01DRO1
Decision Making under Uncertainty

2017/2018

Partially Observable MDP.

Lecture 5

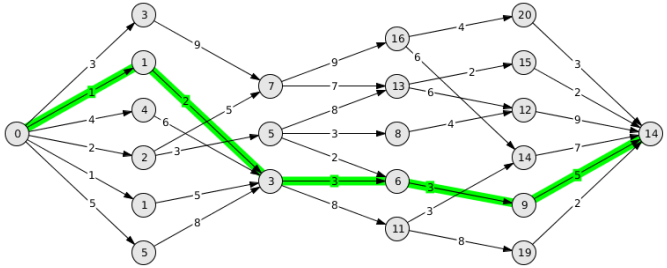
20.3.2018

Readings:

- M. Puterman, *Markov decision processes*, John Wiley & Sons, 1994,
- L.P.Kaelbling, M. L. Littman, A.R.Cassandra, Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101,99–134, 1998. Sec1-Sec 4.
- A.R.Cassandra Optimal Policies for partially Observable Markov Decision Processes 1995, Sec1-Sec4.
- K.J.Astrom “Optimal Control of Markov Processes with Incomplete State Information“ J. Math. Anal. Appl., 10,pp 174-205, 1965.

Where are we?

Dynamic programming



Dynamic programming is an optimization approach that transforms a complex problem into a sequence of simpler problems.

The principle of optimality: Any *optimal DM policy* has the property that, whatever the current state and decision, the remaining decisions must constitute an optimal policy with regard to the state resulting from the current decision.

MDP: solution approaches

Given (*S, A, T, r, h*)

Goal to find $\pi = \arg\max_{\pi} E\left[\sum_{t} r_t(s_{t+1}, a_t, s_t) \middle| \pi\right]$

- Finite Horizon Models
  - Dynamic Programming (Backward Induction)
- Infinite Horizon Models
  - Value Iteration
  - Policy Iteration (Modified Policy Iteration)
  - Linear Programming
  - Reinforcement Learning (Neuro-Dynamic Programming)

Value iteration

- Start with $V_0(s_{t+1}) = 0$ for all $s_{t+1} \in S$.
- for $k = 1, \ldots, h$

  Given V_k, calculate value update for all states $s_{t+1} \in S$:

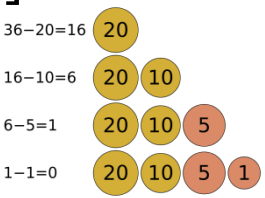
$$V_{k+1}^{opt}(s_t) = \max_{a \in A} \sum_{s_{t+1} \in S} T(s_{t+1}, a_t, s_t)\left[r(s_{t+1}, a_t, s_t) + V_k^{opt}(s_{t+1})\right]$$

  $V_k(s_{t+1})$ is optimal k-stages to go value function = expected sum of rewards accumulated when starting from state s_{t+1} and acting optimally for a horizon of k steps;

- Given V^{opt}, use *greedy policy*:

$$\pi^{opt}(s_t) = \arg\max_{a_t \in A} \sum_{s_{t+1} \in S} T(s_{t+1}, a_t, s_t)\left[r(s_{t+1}, a_t, s_t) + V^{opt}(s_{t+1})\right]$$

- Stop when $\left\| V_{k+1} - V_k \right\| < \varepsilon, \forall s \in S$



VI: Algorithm

Algorithm 1. The value-iteration algorithm for finite state space MDPs.

$V_1(s) := 0$ for all s
$t := 1$
loop
  $t := t + 1$
  loop for all $s \in \mathcal{S}$
    loop for all $a \in \mathcal{A}$
      $Q_t^a(s) := R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') V_{t-1}(s')$
    end loop
  $V_t(s) := \max_a Q_t^a(s)$
  end loop
until $|V_t(s) - V_{t-1}(s)| < \varepsilon$ for all $s \in \mathcal{S}$

L.P.Kaelbling, M. L. Littman, A.R.Cassandra, Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101,99–134, 1998.

Policy iteration (PI)

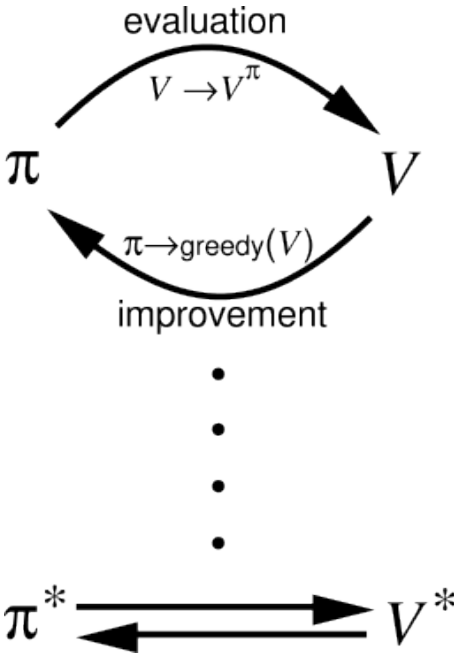
PI optimises the policy *directly* instead of optimising the value function and then inducing a policy.

Policy iteration consist of *repetition of 2 steps*:

1. *policy evaluation* – calculate value function for a fixed policy until convergence
2. *policy improvement* – calculate a new policy for each *s* using one-step-ahead prediction with resulting converged though not optimal value function

   (from step 1) as future utilities

Algorithm continues until no improvement possible at any state.



Linear Programming (LP)

- Recall $V^{opt}(s_t) = \max_{a \in A} \sum_{s_{t+1} \in S} T(s_{t+1}, a_t, s_t)\left[r(s_{t+1}, a_t, s_t) + \gamma V^{opt}(s_{t+1})\right], \forall s_t \in S$

- Optimal DM problem is formulated as linear programming, which can be solved efficiently using standard LP solvers.
- LP formulation: to find optimal V^{opt}

$$\min_V \sum_{s_{t+1} \in S} V(s_{t+1}), \text{ such that } \forall a_t \in A, \forall s_t \in S$$

$$V(s_t) \geq \sum_{s_{t+1} \in S} T(s_{t+1}, a_t, s_t)\left[r(s_{t+1}, a_t, s_t) + \gamma V^{opt}(s_{t+1})\right],$$

- Number of inequality constraints is |S|x|A|

Today..

Agent vs Environment

Agent

(*knowledge*, *observations*) → *actions*

sensors

observations → states

Environment

actions

actuators

built-in knowledge, DM objectives

Agent: human, artificial device
Environment: part of the world

Physical vs. digital or virtual worlds make no difference

What if..

- environment state is *stochastically* changed by actions?
- the states s_t are not fully observable?
- there is *no* certainty in action effects?
- there is *uncertainty* in knowledge of environment state?
- observations are *noisy* and not reliable?
- current state is *not* enough to make an optimal decision anymore?

REAL WORLD!

Our DM policy should respect that.

