

What difference does observability make?

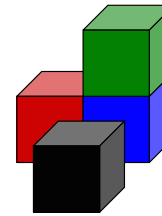
CS-E4800 Artificial Intelligence

Jussi Rintanen

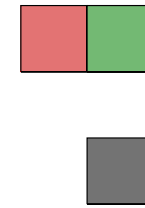
Department of Computer Science
Aalto University

February 23, 2017

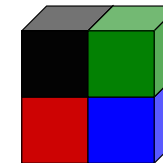
Camera A



Camera B



Goal

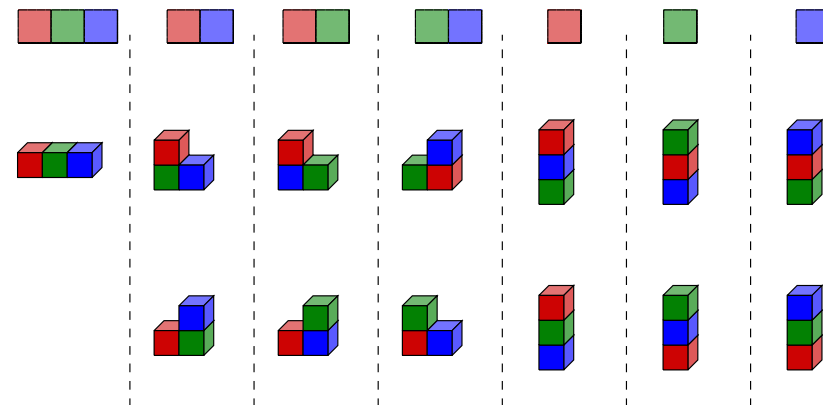


Partial Observability

- 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:
 - Mastermind (1970ies code-breaking game)
 - Card games: Poker, Bridge
- **Memory** (= remembering observation history) becomes necessary (think about Chess vs. Poker)

Observability

Deterministic observations partition the state space



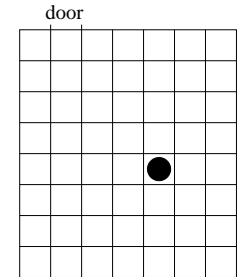
Belief State

- 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 → state very incompletely known
- small belief state → state known more accurately
- observations can **reduce** the size
- nondeterministic actions can **increase** or **reduce**
- deterministic actions can **reduce**

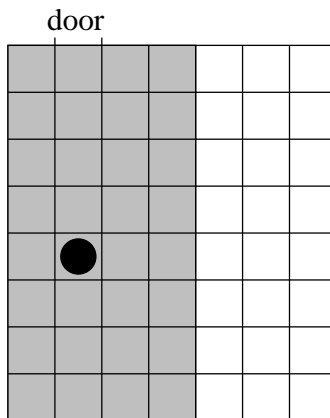
The belief space

Example

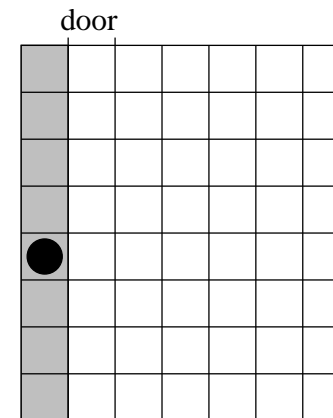
- Robot without sensors, in a 7×8 room, trying to exit
- Position unknown, orientation known
- Actions: North, South, East, West
- Next slides: *One possible* location of the robot (●) and the change in the belief state at every execution step.



