CS-E4800 Artificial Intelligence

Jussi Rintanen

Department of Computer Science
Aalto University

February 7, 2019

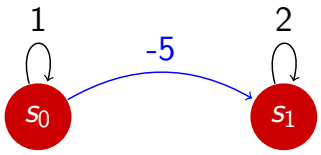
Sequential Decision-Making

- A sequence of actions is taken
- A reward or cost incurred after every action
- Actions change the current state
  - possibility of actions depends on the current state
  - rewards depend on the action (+ state, successor state)
- Objective: maximize rewards, minimize costs

Where is this used?

- intelligent systems control: power, telecom, ...
- robotics
- game playing (non-player characters)

Example: Picking Mushrooms



Actions: Pick mushrooms, Move

Move cost: 5 units
Mushroom yield:
Nearby forest: 1 unit
Far-away forest: 2 units

Question: What to do?

Example: Picking Mushrooms

Which reward sequence preferable?

$$\begin{array}{cccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & \cdots \\ 1 & 1 & 1 & -5 & 2 & 2 & 2 & \cdots \\ -5 & 2 & 2 & 2 & 2 & 2 & 2 & \cdots \end{array}$$

Value of a Reward Sequence $r_0, r_1, r_2, \ldots$

1. finite sum (given fixed N)

$$\sum_{i=0}^{N} r_i$$

2. infinite geometrically discounted sum (with $0 < \gamma < 1$)

$$\sum_{i=0}^{\infty} \gamma^i \cdot r_i$$

3. infinite average

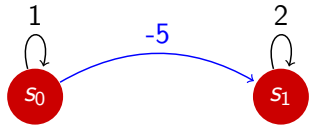
$$\lim_{N \to \infty} \frac{\sum_{i=0}^{N} r_i}{N-1}$$

Value of a Reward Sequence

- Finite sums are used when
  - time horizon is bounded, or
  - approximating infinite with finite: receding-horizon control
- Discounted sums are used often
  - Finite sum for infinite sequences when $0 < \gamma < 1$
  - Easy to handle in algorithms (the Bellman equation)
- Averages useful, but difficult to handle
  - Bellman equation does not apply
  - Easy in special cases only (unichain systems)

Example: Picking Mushrooms

r_0	r_1	r_2	r_3	r_4	r_5		$\sum_{i=0}^{6} r_i$	$\sum_{i=0}^{\infty} \gamma^i \cdot r_i$ $\gamma = 0.9$	$\gamma = 0.8$	$\gamma = 0.7$	avg
1	1	1	1	1	1	$\cdots$	6.00	10.00	5.00	3.33	1.00
1	1	1	−5	2	2	$\cdots$	2.00	12.19	3.98	2.08	2.00
−5	2	2	2	2	2	$\cdots$	5.00	13.00	3.00	−0.33	2.00



Motivation

- Search algorithms from Lecture 2:
  - system fully predictable (only actions change things)
  - actions are deterministic
  - starting state fully known
  - objective: reach a goal state
- Complications in many applications:
  1. unpredictability: actions' effects not known with certainty
  2. unpredictability: external causes of change (uncontrollable change)
  3. multiple "goals" with different rewards (weighed against actions' costs)
  4. system state only partially observable
  5. other decision-makers
- This lecture: 1, 2, 3, 4 (later: 5)

Stochastic Actions

- What to do when actions are stochastic (non-deterministic)?
- Multiple possible successor states
- Reaching goals cannot always be guaranteed
- Options:
  1. Try to maximize probability of reaching goals
  2. Try to minimize expected cost of reaching goals
  3. Try to maximize expected rewards (no goal states!)
- This lecture: Markov decision processes (option 3)

Markov Decision Processes (MDP)

Definition (MDP $\langle S, A, P, R \rangle$)

- S is a (finite) set of states
- A is a (finite) set of actions
- $P : S \times A \times S \to \mathbb{R}$ gives transition probabilities
- $R : S \times A \times S \to \mathbb{R}$ is a reward function