MDP vs. POMDP

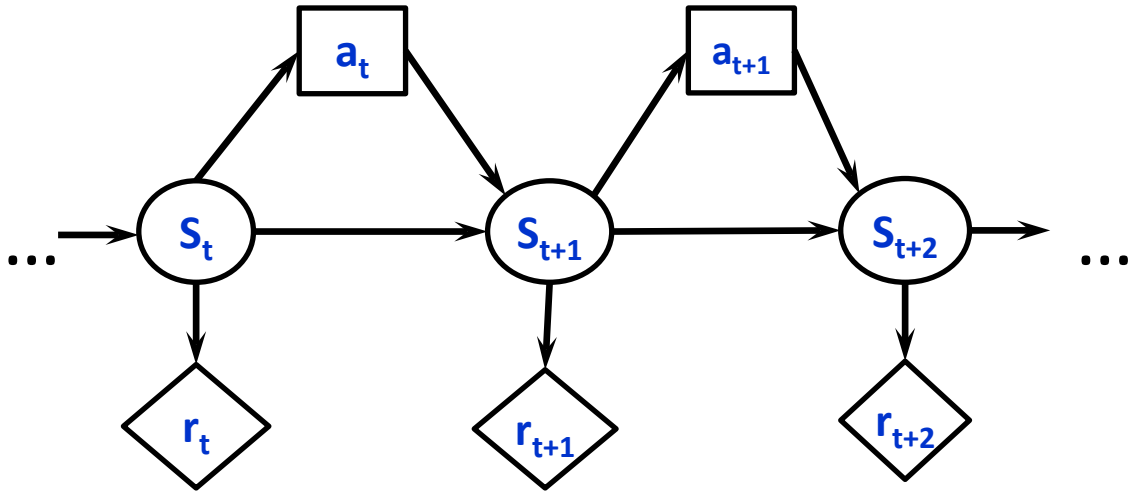
MDP (fully observable) :

- assumes that agent knows the state at each time step
- unrealistic in real-life applications
- policy $\pi : S \rightarrow A$ cannot be implemented if state is unknown
- optimal policy at s_t under uncertainty about true state can significantly differ from computed π

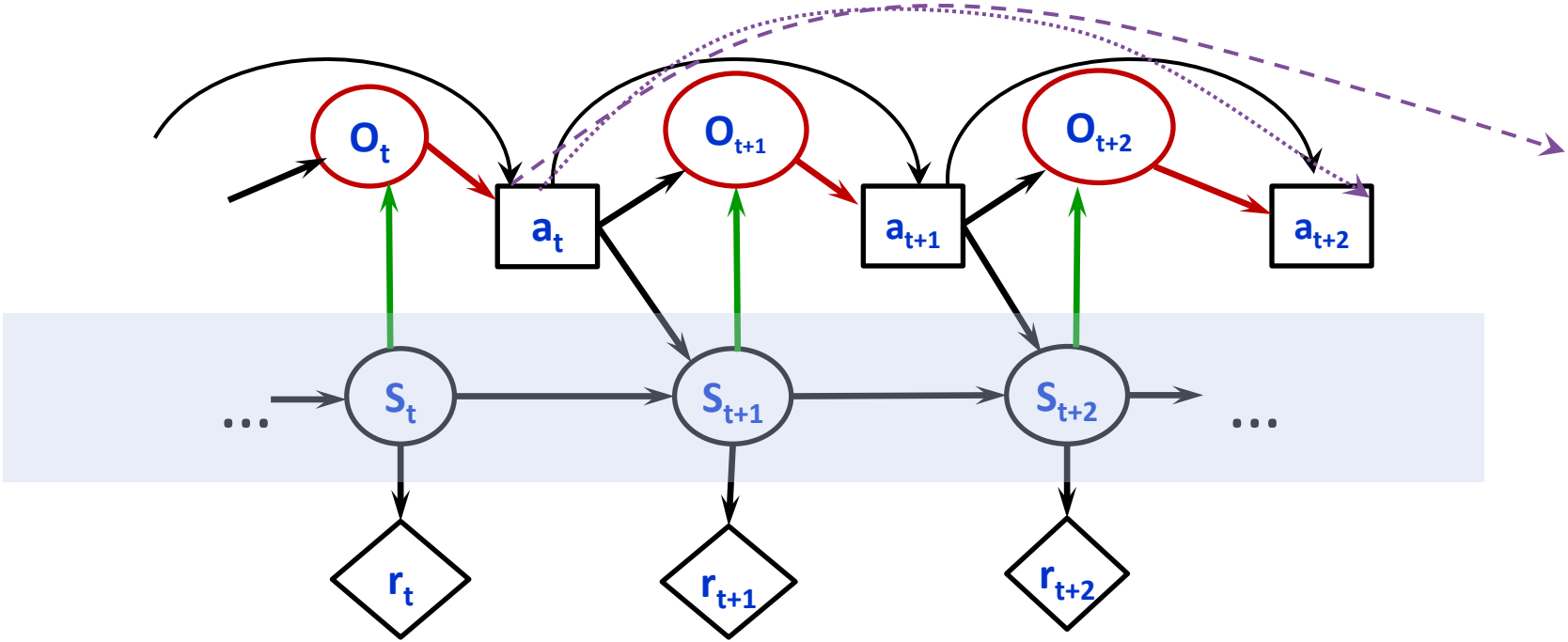
POMDP (Partially Observable MDP):

- Agent does not know state s_t => cannot use $\pi(s)$ as has no idea whether $s_t=s$
- Agent decision should be affected by how much it knows about s_t
- Agent gets noisy sensory measurements about (a part of) the state that are *correlated* with the underlying state

Fully observable MDP



Partially observable MDP



MDP vs. POMDP (cont.)

The uncertainty in system state distinguishes POMDP from MDPs:

*Agent does not get the info "*Environment is now in state *s",*

*instead he gets "*You now see the observation *o"*

MDP (Fully Observable) :

- relatively easy to specify
- tractable to solve

POMDP (Partially Observable MDP):

- treats all sources of uncertainty universally
- allows for learning decisions
- extremely intractable to solve optimally: much more complex compare to MPD, but world is more POMDP

PODMP: a wide range of potential applications

- robot navigation, helicopter/unmanned aerial vehicle control
- finance, economics, management
- preference elicitation
- stochastic resource allocation
- machine maintenance scheduling
- quality control
- ...*medical diagnosis, questionnaire design, moving target search, target identification, corporate policy, health-care policy making, human decision making (cognitive science),…*

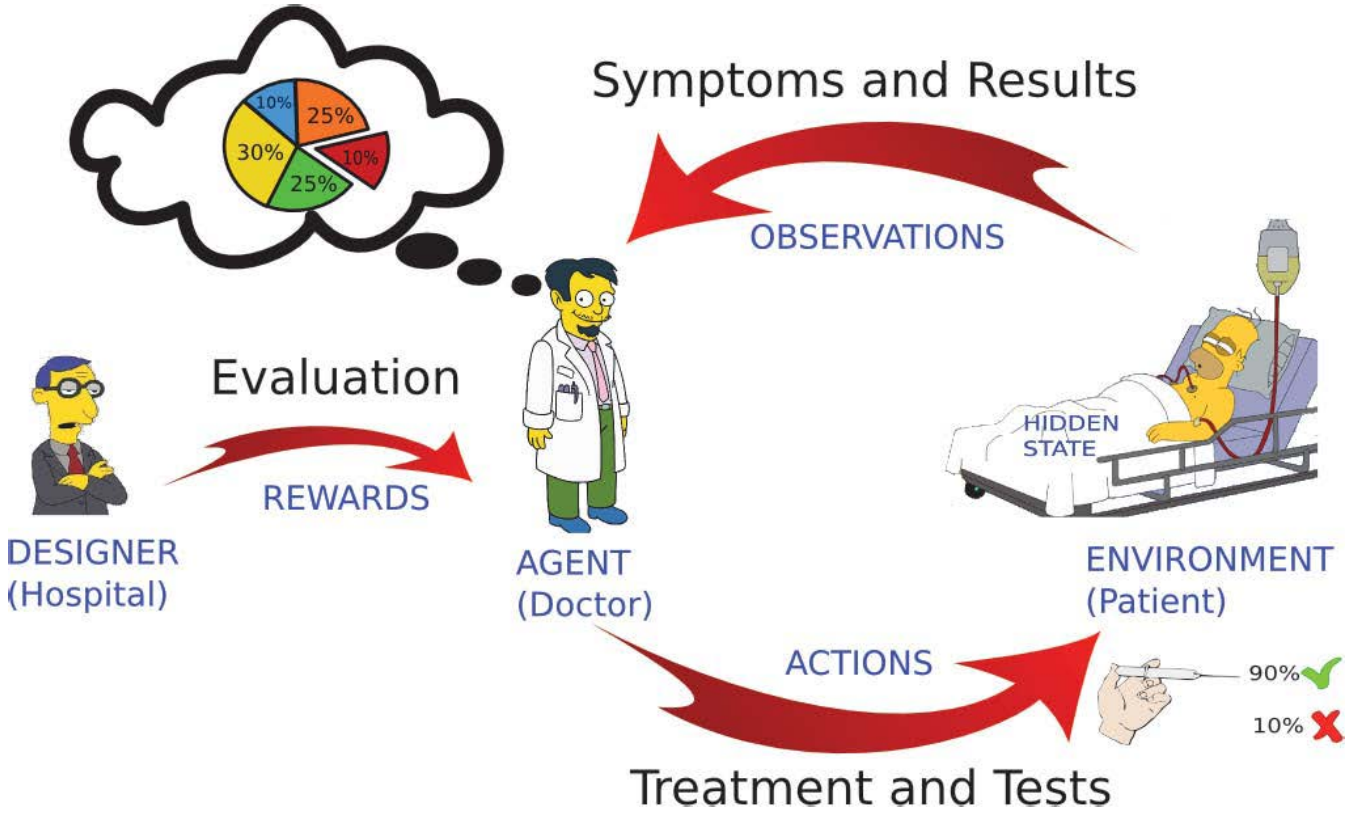
computational complexity => limited practical experience => job for you!

