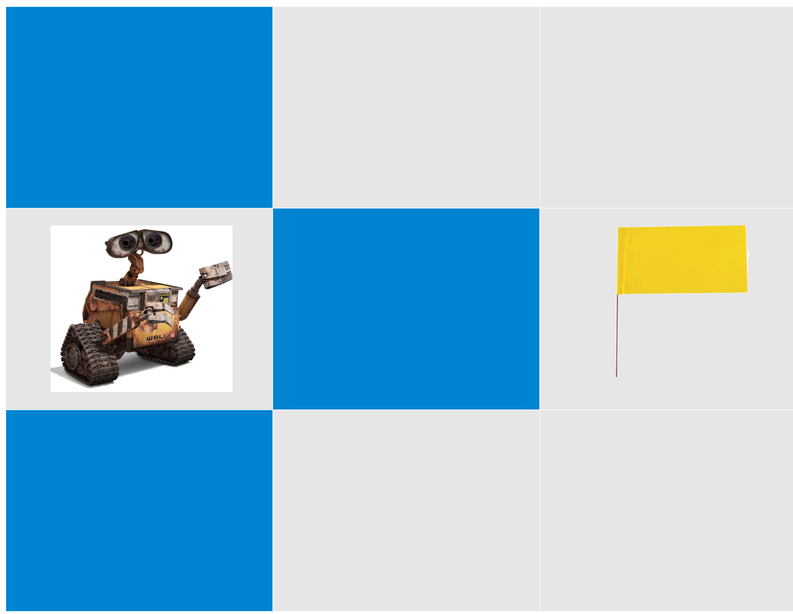




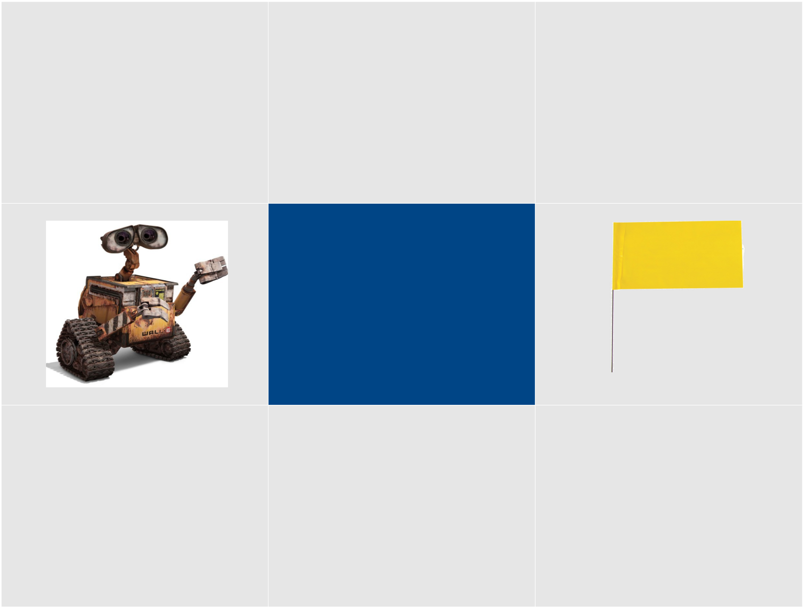
Look Ahead Search



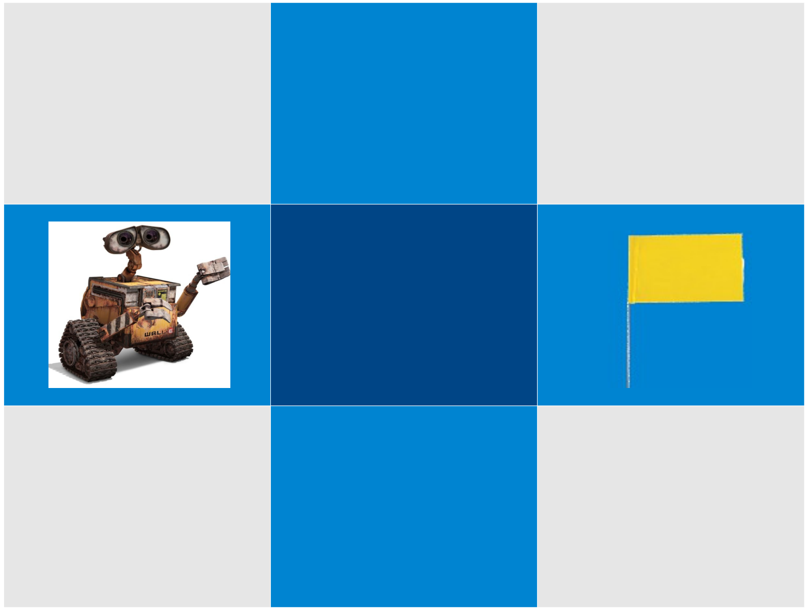
Look Ahead Search