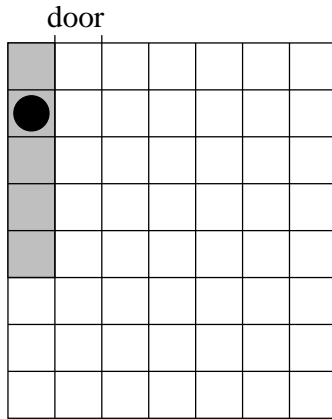
Example: belief state after WWW



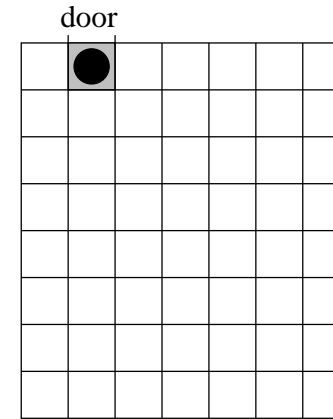
Example: WWWWWW



Example: WWWWWN



Example: WWWWWNNNNNNE



The belief space

Example

Example

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

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

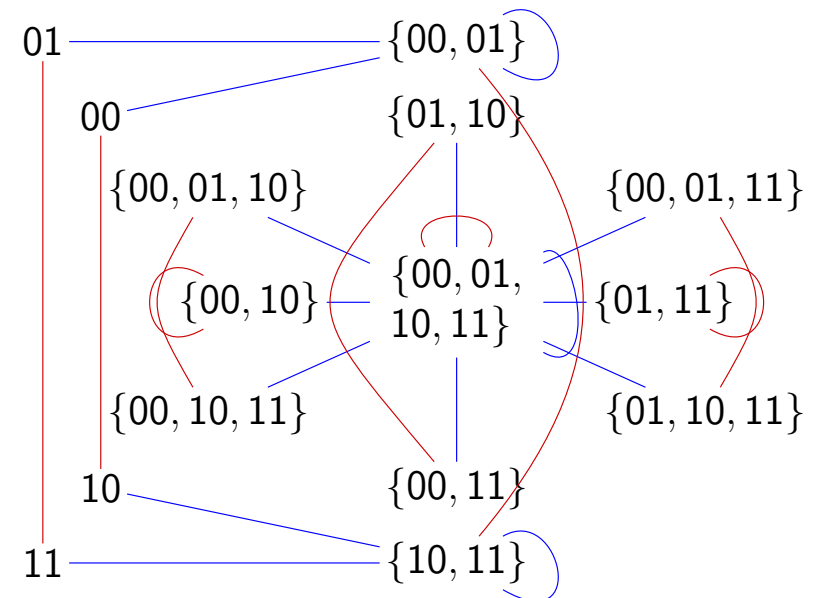
Next slide:

red action: the complement 1st bit

blue action: assign a random value to the 2nd bit

The belief space

Example



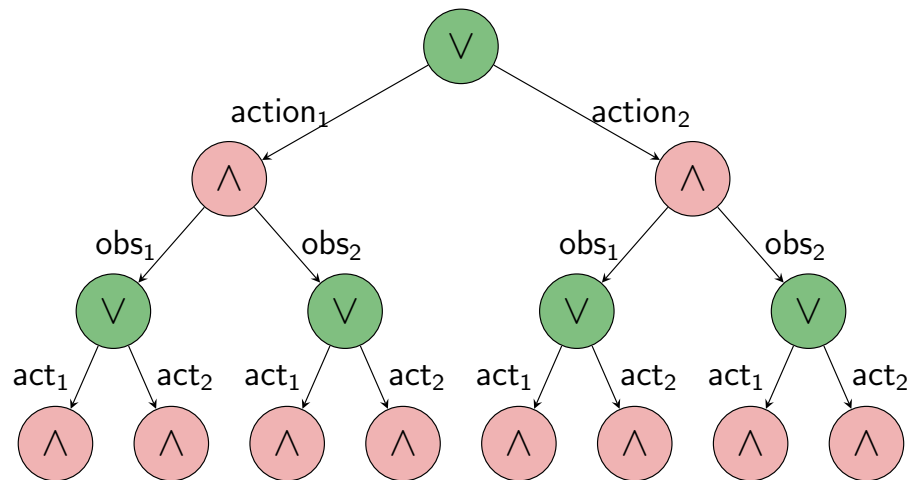
Algorithms

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

(Special case: deterministic actions, one initial state. This is the case covered in the first lectures!)

AND-OR Trees and Conditional Plans



action₁; IF obs₁ THEN (action₂;...) ELSE (action₁; ...)