Related Models

Markov Models		Do we have control over the state transitons?	
		NO	YES
Are the states completely observable?	YES	**Markov Chain**	**MDP** Markov Decision Process
	NO	**HMM** Hidden Markov Model	**POMDP** Partially Observable Markov Decision Process

A.Casssandra et al ,AAAI-94 talk

Example



Mauricio Araya-Lopez, JFPDA 2013

POMDP = MDP + ???

MDP is defined by *(T, S, A, r):*

- *S* - finite set of *states*, $|S| = n$
- *A* - finite set of allowable *actions*, $|A| = m$
- *T* : S x S x A $\rightarrow$[0,1] - transition model $T(s_{t+1} | s_t, a_t)$
- *r*: S x A$\rightarrow \mathbb{R}$ - bounded, real-valued *reward function* $r(s_t, a_t)$
- h - horizon over which the agent will act

BUT: we cannot get the state itself, only noisy sensory observations!

POMDP = MDP + observation (sensor) model

MDP is defined by (T, S, A, r):

- S - finite set of *states*, $|S| = n$
- A - finite set of allowable *actions*, $|A| = m$
- $T : S \times S \times A \rightarrow [0,1]$ - transition model $T(s_{t+1} | s_t, a_t)$
- $r: S \times A \rightarrow \mathbb{R}$ - bounded, real-valued *reward function* $r(s_t, a_t)$
- h - horizon over which the agent will act
- O - finite set of observations, $|O| = l$
- $Pr : O \times S \times A \rightarrow [0,1]$ – *observation model,* probability $Pr(o_{t+1} | s_{t+1}, a_t)$ of observing o_{t+1} given that the agent took action a_t and reached state s_{t+1}. Sometimes it can be $Pr(o_{t+1} | s_{t+1})$ only.

The goal: to find $\pi: O \rightarrow A$ maximising expected future (discounted) reward .

Problems with POMDP

- Solution mapping states on actions is *not* possible
- Direct mapping observations on actions is *not possible*

- Idea: a policy π should map from some "*decision state*" to *actions* A.

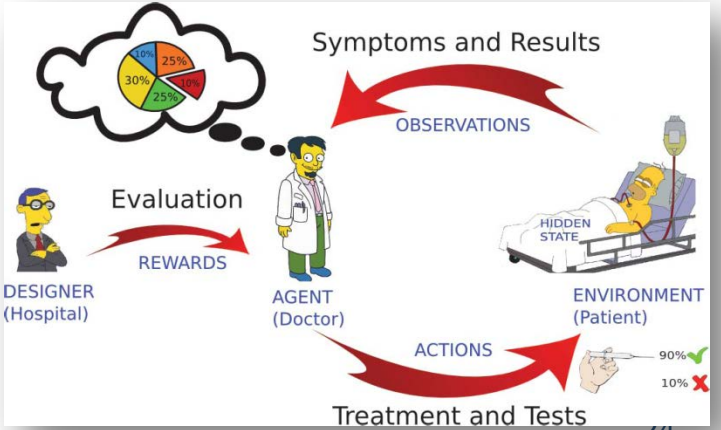
  Such a "*decision state*" can be defined either by:
  - available history $(a_0,o_0,\ldots,a_t,o_t)$, i.e. sequence of actions and observations

    but history grows exponentially **=>** *not* suitable for infinite horizon task
  - finite-state controller memory
  - some probability distribution over states

A POMDP extends an MDP by adding:

- Observation - a finite set of observations of the state, (also known as responses, diagnoses, perceptions, etc.)
- Observation function - models the relationship between the unknown state and the observations and can be action dependent, (also known as *sensor model*).

Note: set of observations ≠ the set of states. What you observe from a state is *not* the same as the state itself.



A POMDP extends an MDP by adding:

- Observation - a finite set of observations of the state, (also known as responses, diagnoses, perceptions, etc.)
- Observation function - models the relationship between the state and the observations and can be action dependent, (also known as sensor model).

Note: set of observations ≠ the set of states. What you observe from a state is *not* the same as the state itself.

How to model an unknown state?

- Belief state - probability distribution over *S* that reflects agent experience.

What is belief state?

- the most probable state given the agent experience, i.e. agent's subjective probability about the state based on either *personal experience* or some *prior knowledge*
- they comprise a *sufficient statistic* of the past history and agent's initial belief state, i.e. $(a_0, o_0, a_1, o_1, \ldots, a_t, o_t)$
- respects agent's degree of uncertainty
- preserves Markovian property
- information-gathering property of actions via obser. function
- Introduced by K J Åström in 1985.

see *"Optimal Control of Markov Processes with Incomplete State Information"*
J. Math. Anal. APPL., 10,pp 174-205, 1965.


Karl Johan Åström

Bayes belief state update

- Let $b_{t+1} = b_{t+1}(s_{t+1}) = \Pr(s_{t+1} | o_{t+1}, a_t, b_t)$, be belief state (degree of belief in s_{t+1})

  s.t. $0 \leq b(s) \leq 1, \forall s \in S, \sum_{s \in S} b(s) = 1$.

- Upon taking new action a_t and observing o_{t+1}, $b(s_t)$ should be updated, i.e. $(b_t\ a_t) \mid o_{t+1} \rightarrow b_{t+1}$.

$$b_{t+1}(s_{t+1}) = \Pr(s_{t+1} | o_{t+1}, a_t, b_t) = \frac{\Pr(o_{t+1} | s_{t+1}, a_t, b_t) \Pr(s_{t+1} | a_t, b_t)}{\Pr(o_{t+1} | a_t, b_t)}$$ Bayes' theorem

$$= \frac{\Pr(o_{t+1} | s_{t+1}, a_t) \sum_{s \in S} \Pr(s_{t+1} | a_t, b_t, s_t) \Pr(s_t | a_t, b_t)}{\Pr(o_{t+1} | a_t, b_t)}$$ chain rule

$$= \frac{\Pr(o_{t+1} | s_{t+1}, a_t) \sum_{s \in S} \Pr(s_{t+1} | a_t, b_t, s_t) b_t(s_t)}{\Pr(o_{t+1} | a_t, b_t)}$$ belief definition

$$\propto \Pr(o_{t+1} | s_{t+1}, a_t) \sum_{s \in S} \Pr(s_{t+1} | a_t, s_t) b_t(s_t)$$

Decomposition of POMDP

- Belief state updated by Bayesian conditioning is a sufficient statistic providing all relevant information about the history.
- We can define belief MDP with a state set consisting of all possible belief states (i.e. map unsolvable POMDP into MDP) as follows:
- B – a set of *belief* states - ***comprises original state space*** S
- A – a set of actions – ***no change***
- O – a set of observations – ***newly introduced***
- $\mathcal{T}$: B x B x A$\rightarrow$[0,1] – state transition model $Pr(b_{t+1} | b_t, a_t)$ - ***newly defined*** as $Pr(b_{t+1} | b_t, a_t) = \Sigma_{o \in O} Pr(b_{t+1} | o_{t+1}, b_t, a_t) Pr(o_{t+1} | b_t, a_t)$
- ρ: B x A$\rightarrow \mathbb{R}$ - bounded reward function on belief states $\rho(b_t, a_t)$ - ***new*** defined as $\rho(b_{t+1}, a_{t+1}) = \Sigma_{s \in S} Pr(b_{t+1} | o_{t+1}, b_t, a_t) r(s_{t+1}, a_t)$, where $r(s_{t+1}, a_t)$ is original reward