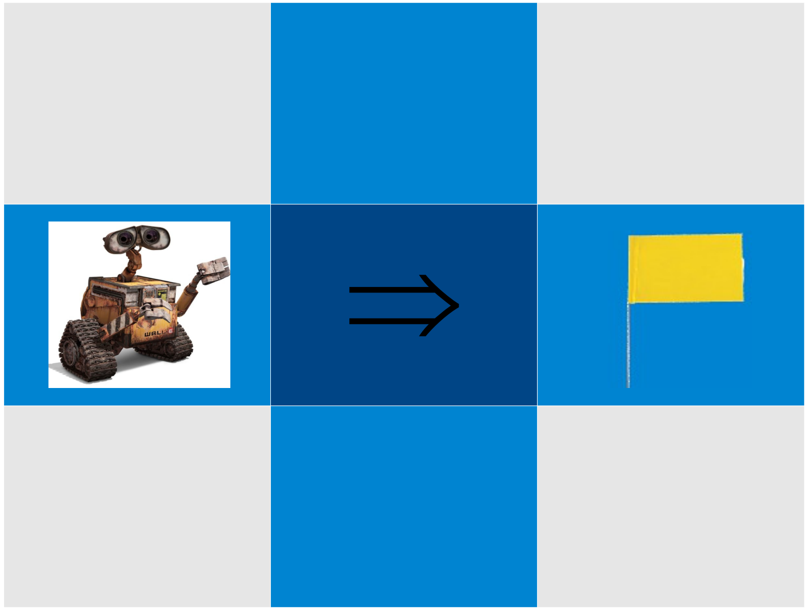
Look Ahead Search



Look Ahead Search

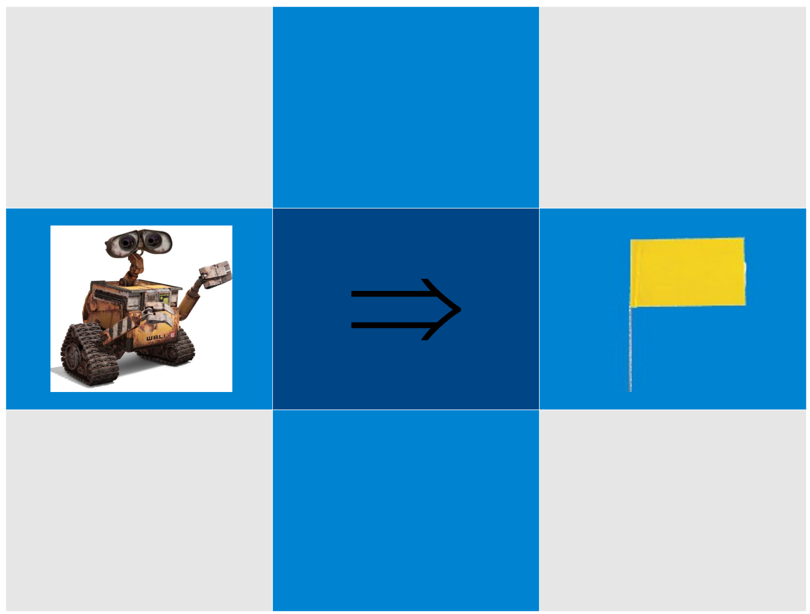


Look Ahead Search