Conditional Plans and AND-OR Trees

- Possible behaviors under nondeterminism and sensing/observations
- AND-node: Uncontrollable events (nondeterminism, observation)
- OR-node: Controllable events (actions)
- A conditional plan represent one possible action at each (reachable) OR-node
- AND-OR tree ~ game tree (next lecture)



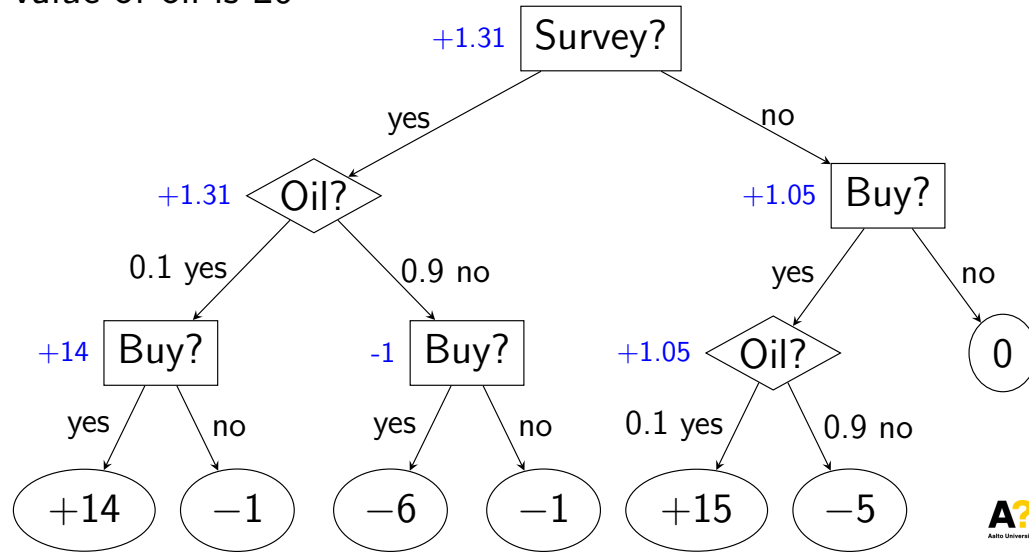
Mastermind

- 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 max. 5 moves (but is difficult)



Expectimax Trees

survey action: cost 1
 block: cost 5
 value of oil is 20



Partial Observability and Sensing

- Most applications include **sensing** / **observations**
- Sensing (observations) allows reducing size of belief state
 - sensing → alternative belief states
- In many application, **probabilities** of states critical
 ⇒ extensions of probabilistic methods like MDPs

Expectimax Trees

- decision/action node: **maximum**

$$\max_{c \in \text{children}} v(c)$$

- chance/observation node: **expected value**

$$\sum_{c \in \text{children}} P(c) \times v(c)$$

- solution tree = prune all but best child of action nodes

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 $B = \{ \text{Sunday, Monday, Tuesday} \}$ the observation "weekend" $O = \{ \text{Saturday, Sunday} \}$ yields

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

Observations + Probabilistic Belief States

Instead of set of possible states, a belief is **probability distribution** $B(S)$ over states.

Belief State update with an Observation

An observation O updates $B(S)$ to $B'(S)$ 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.

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

Observations + Probabilistic Belief States

Example

$$B(s_0) = 0.01$$

$$B(s_1) = 0.99$$

$$P(O|s_0) = 0.1$$

$$P(O|s_1) = 0.0$$

$$P(s_0|O) = \frac{0.1 \cdot 0.01}{0.1 \cdot 0.01 + 0.0 \cdot 0.99} = 1.0$$

$$P(s_1|O) = \frac{0.0 \cdot 0.99}{0.1 \cdot 0.01 + 0.0 \cdot 0.99} = 0.0$$



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 $e \in E$

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



Value Iteration for POMDPs

- Problem: Belief space **infinite** \Rightarrow Size of value function representation **unbounded**
- Solution:
 - Iteration i : generate **conditional plans** π of depth i
 - Evaluate value of each plan π for every state $(v_\pi(s_1), \dots, v_\pi(s_n))$
 - The collection of **all these vectors** $(v_\pi(s_1), \dots, v_\pi(s_n))$ represent **value function** for all possible **belief states**: The maximum value obtainable for a given belief state by one of the conditional plans