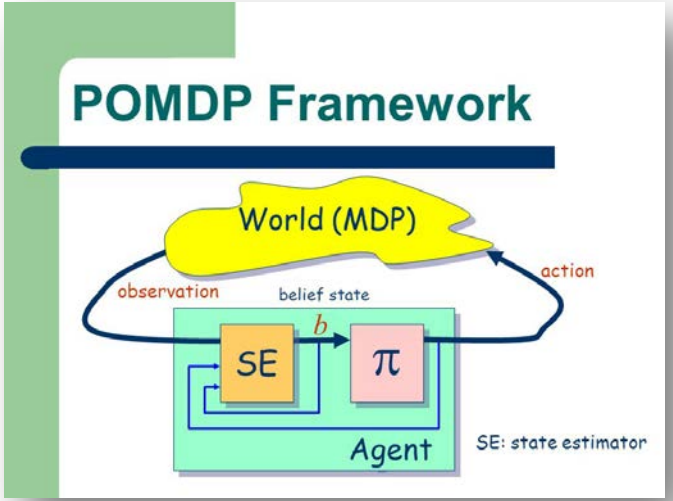
Remarks

- Agent receives observations related to unknown state
- Observation process, depends on (unknown)

  state, generated from a probability distribution function
- Hidden variables form a standard MDP
- Problem of hard POMDP is decomposed into:

  belief states MDP + **belief state estimate**

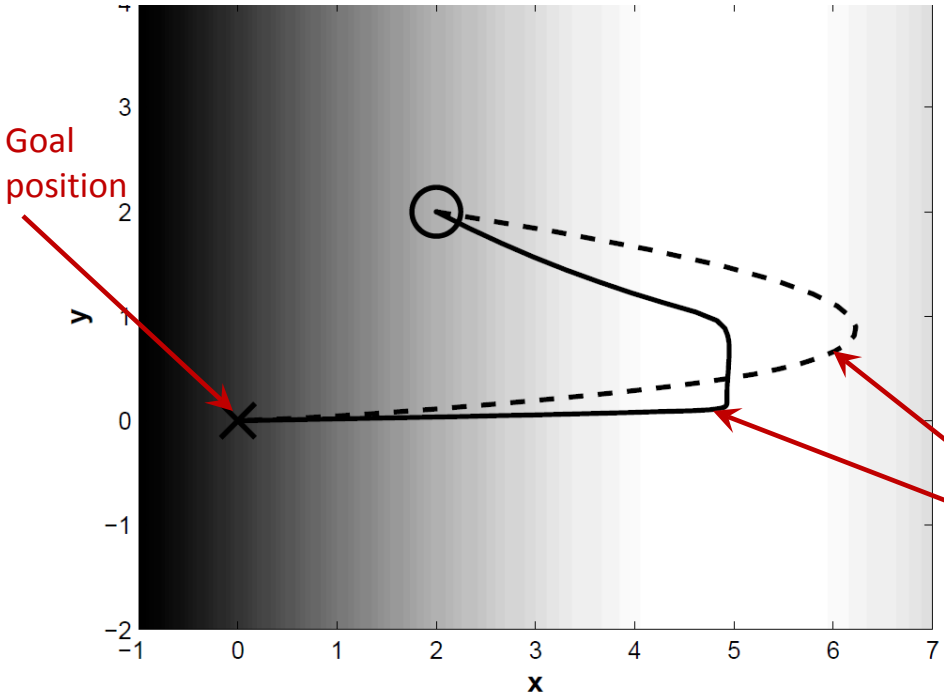
- fully observable
- continuous belief space



Solution will give optimal policy for the original infinite horizon POMDP.
But it is difficult to solve continuous space MDP generally.

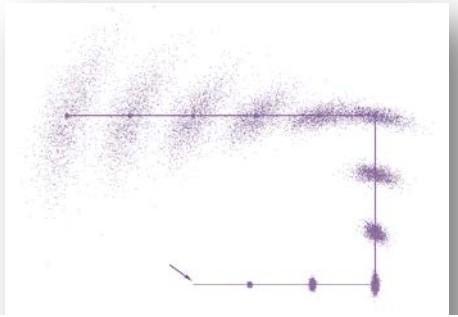
Model of Uncertainty: Robot example

Example taken from Platt,Tedrake, Kaelbling, Lozano-Perez, 2010



Example:

- Robot aim: to localise its position in light-dark domain.
- amount of light varies as quadratic function of horizontal coordinate
- ability to localise depends on amount of light
- to localize robot may need to move away from its goal position

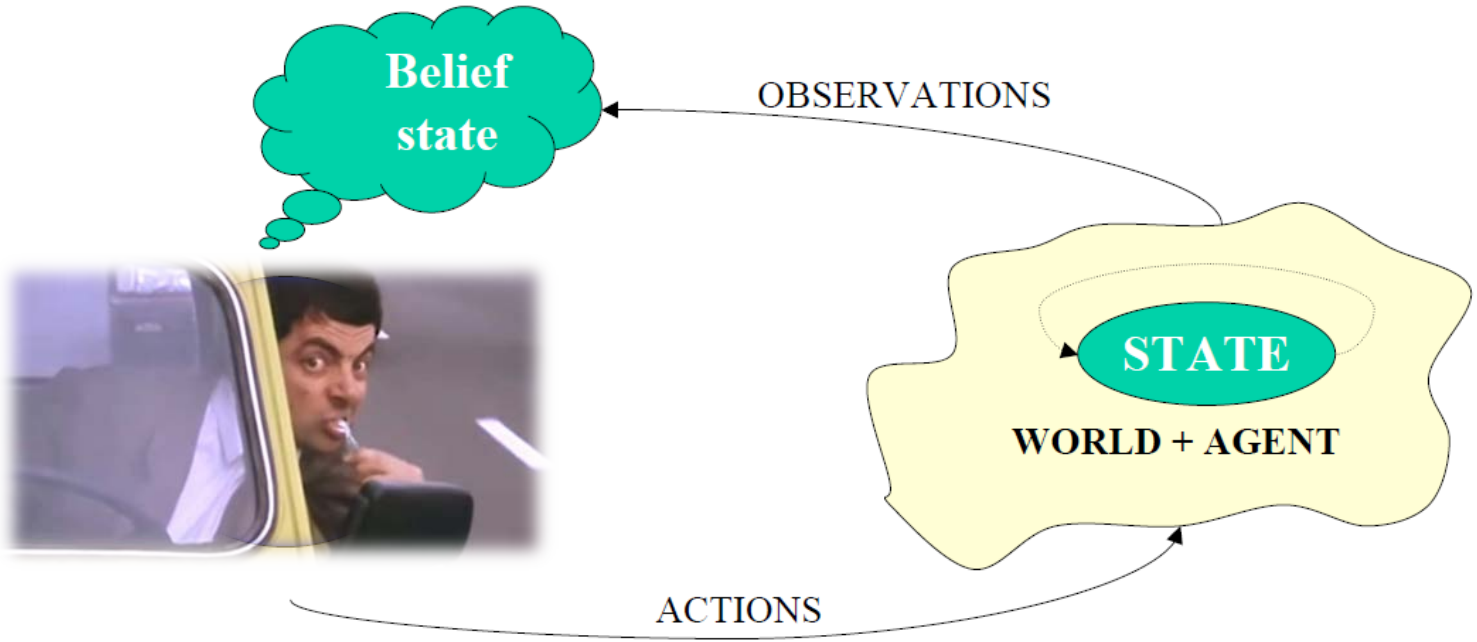


Gaussian model of uncertainty

The intensity illustrates the magnitude of the light .
The robot's initial position is unknown.

