RL 13: POMDPs: Partially Observable Markov Decision Processes

Michael Herrmann

University of Edinburgh, School of Informatics

03/03/2015

A trace through an MDP

Environment:	You are in state 65. You have 4 possible actions.
Agent:	I'll take action 2.
Environment:	You received a reinforcement of 7 units. You are now in state 15. You have 3 possible actions.
Agent:	I'll take action 1.
Environment:	You received a reinforcement of -4 units. You are now in state 16. You have 2 possible actions.
Agent:	I'll take action 2.
Environment:	You received a reinforcement of 8 units. You are now in state 15. You have 3 possible actions.
⋮	⋮

How is this different for a POMDP?

Types of Planning Problems

	State	Action Model
Classical Planning	observable	deterministic accurate
MDP	observable	stochastic
POMDP	partially observable	stochastic

Two types of uncertainty

- Stochasticity: Only parameters of a distribution can be known
- Partial observability:
  - There is an underlying deterministic process that in principle can be inferred
  - This deterministic process may govern the parameters of a stochastic process

Background: COMDPs vs POMDPs

Same: set of states and actions, transitions and immediate rewards.

Different:

- Previously: (regular) discrete MDPs $\longrightarrow$ completely observable (COMDPs)
- Value iteration algorithm for COMDPs gives a value per state
- accurate state information is available
- Markovian

- POMDPs are also discrete MDPs.
- No certainty about the current state
- How represent values and actions?
- Probabilistic observations replace explicit state information
- Observation model needed: Bayesian estimation of states
- Taking into account information about previous states: Non-Markovian for states, but Markovian in terms of belief states.

Reminder (Bellman for MDPs)

Bellman optimality equation for states s and actions a

$$V^*(s) = \max_a \sum_{s'} P^a_{ss'} \left(R^a_{ss'} + \gamma V^* \left(s' \right) \right)$$

Usually we can assume that $R^a_{ss'} = r\left(s, a\right)$ (i.e. independent on next state)

$$V^*(s) = \max_a \left(r\left(s, a\right) + \gamma \sum_{s'} P^a_{ss'} V^* \left(s' \right) \right)$$

For continuous states x and actions u this becomes

$$V^*(x) = \max_u \left(r\left(x, u\right) + \gamma \int p\left(x' | x, u\right) V^* \left(x' \right) dx' \right)$$

Value iteration

$$V_t(x) = \max_u \left(r\left(x, u\right) + \gamma \int p\left(x' | x, u\right) V_{t-1} \left(x' \right) dx' \right)$$

POMDPs: Beliefs instead of state information

- Generalisation of COMDPs: POMDPs
- State is not observable: Agent relies on beliefs about its state
- A belief is a probability distribution over states.
- Simple example: Assume two states (1 and 2),
  - the agent could, e.g., have the belief $b_1 = .95$ to be in state 1
  - For consistency, we will assume that $b_2 = 1 - b_1 = 0.05$
  - From the point of view of the agent, the state is now described by the parameter b_1
  - If the agent correctly believes that $b_1 = 1$ or that $b_1 = 0$, we have the special case of a COMDP
- If the belief state is nontrivial, the agent will either decide under uncertainty or can choose to reduce uncertainty.

Bellman for belief states

- Define the belief b of the agent about the its state and formulate a POMDP with a value function over a belief space:

$$V_t(b) = \max_u \left(r(b,u) + \int V_{t+1}(b')\, p(b'|u,b)\, db' \right)$$

- Bayesian belief propagation (given action a):

$$b'(s') = \frac{\Omega(o \mid s', a) \sum_{s \in S} T(s' \mid s, a) b(s)}{\sum_{s'' \in S} \Omega(o \mid s'', a) \sum_{s \in S} T(s'' \mid s, a) b(s)}$$

  where s are the previous states with distribution b, s' the new states with distribution b', T the actual state transitions, and Ω the actual observation probabilities for signals o.
- Usually, T increases uncertainty, Ω reduces uncertainty.