Value Iteration for POMDPS

Example

- Two states s_0 and s_1
- Two actions **Stay** and **Go**
- Two observations s_0 and s_1
- $R(s_0) = R(s_0, \cdot, \cdot) = 0$ and $R(s_1) = R(s_1, \cdot, \cdot) = 1$
- $P(s_i, \text{Stay}, s_i) = 0.9$ and $P(s_i, \text{Stay}, s_{1-i}) = 0.1$
- $P(s_i, \text{Go}, s_{1-i}) = 0.9$ and $P(s_i, \text{Go}, s_i) = 0.1$
- $O(\cdot, s_i, s_i) = 0.6$ and $O(\cdot, s_i, s_{1-i}) = 0.4$



Values of states under 1-step plans

$$v_{Go}(s_0) = R(s_0) + \gamma(0.9R(s_1) + 0.1R(s_0)) = 0.9\gamma$$

$$v_{Go}(s_1) = R(s_1) + \gamma(0.9R(s_0) + 0.1R(s_1)) = 1.0 + 0.1\gamma$$

$$v_{Stay}(s_0) = R(s_0) + \gamma(0.9R(s_0) + 0.1R(s_1)) = 0.1\gamma$$

$$v_{Stay}(s_1) = R(s_1) + \gamma(0.9R(s_1) + 0.1R(s_0)) = 1.0 + 0.9\gamma$$

(Reward from starting state + reward from successor state!)

With $\gamma = 0.9$ these are:

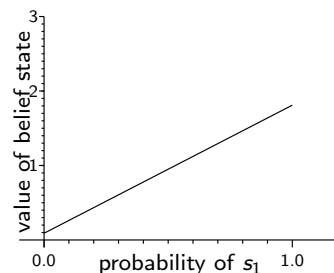
$$v_{Go}(s_0) = 0.81$$

$$v_{Go}(s_1) = 1.09$$

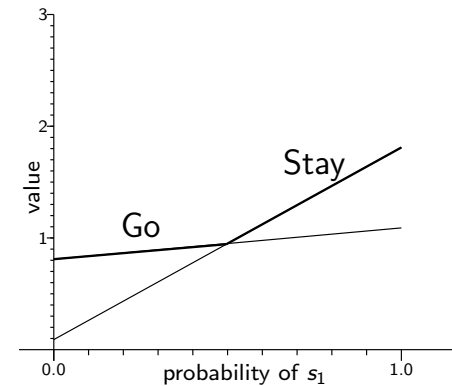
$$v_{Stay}(s_0) = 0.09$$

$$v_{Stay}(s_1) = 1.81$$

Values under **Stay**:



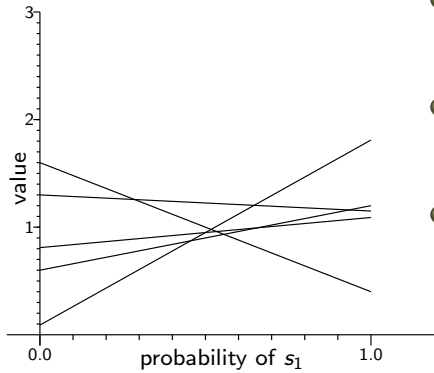
Values under optimal 1-step plans



For every belief state (probability distribution), use the plan that has the highest value.