Belief is sufficient statistics for a given history

$b_t = Pr(s_t | b_0, a_1, o_1, \ldots, a_{t-1}, o_{t-1}, o_t)$



POMDP as a belief state MDP

- **Belief state** $b_t \in B(S)$ *sumarises* history $h^t = (a_1, o_1, \ldots, a_{t-1}, o_{t-1}, o_t)$
- Suppose we take action a_t and observe o_t
- **Observation model:** $Pr(o_{t+1} | s_{t+1}, a_t)$
- Let $(b_t\ a_t) \mid o_{t+1} \rightarrow b_{t+1.}$ be Bayesian **update of belief state**.

$$b_{t+1}(s_{t+1}) = \Pr(s_{t+1} | o_{t+1}, a_t, b_t) \propto \Pr(o_{t+1} | s_{t+1}, a_t) \sum_{s \in S} T(s_{t+1} | a_t, s_t) b_t(s_t)$$

  This is essentially filtering task.

- **Reward:** $\boldsymbol{R}(b_{t+1}, a_{t+1}) = \Sigma_{s \in S} Pr(b_{t+1} | o_{t+1}, b_t, a_t) r(s_{t+1}, a_t)$

Note: belief state MDP requires knowing the *environment* and *observation* models!

POMDP: Policy

- Knowledge available at time t:
  - initial distribution (belief state) $b \in B(S)$
  - history of actions and observations up to t: $h^t=(a_1,o_1,\ldots, a_{t-1},o_{t-1}, o_t)$

    cf. data vector Ψ → *regression vector* φ
- Goal: to find policy π maximising expected future (discounted) reward.
  - history-based policy π_t: $B(S) \times h^t \rightarrow A$ => non-Markovian, complexity grows exponentially with horizon
  - belief-based policy π_t: $B(S) \rightarrow A$ => Markovian, b 'summarises' h^t

POMDP: solution

Discretise the POMDP belief space and solve by:

- Exact solution methods
  - Value Iteration
  - Policy iteration
- Heuristic-based approaches
- Other

POMDP: optimal approach

Value Iteration (VI) can be used for solving continuous-state belief MDP, but

- the state space (beliefs) is the space of probability distributions (infinitely many number of belief states)
- value function and optimal action are computed for every possible probability distribution
- VI will trade off information-gathering actions vs actions affecting the state

The techniques for MDP cannot be directly applied => consider either approximating V or the belief state b, or both.

Policy evaluation

- Value of POMDP policy is given by expected sum of rewards:

$$V^{\pi}(b) = E\left[\sum_t \gamma^t R\left(b_t, \pi(b_t)\right)\right]$$

- Policy evaluation (Bellman's equation)

$$V^{\pi}(b_t) = R(b_t, \pi(b_t)) + \gamma \sum_{b_{t+1} \in B} \Pr(b_{t+1} | b_t, \pi(b_t)) V^{\pi}(b_{t+1})$$

Problem: Solving continuous space MDPs is very hard.

Uncountable amount of belief states => DP update is not of help as uncountable amount of equations of value function.

How to construct the optimal *h*-step value function over belief space using value iteration?

Way out: Consider the properties of value function to find an approximation.

see Example

Theorem on Policy Tree Value Function

Theorem: for |S|< ∞ value function $V^{\pi}(b)$ is *piecewise linear and convex* in b .

$$V^{\pi}(b) = \sum_{s \in S} \alpha(s) b(s)$$

Proof: by induction, see Sondik(1973)

Corollary: A policy made up of a set of trees is piece-wise linear.

Practical result: though the set of belief states is **continuous and infinite** a set of real number basis vectors of size $|S|$ can be used to represent V since DP preserves the piecewise linearity and convexity of the value function.

Policy tree

Policy tree p specifies what to do from the initial $b_0=\{b_0(0),\ldots, b_0(|S|-1)\}$ after each observation o up to horizon h. Given b_0 policy can be represented by a tree corresponding to a policy tree $p \in P$.

$V_p(s)$ is constant for tree p. Then for belief b

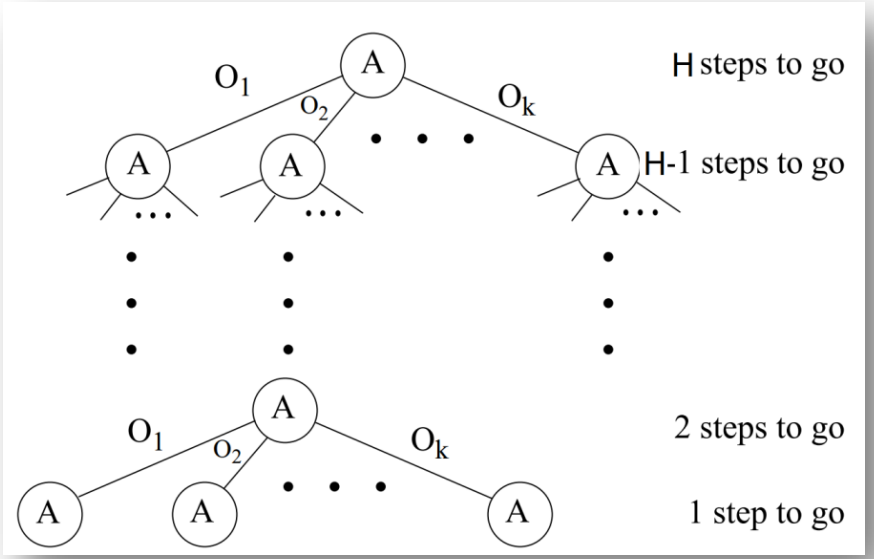
$$V_p(b) = \sum_{s\in S} b(s)V_p(s)$$

denoting $\alpha_p = \langle V_p(s_1),\ldots,V_p(s_n)\rangle$,

$$V_p^{\pi}(b) = \sum_{s\in S} \alpha_p(s)b(s)$$

and the best policy tree for b is

$$V^{opt}(b) = \max_{p\in P} V_p^{\pi}(b)$$



Solution algorithm

Unlike in classical VI we cannot enumerate all beliefs to compute V

Solution: Each policy tree p gives rise to a linear piece α, so instead we can compute linear pieces α for a subset of beliefs.

- Let $B = \{b_1, \ldots, b_k\}$ be a subset of beliefs
- Let $\Delta = \{\alpha_1, \ldots, \alpha_k\}$ be a set of α-vectors such that α_i is associated with b_i
- Repeatedly improve $V(b_i)$ at each b_i
- Find $\alpha_i(b)$ such that $V(b_i) = \sum_{s \in S} \alpha(s) b_i(s)$

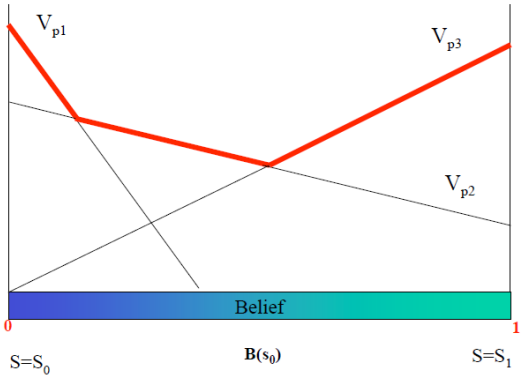
- α – linear function , called α-vector such that $V(b_i) = \max_{\alpha} \sum_{s \in S} \alpha(s) b\,(s)$

Interpretation as Policy Trees

- Each α_i corresponds to a *i*-step policy tree
- To implement policy given by set of policy trees (or α*−vectors)* exploit dynamic programming principle
  - find max vector for belief state *b*
  - execute action associated with vector *b*
  - observe new *o,* update belief *b*
  - repeat

Solution algorithm-optimisation

- The optimal policy is determined by projecting the optimal value function back down onto the belief space.
- The projection of the optimal value function => a partition into regions with a single maximal policy tree. The optimal action in that region is the action in the root of the policy tree *p*.