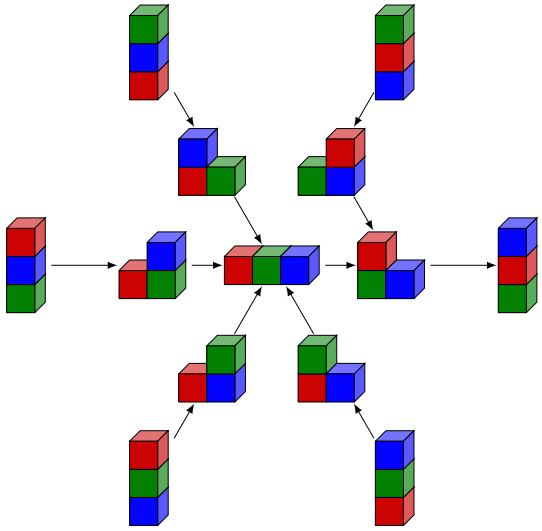
Notice that

- For given $s \in S$ and $a \in A$, $\left(\sum_{s' \in S} P(s, a, s')\right) = 1.0$
- Plan/policy given as $\pi : S \to A$
- Usually no designated initial state
- Reward functions are often $R(s, a) : S \times A \to \mathbb{R}$

Policies for MDPs (deterministic example)

All actions

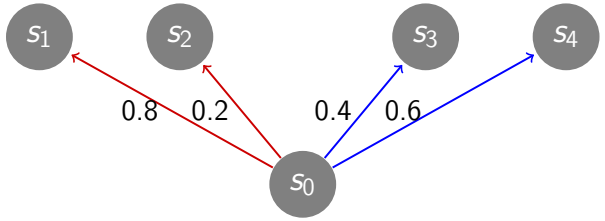
Policy/plan



Value of a State (with nondeterminism)

The Bellman Equation

$$\max(R(s, \bullet) + \gamma(0.8v(s_1) + 0.2v(s_2)), R(s, \bullet) + \gamma(0.4v(s_3) + 0.6v(s_4)))$$



$$v(s) = \max_{a \in A} \left(R(s, a) + \gamma \sum_{s' \in S} P(s, a, s') v(s') \right)$$

Bellman Equation

The value $v(s)$ of state s under the best possible plan/policy:

$$v(s) = \max_{a \in A} \left(R(s,a) + \gamma \sum_{s' \in S} P(s,a,s') v(s') \right)$$

Or, if the reward is dependent on the successor state s':

$$v(s) = \max_{a \in A} \sum_{s' \in S} P(s,a,s') [R(s,a,s') + \gamma v(s')]$$

"A state's value is what is achieved with the action that maximizes the expected value of immediate reward + discounted future rewards."

Value Iteration

1. Let $n := 0$ and $v_0 : S \to \mathbb{R}$ be any value function.
2. For every $s \in S$

$$v_{n+1}(s) = \max_{a \in A} \left(\sum_{s' \in S} P(s,a,s') \left(R(s,a,s') + \gamma v_n(s') \right) \right).$$

   Go to 3 if $|v_{n+1}(s) - v_n(s)| < \frac{\epsilon(1-\gamma)}{2\gamma}$ for all $s \in S$.
   Otherwise set $n := n + 1$ and repeat this step.
3. Policy $\pi : S \to A$ given by

$$\pi(s) = \arg\max_{a \in A} \sum_{s' \in S} P(s,a,s') \left(R(s,a,s') + \gamma v_n(s') \right).$$

Algorithms for Finding Optimal Policies

Value Iteration

- Iterate by finding value functions closer to optimal
- Policy implicit in value function
- Terminate when change smaller than given bound

Policy Iteration

- Iterate by improving policy bit by bit
- Fewer rounds than Value Iteration
- Termination when policy not improved

(Not covered in this lecture!)

Value Iteration

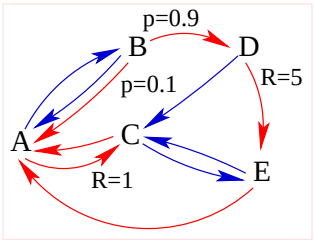
Theorem

Let v_π be the value function of the policy produced by the value iteration algorithm, and let v^* be the value function of an optimal policy. Then $|v^*(s) - v_\pi(s)| \leq \epsilon$ for all $s \in S$.