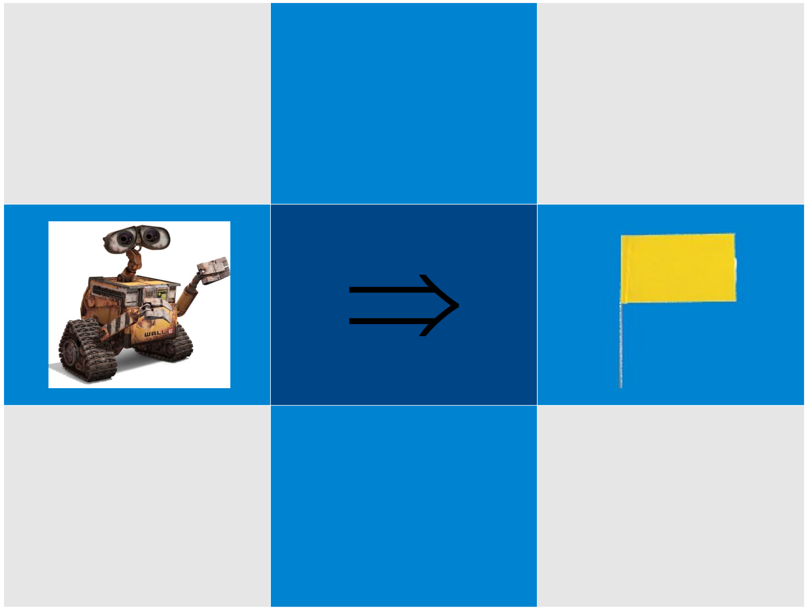
Look Ahead Search

How can we represent look-ahead search?

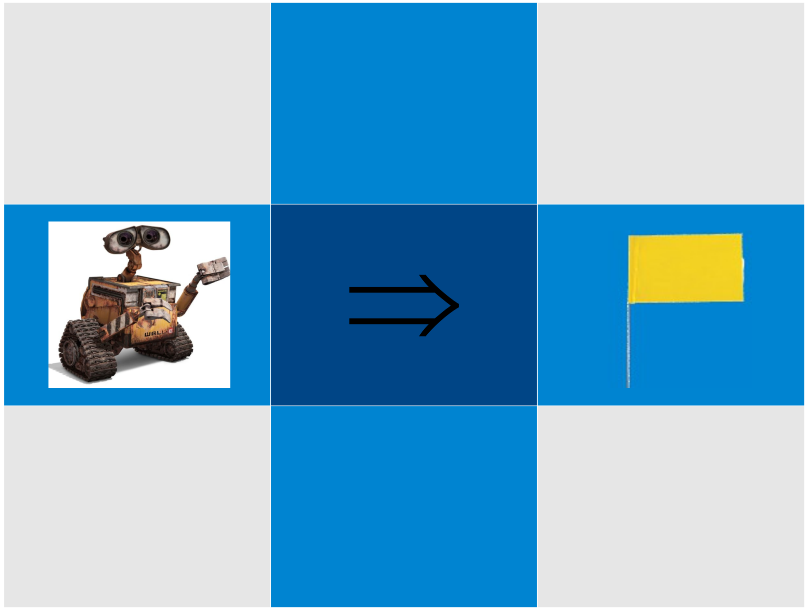


How can we represent look-ahead search?