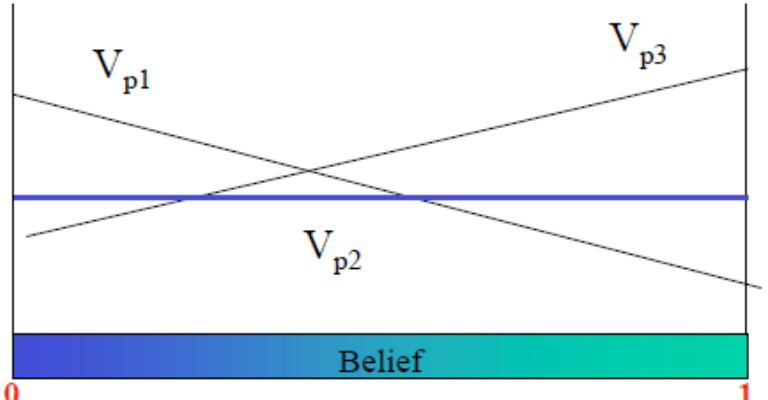
Solution algorithm - Pruning

- Some vectors (policy trees) are dominated and useless.
- That are not considered in the next step (pruned)
- Important: pruning dominated vectors before evaluating keeps set of α-vectors smaller => it decreases computation complexity
- *enumeration* algorithms generate vectors that are subsequently pruned (generate $\alpha 1$, prune; generate $\alpha 2$, prune; ..)
- *witness* methods add only potentially useful vectors

Given current approximate version α:

find *b* s.t. *V(b)> max{b* α*}* (*b* is a *witness*)

generate vector suitable for *b*, add to α



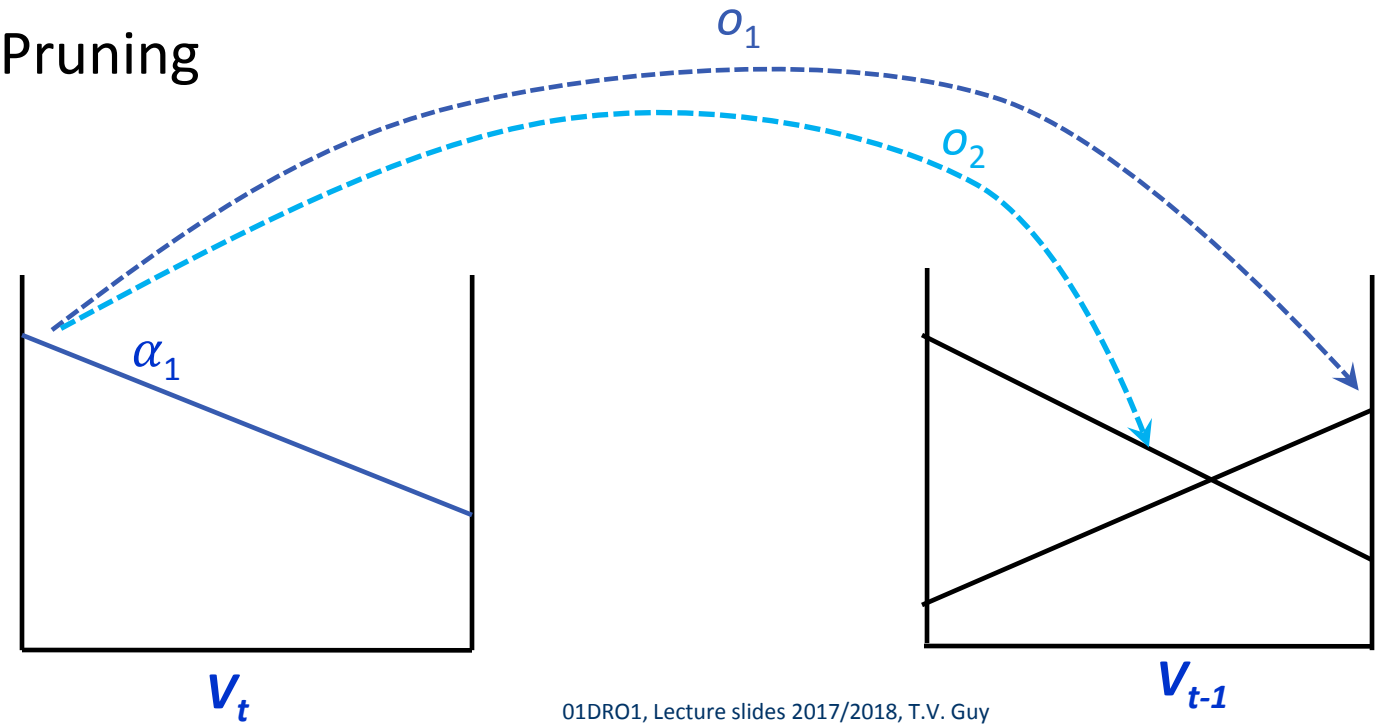
Importance of pruning

- Each update introduces additional linear components to *V*.
- Each measurement squares the number of linear components.
- Un-pruned *V* for *h=20* includes more than 10^{500000} linear functions.
- h=30 includes $10^{561,012,337}$ linear functions.
- The pruned *V* contains only 12 linear components!

This is main reason why this simple formulation of POMDP is impractical for most applications.