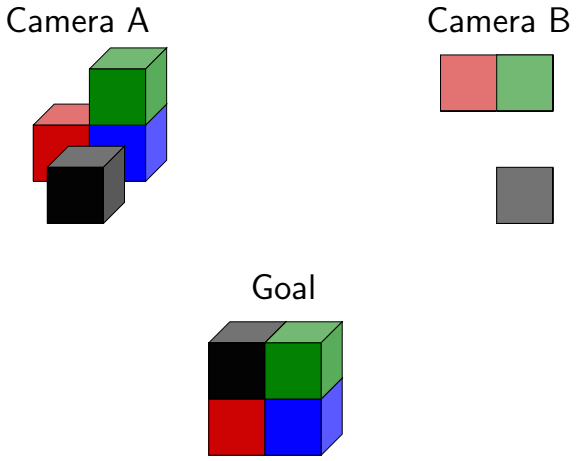
Value Iteration

Let $\gamma = 0.6$.

i	$v_i(A)$	$v_i(B)$	$v_i(C)$	$v_i(D)$	$v_i(E)$
0	0.000	0.000	0.000	0.000	0.000
1	1.000	0.000	0.000	5.000	0.000
2	1.000	2.760	0.600	5.000	0.600
3	1.656	2.760	0.600	5.360	0.600
4	1.656	2.994	0.994	5.360	0.994
5	1.796	2.994	0.994	5.596	0.994
6	1.796	3.130	1.078	5.596	1.078
7	1.878	3.130	1.078	5.647	1.078
8	1.878	3.162	1.127	5.647	1.127
⋮					
19	1.912	3.186	1.147	5.688	1.147
20	1.912	3.186	1.147	5.688	1.147



What Difference Does Observability Make?

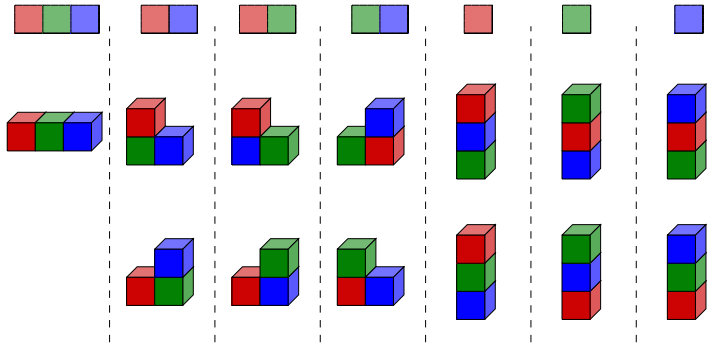


Partial Observability

- Not all features of the state can be observed
  ⇒ Cannot determine current state unambiguously
  ⇒ Many possible current states
- Sensors incomplete, imprecise, not enough sensors
- Games:
  - Card games: Poker, Bridge, ...
  - Mastermind (1970ies code-breaking game)
- Memory becomes necessary (think about Chess vs. Poker)

Observability

Deterministic observations partition the state space

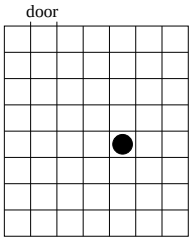


Belief State

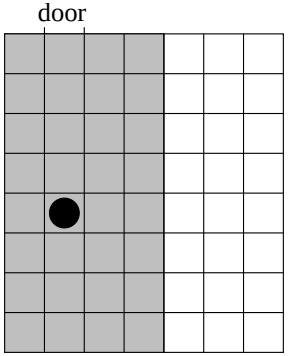
- Belief State
  - with probabilities: probability distribution over states (discussed later)
  - without probabilities: set of possible current states
- belief state of size 1: state known unambiguously
- large belief state $\rightarrow$ state very incompletely known
- small belief state $\rightarrow$ state known more accurately
- observations can reduce the size
- nondeterministic actions can increase or reduce
- deterministic actions can reduce

The Belief Space: Example

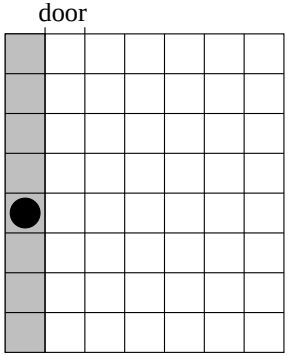
- Robot without sensors, in a 7×8 room, trying to exit
- Position unknown, orientation known
- Actions: North, South, East, West
- Next slides: *One possible* location of the robot ($\bullet$) and the change in the belief state at every execution step.