Values under optimal plans



- Next iterations generate 2, 3, 4, ... step plans.
- **Dominated plans** (not best anywhere) **ignored**
- Optimal plan has a **piecewise-linear** value function consisting of **non-dominated segments** (top three in the diagram left)

Domination test

The following has a solution iff (w_1, \dots, w_n) **not dominated** in $\{(w_1^1, \dots, w_n^1), \dots, (w_1^N, \dots, w_n^N)\}$.
 (Vectors are values of π and members of U_0 in all states.)

$$\begin{aligned}
 p_1 + \dots + p_n &= 1 && (p_1, \dots, p_n \text{ represent a belief state}) \\
 p_1 &\geq 0 \\
 &\vdots \\
 p_n &\geq 0 \\
 w_1 p_1 + \dots + w_n p_n &\geq w_1^1 p_1 + \dots + w_n^1 p_n \\
 &\vdots \\
 w_1 p_1 + \dots + w_n p_n &\geq w_1^N p_1 + \dots + w_n^N p_n
 \end{aligned}$$

POMDP Value Iteration Algorithm

- 1 $i := 1, U' := \emptyset$
- 2 $U := U'$
- 3 $U_0 :=$ all policies of depth i
- 4 $U' :=$ policies in U_0 not **dominated** in U_0
- 5 $i := i + 1$
- 6 If improvement from U and U' "small", go to 2.

Dominated policy

π is **dominated** in U_0 if for every belief state B there is $\pi' \in U_0$ such that $v_{\pi'}(B) > v_{\pi}(B)$, where

$$v_{\pi}(B) = \sum_{s \in S} B(s) \cdot v_{\pi}(s)$$

Execution of a Policy

- 1 Belief state initially is some $B = (p_1, \dots, p_n)$
- 2 Find best plan π for belief state $B = (p_1, \dots, p_n)$ from value function
- 3 Execute **first action** a of π (rest of π **ignored!**)
- 4 Get observation e
- 5 Update B to B' according to a and e
- 6 Go to 2.

Auxiliary concept: Markov Chains

Definition (Markov chain $\langle S, P, R \rangle$)

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

Value function of a Markov chain can be evaluated by solving a set of linear inequalities.

(This is what we did when **evaluation a policy** in Policy Iteration: the Markov chain $\langle S, P, R \rangle$ with $P(s, s') = P_0(s, \pi(s), s')$ and $R(s, s') = R_0(s, \pi(s), s')$ was obtained from MDP $\langle S, A, P_0, R_0 \rangle$ and a policy π .)

Bounded-Memory Policies for POMDPs

Definition

A **bounded memory policy** with memory state M , for a POMDP $\langle S, A, P, R, E, O \rangle$ is a mapping $\pi : M \times E \rightarrow A \times M$.

The execution of π is a Markov chain $\langle S_x, P_x, R_x \rangle$ with

- $S_x = S \times M \times E$ (state, memory, last observation)
- $P_x((s, m, e), (s', m', e')) = P(s, a, s')O(a, s', e')$ where $\pi(m, e) = (a, m')$
- $R_x((s, m), (s', m')) = R(s, a, s')$ where $\pi(m, e) = (a, m')$

Bounded-Memory Policies for POMDPs

- Bounded-memory policies with **memory/belief states** $M = \{m_1, \dots, m_N\}$ instead of the **infinite** and **continuous** space of probability distributions over S
- Algorithms: Search in the (finite) space of bounded-memory policies



Bounded-Memory Policies for POMDPs

General scheme for finding bounded-memory policies:

- 1 Generate policies π
- 2 Evaluate the Markov chain corresponding to π
- 3 Choose the best policy

For example, choose policy π and initial memory state m to maximize

$$\sum_{s \in S} v_{\pi}(s, m, \text{obs}(s))B(s)$$

where B represents the initial *belief state* (unrelated to the initial memory state m) and $\text{obs}(s)$ is observations with initial states.



Execution of a Bounded-Memory Policy

- ① Memory state initially some $m \in M$ and initial observation $e = obs(e)$
- ② $(a, m') := \pi(m, e)$
- ③ Execute a , obtaining new observation e'
- ④ Update belief state $m := m'$ and $e := e'$
- ⑤ Go to 2.

Special Case: Memory-Free Policies

- Memory-free policy = Bounded-memory policy with 1 memory state
- Policies are mappings from **observations** to **actions**