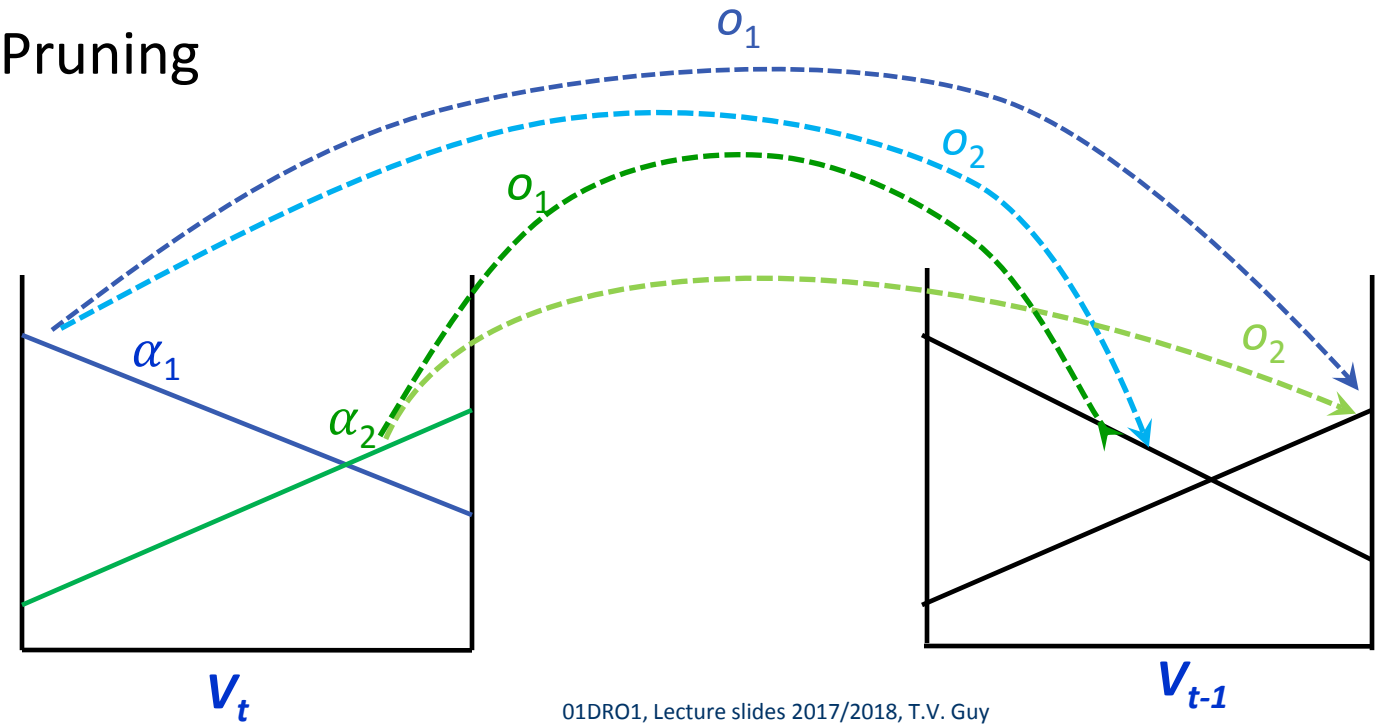
Graphic interpretation

- Dynamic Programming
- Pruning



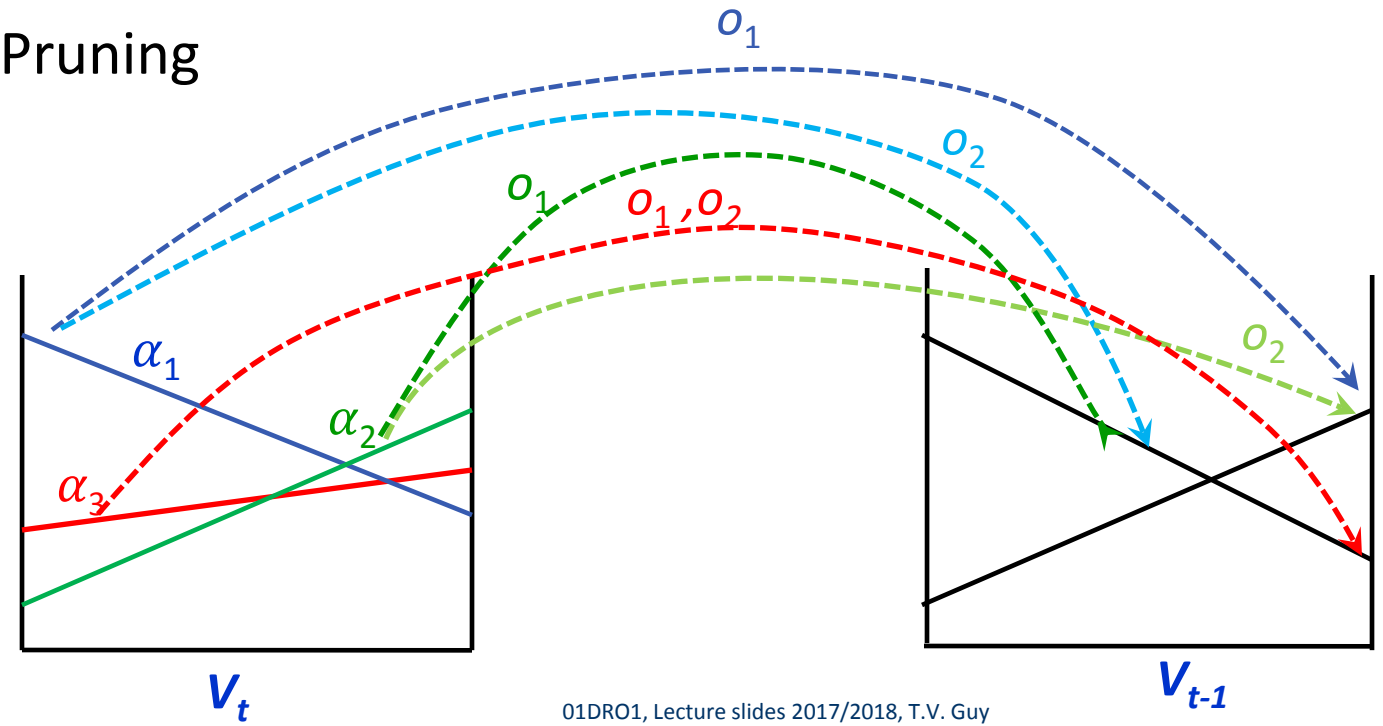
Graphic interpretation

- Dynamic Programming
- Pruning



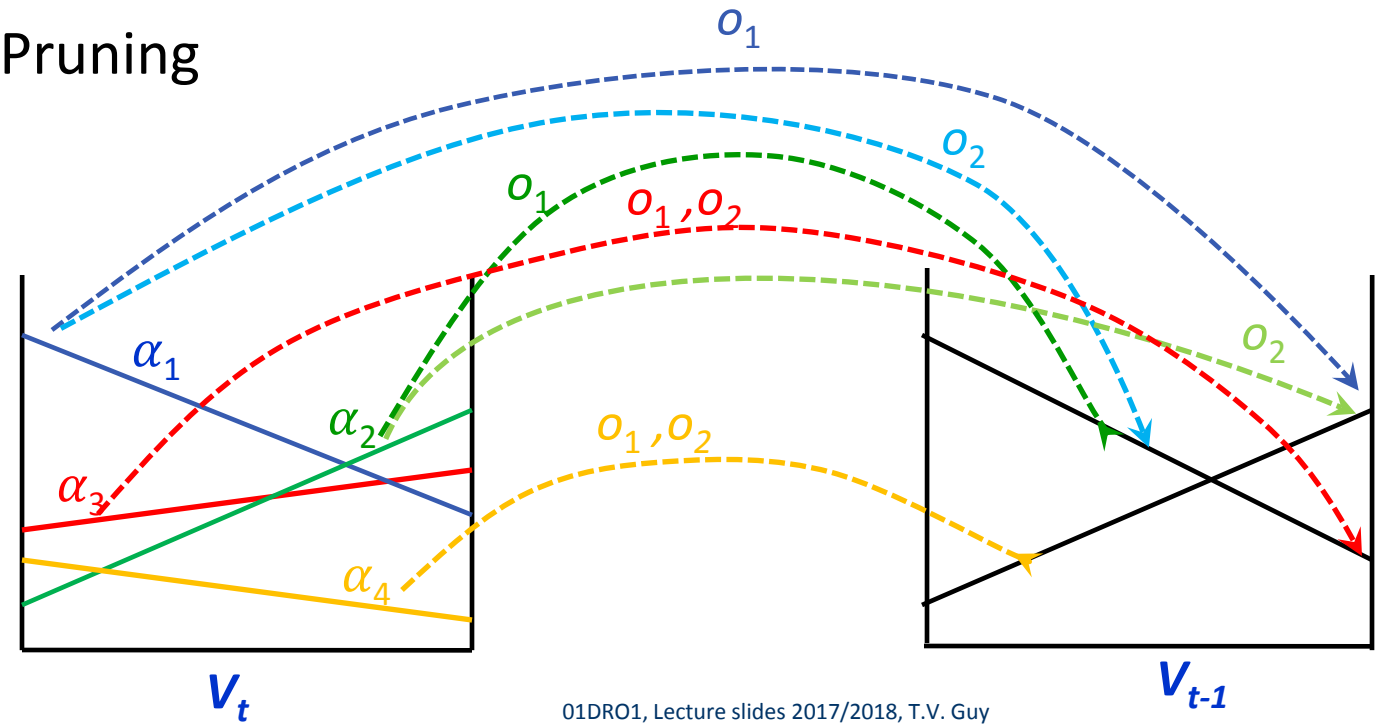
Graphic interpretation

- Dynamic Programming
- Pruning



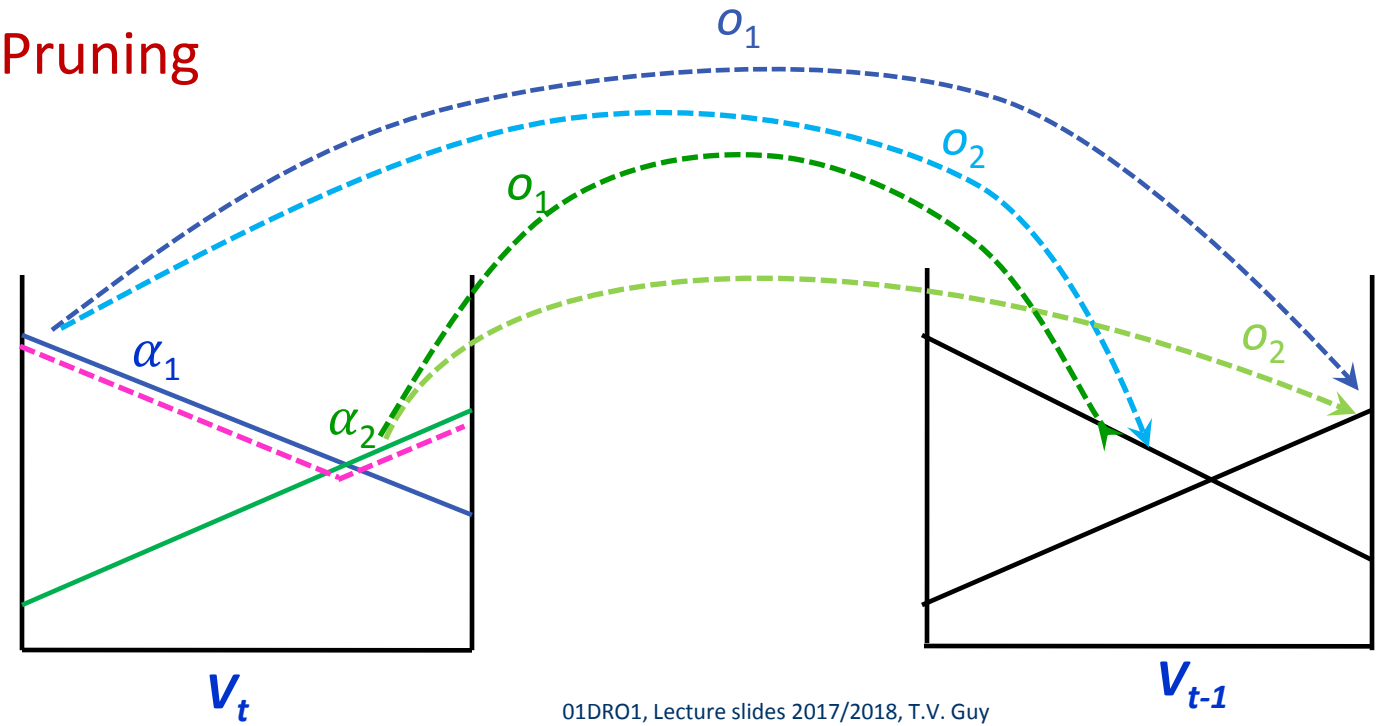
Graphic interpretation

- Dynamic Programming
- Pruning



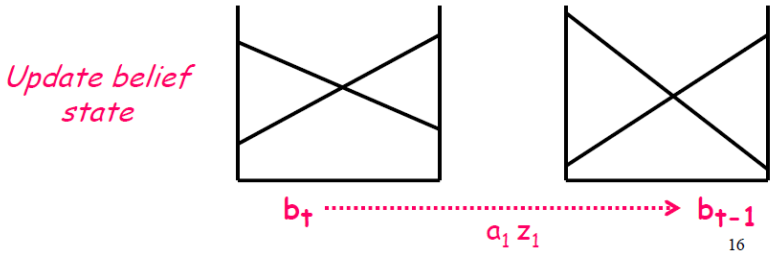
Graphic interpretation

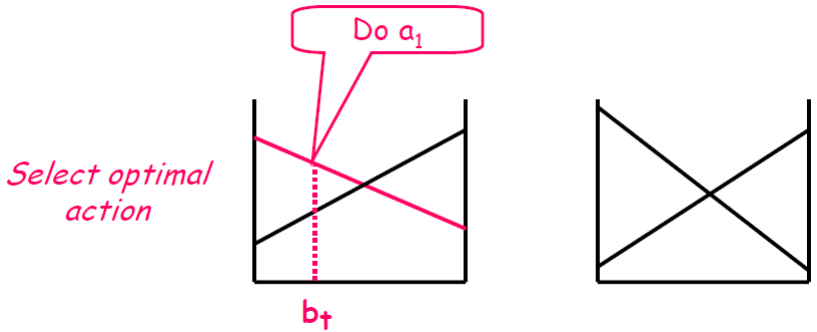
- Dynamic Programming
- Pruning

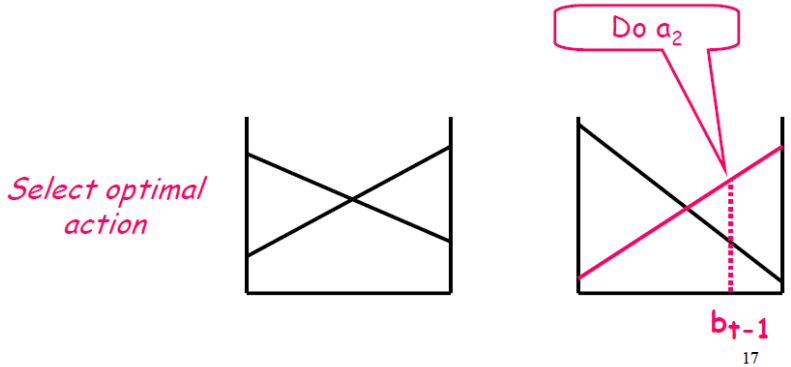


Policy execution

- Belief state monitoring
- Finite state controller







Sources of Intractability

- Size of α–vectors

  each is size of state space (exponential in number of variables)
- Number of α–vectors

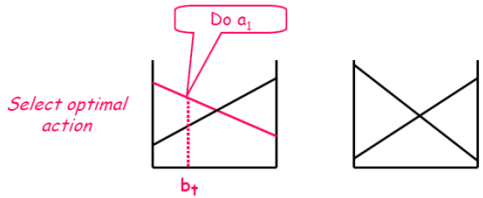
  potentially grows exponentially with horizon
- Belief state

  must maintain belief state online in order to implement policy

  using value function

  belief state is a repetition of state space

Value iteration for POMDP



☺

Value iteration is an exact method for determining the value function of POMDP.

The optimal action can be read from the value function for any belief state. Good for finite horizon

☹

- Time complexity is exponential in |O| and |A| when action choice depends on all previous observations
- Dimensionality of the belief space grows with number of states
- In the worst case, algorithm will end up considering every possible policy tree (hopefully it will converge before this happens)
- infinite horizon optimal VF: not always piecewise linear,

☺

In practice, VF still can be approximated by a finite-horizon VF if *h*is sufficiently large, so good policies based on subset of past observations can still be found

Approximation Strategies

- Exact solution is feasible only for small-dimensional (|S|<20,|O|<15) problems

  various approximation methods developed

  often deal with not more than 1000 states

- Grid-Based Approximations

  compute value function at small subset of belief space

  need reliable interpolation of value function