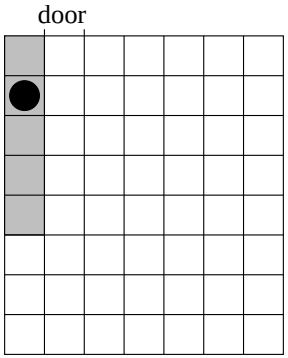
Example: Belief State After WWW



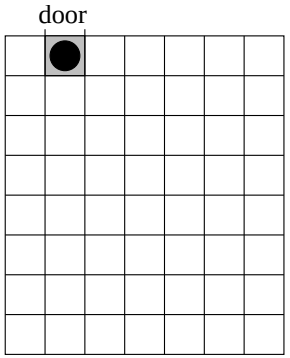
Example: WWWWWW



Example: WWWWWWNNN



Example: WWWWWWNNNNNNNE



The Belief Space

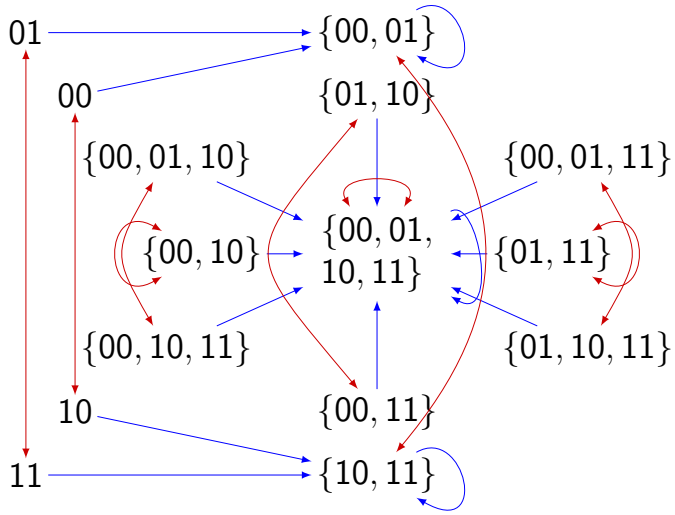
Example

Belief space over state space $\{00, 01, 10, 11\}$

state variables	states	belief states
$n = 2$	$2^n = 4$	$2^{2^n} = 16$

Next slide:
red action: the complement 1st bit
blue action: assign a random value to the 2nd bit

The Belief Space



Algorithms (Special Case: No Observations At All)

Without observations, plans are sequences of actions from the initial belief state to a belief state included in the goals

- Belief states represented as formulas, BDDs, ...
- Standard search algorithms (A$*$, Greedy BFS, ...)
- Belief space very large; limits applicability of these algorithms with belief states

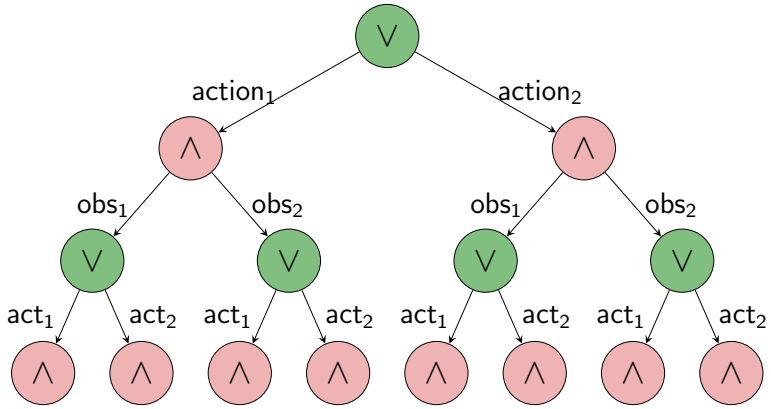
Conditional Plans and AND-OR Trees

Observations allow choosing what to do in different circumstances. The interplay between agent's choices and the environment's choices represented as an AND-OR tree:

- AND-node: Uncontrollable events (nondeterminism, observation)
- OR-node: Controllable events (actions)
- Conditional plans include one action for each OR-node
- AND-OR tree $\sim$ game tree (lecture 7)

Algorithms for systematic AND-OR tree search: AO*, LAO*, ...

AND-OR Trees and Conditional Plans



action$_1$; IF obs$_1$ THEN (action$_2$;...) ELSE (action$_1$; ...)

Mastermind

- Code: sequence of 4 colors
- The code is hidden from the player
- Player tries to guess the code
- Response to each guess: 0 to 4 of
  - correct color, correct position
  - correct color, wrong position
- Solution by AND-OR search in discrete belief space

- Critical: heuristics for choosing the guesses
- Can always be solved with ≤ 5 moves (difficult for humans)

Observations + Discrete Belief States

Deterministic observations corresponds to sets of states (those states where the observation can be made.)