Search Tree

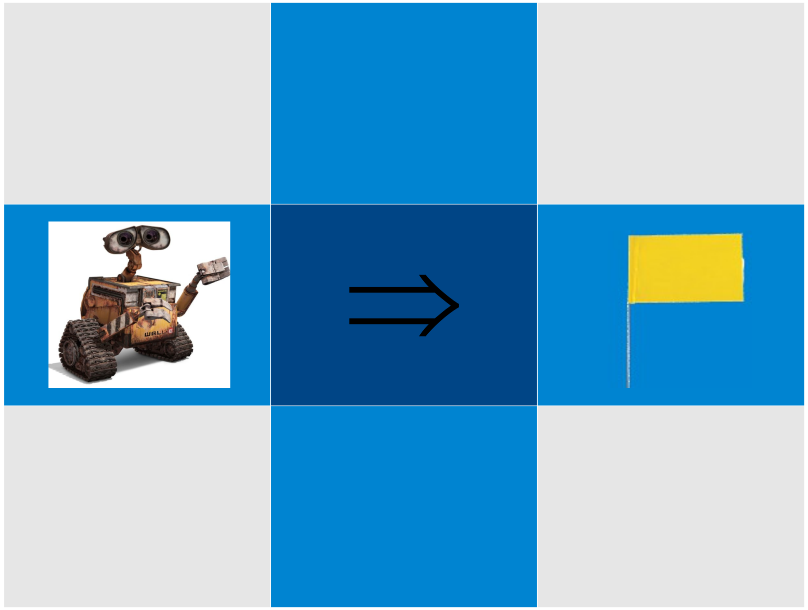


How can we represent look-ahead search?



Look Ahead Search

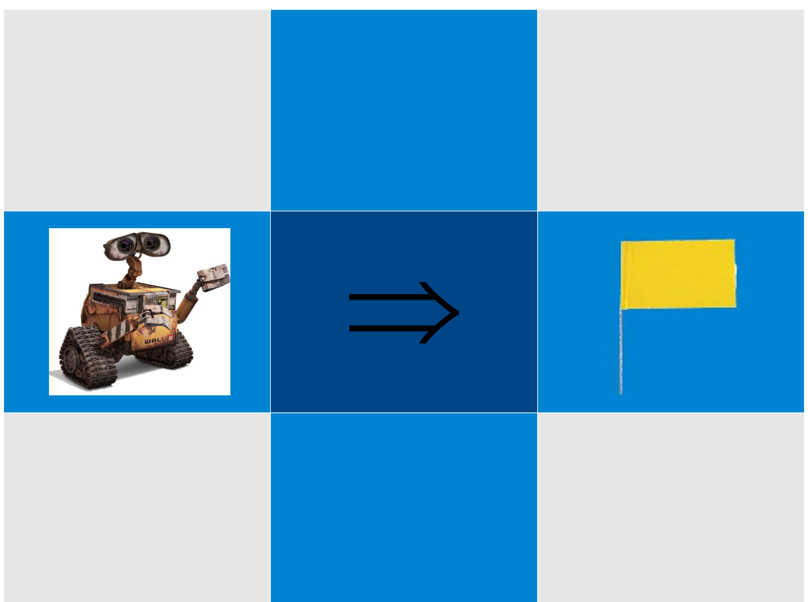
How can we represent look-ahead search?