  need a method for grid selection

- Finite Memory Approximations

  policy as a function of most recent actions, observations

Approximation

- **Learning Methods**

  assume specific value function representation, for instance linear VF, smooth approximation

- **Heuristic Search Methods**

  provide search through belief space from an initial state

  quality of heuristic is decisive

  generally heuristics could be given by other methods

Heuristics based on the underlying MDP

Compute *V* for underlying MDP and heuristically combine it with *b*, for instance:

- Compute the most likely state $s^{opt} = \arg\max_{s \in S} b(s)$
- Define $\pi^{opt} = \pi^{MDP}(s^{opt})$

Problem: due to the assumption that uncertainty diminishes with time heuristic techniques do not perform information-gathering actions, *cf. certainty equivalence in control theory.*

Partial help: receding-horizon control

Connection Between HMM and POMDP

- Similar set up but different problem being solved
- Given an observation sequence, find the most likely hidden state sequence (tagging)

  No actions and rewards in HMM – thus you are not trying to find a optimal policy
- Length of sequence of observations in HMM is finite
- Start out with well-defined initial state
- Task: efficiently to find the state sequence that gives the highest probability to the observed outputs

Notes on POMDP solution

Computing *V* and *b* can be intractable (computing *V is doubly* exponential in the horizon time, and computing *b is exponential in the number of discrete state variables)*

To solve belief state MDP, we can consider computing:

1. the *exact* value function of the *exact* belief state.
2. an *approximate value function* of the exact belief state.
3. the *exact* value function for an *approximate* belief state.
4. an *approximate* value function for an *approximate* belief state.

- Redefine problem and convert environment dynamics into belief space dynamics (Bayesian filter)

- Do exact update of the belief state for *linear* dynamical systems => Kalman filter
- Approximate update for general systems => Particle filter