POMDP: Formal definition and belief propagation

A POMDP is a tupel (S, A, O, T, Ω, R), where

S is a set of states,

A is a set of actions,

O is a set of observations,

T is a set of conditional transition probabilities,

Ω is a set of conditional observation probabilities,

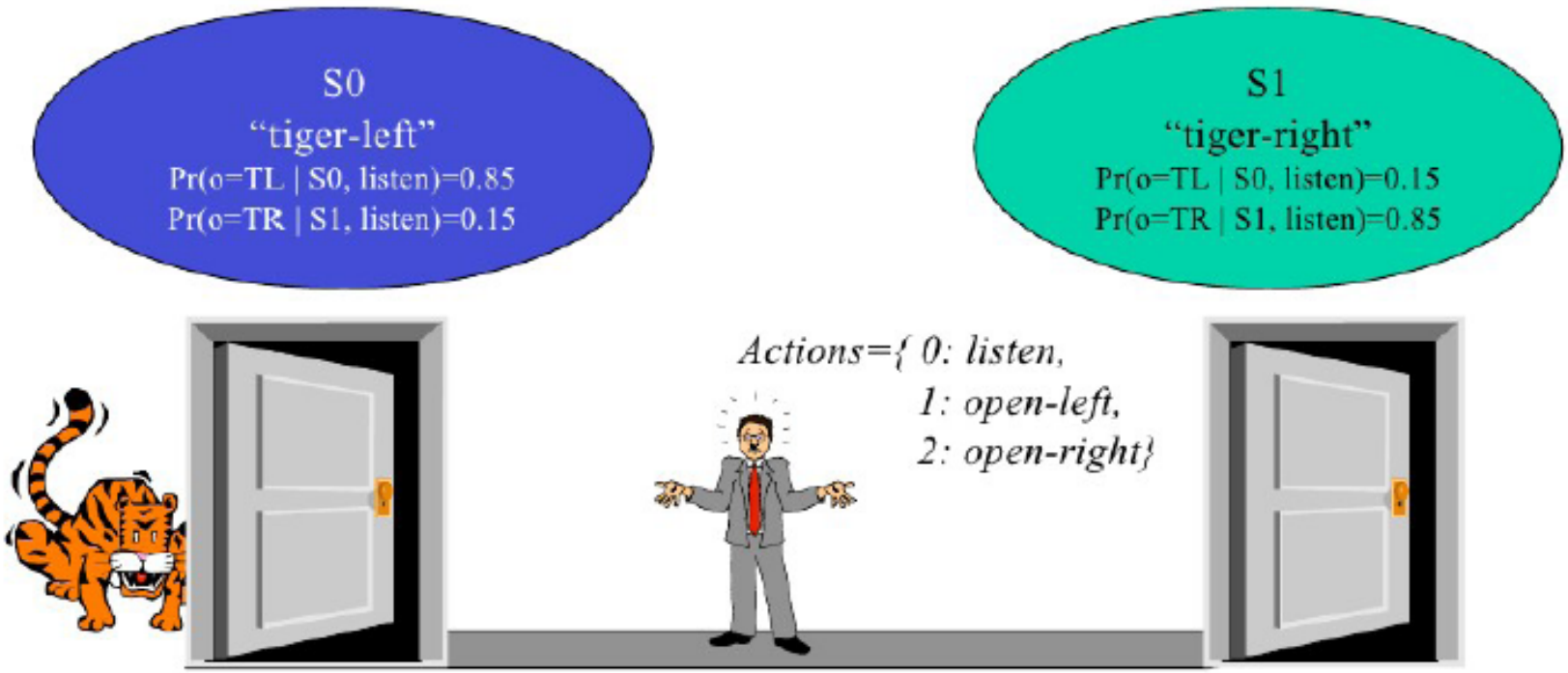
$R : A \times S \to \mathbb{R}$ is the reward function

Consequences

- As beliefs are probability distributions, POMDPs involve functions on (continuous) probability distributions
- Belief spaces are generally huge
- Because of continuity, belief spaces have a relatively simple structure
- If we assume
  - finite state space
  - finite action spaces
  - finite horizons

then we can represent value functions by piece-wise linear functions

The Tiger problem
(from: Dr. Stephan Timmer "Introduction to POMDPs")



Reward Function

- *Penalty for wrong opening: -100*
- *Reward for correct opening: +10*
- *Cost for listening action: -1*