Belief State update with an Observation

An observation $O \subseteq S$ updates a belief state $B \subseteq S$ to

$$B' = B \cap O$$

Example

For belief B = { Sunday, Monday, Tuesday } the observation "weekend" O = { Saturday, Sunday } yields

$$B' = B \cap O = \{\text{Sunday}\}$$

Probabilistic Belief States

Assigning a numeric probability to the states in a belief state is needed:

- Defining expected value of a plan
- Defining probability of success of a plan

Applications:

- Active diagnosis: planning for diagnostic tests to maximize success or optimize costs
  - Medical diagnosis
  - Technical systems diagnosis

Actions + Discrete Belief States

A (nondeterministic) action a correspond to a relation $R_a \subseteq S \times S$ that relate a state with its successor states.

Belief State update with an Action

$$B' = \{s' \in S | s \in B, sR_as'\}$$

Applicability of an Action

An action a is considered to be applicable only if for all $s \in B$, sR_as' for some $s' \in S$.

Observations + Probabilistic Belief States

Instead of set of possible states, a belief is probability distribution B over all states S.

Belief State update with an Observation

An observation O updates B to B' according to Bayes Rule:

$$B'(s) = P(s|O) = \frac{P(O|s)B(s)}{\sum_{s_2 \in S} P(O|s_2)B(s_2)} \quad (1)$$

(The denominator represents, for belief state B, and before any observations are made, the probability that observation O will be made.)

Observations + Probabilistic Belief States

Example

$$
\begin{aligned}
B(s_0) &= 0.01 \\
B(s_1) &= 0.99 \\
P(O|s_0) &= 0.1 \\
P(O|s_1) &= 0.0 \\
P(s_0|O) &= \frac{0.1 \cdot 0.01}{0.1 \cdot 0.01 + 0.0 * 0.99} = 1.0 \\
P(s_1|O) &= \frac{0.0 \cdot 0.99}{0.1 \cdot 0.01 + 0.0 * 0.99} = 0.0
\end{aligned}
$$

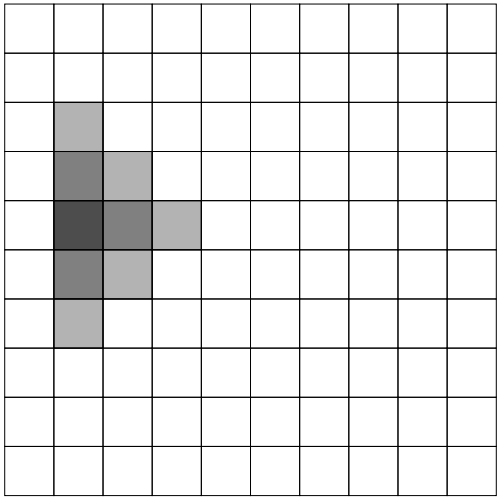
Action + Probabilistic Belief States

Successor State Probabilities

Action a maps B to B', with a applicable in all s with $B(s) \neq 0$.

$$B'(s') = \sum_{s \in S} B(s) P(s, a, s')$$

Probabilistic Belief States: Example



- Robot moves in a grid
- Wall detectable only when robot next to it
- Wall not visible (yet)

Partially Observable MDPs (POMDP)

Definition (POMDP $\langle S, A, P, R, E, O \rangle$)

- S is a (finite) set of states
- A is a (finite) set of actions
- $P : S \times A \times S \to \mathbb{R}$ gives transition probabilities
- $R : S \times A \times S \to \mathbb{R}$ is a reward function
- E is a (finite) set of observations (evidence)
- $O : A \times S \times E \to \mathbb{R}$ gives observation probabilities

Belief Update for POMDPs

Belief update: action $a \in A$

$$B'(s') = \sum_{s \in S} P(s, a, s') B(s)$$

Belief update: action $a \in A$, observation $e \in E$

$$B'(s') = \frac{O(a, s', e) \sum_{s \in S} P(s, a, s') B(s)}{\sum_{s_2 \in S} (O(a, s_2, e) \sum_{s_1 \in S} (P(s_1, a, s_2) B(s_1)))}$$

Algorithms for POMDPs

- Not covered in this lecture
- There is a value iteration algorithm, which represents the value function of an POMDP as a collection of conditional plans and their associated values for all states.
- Solving POMDPs is hard:
  - "Is there plan of value $\geq c$" is undecidable (Madani, Hanks & Condon 2003)
  - Many restricted cases are EXPTIME-hard, requiring 2^n time (worst case)