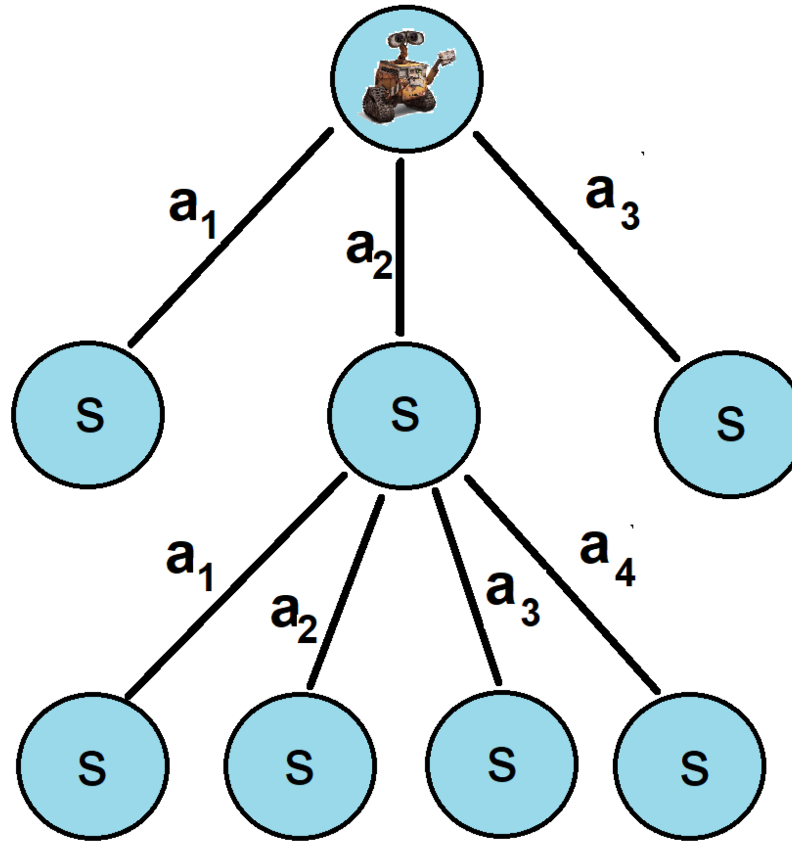


Search Trees

Represent as a tree

- Nodes represent states
- Edges represent actions
- Root is current state
- Children - states you get by taking each action





Depth First (Beam) Search

Depth first search

```
stack.push(initial state)
While(!stack.empty())
    s = stack.pop
    if(s == goal)
        return
    if (!s.visited)
        s.visited=true
        for all actions aϵA
            s'=τ(s,a)
            stack.push(s')
```

Depth First (Beam) Search

Depth first search

```
stack.push(initial state)
While(!stack.empty())
    s = stack.pop
    if(s == goal)
        return
    if (!s.visited)
        s.visited=true
        for all actions aϵA
            s'=τ(s,a)
            stack.push(s')
```

Depth first search in example environment

Breath first search

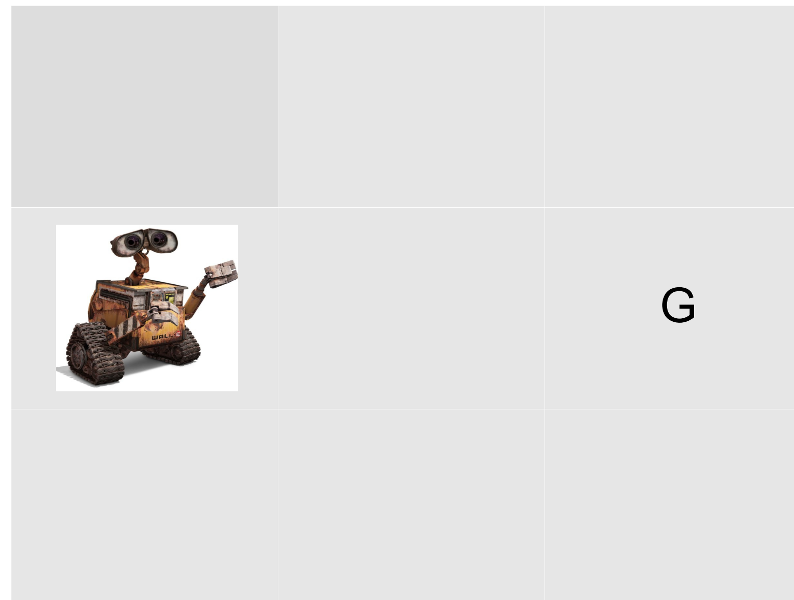
```
queue.push(initial state)
While(!queue.empty())
    s = queue.pop
    if(s == goal)
        return
    if (!s.visited)
        s.visited=true
        for all actions aϵA
            s'=τ(s,a)
            queue.push_back(s')
```

Breath First Search

Breath first search

```
queue.push(initial state)
While(!queue.empty())
    s = queue.pop
    if(s == goal)
        return
    if (!s.visited)
        s.visited=true
        for all actions aϵA
            s'=τ(s,a)
            queue.push_back(s')
```

Depth first search in example environment



Discussion

What are some of the advantages / drawbacks of
- Depth first search
- Breath first search

When would each method be appropriate

How can we combine search with heuristics