Observations

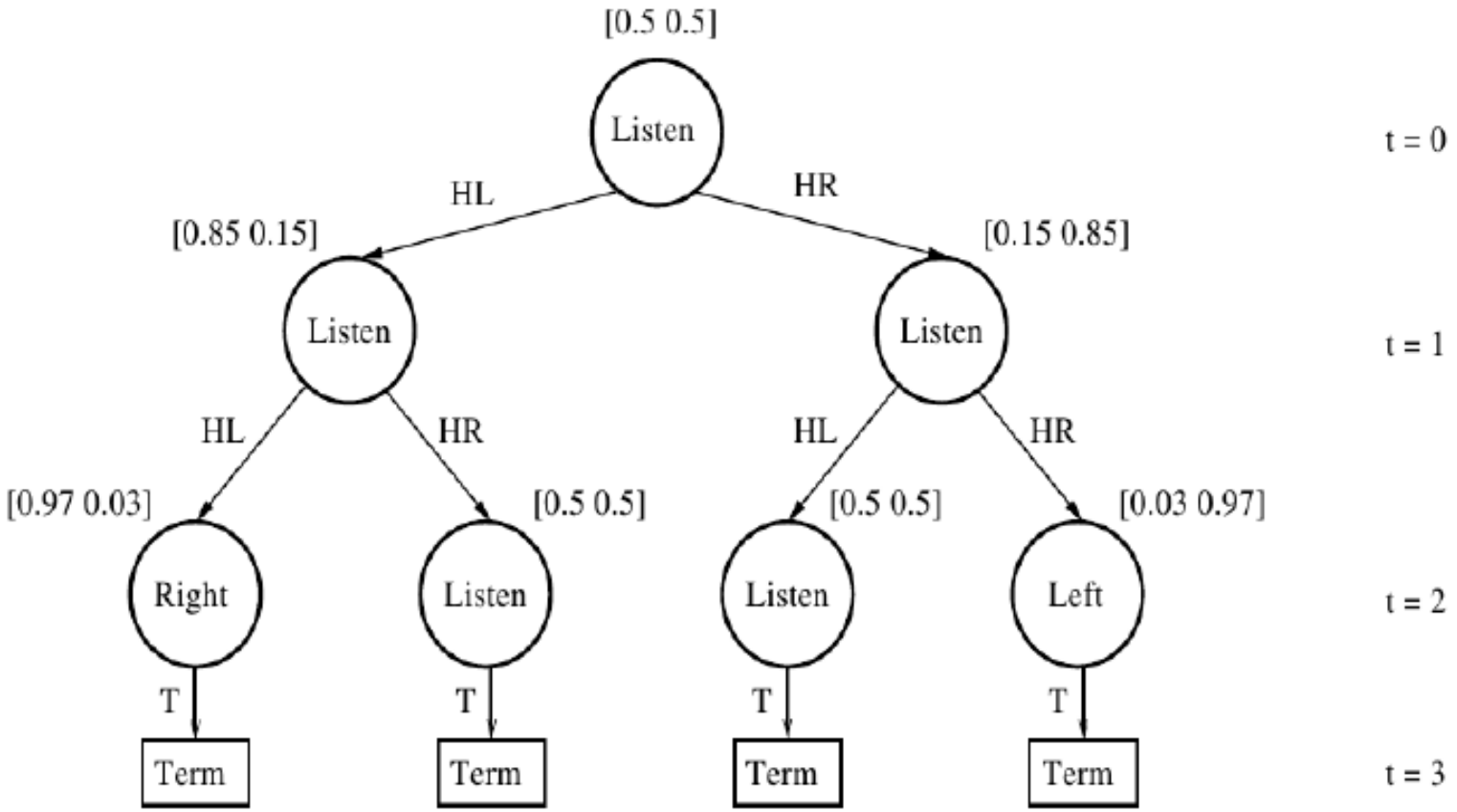
- *to hear the tiger on the left (TL)*
- *to hear the tiger on the right(TR)*

The Tiger problem

```
# This is the tiger problem of AAAI paper fame in the new POMDP
# format. This format is still experimental and subject to change

discount: 0.75
values: reward


        states: tiger-left tiger-right
        actions: listen, open-left, open-right
        observations: tiger-left, tiger-right

        Transitions:
        listen -> identity
        open-left -> uniform
        open-right -> uniform

        Observations
        listen (in either state):     0.85 0.15
                                      0.15 0.85
        open-left: uniform
        open-right: uniform

        Rewards:
        R:listen : * : * : * -1
        R:open-left : tiger-left : * : * -100
        R:open-left : tiger-right : * : * 10
        R:open-right : tiger-left : * : * 10
        R:open-right : tiger-right : * : * -100
```

The Tiger problem

Initialise beliefs by [0.5 0.5]: Equal probability for Tiger left or right.



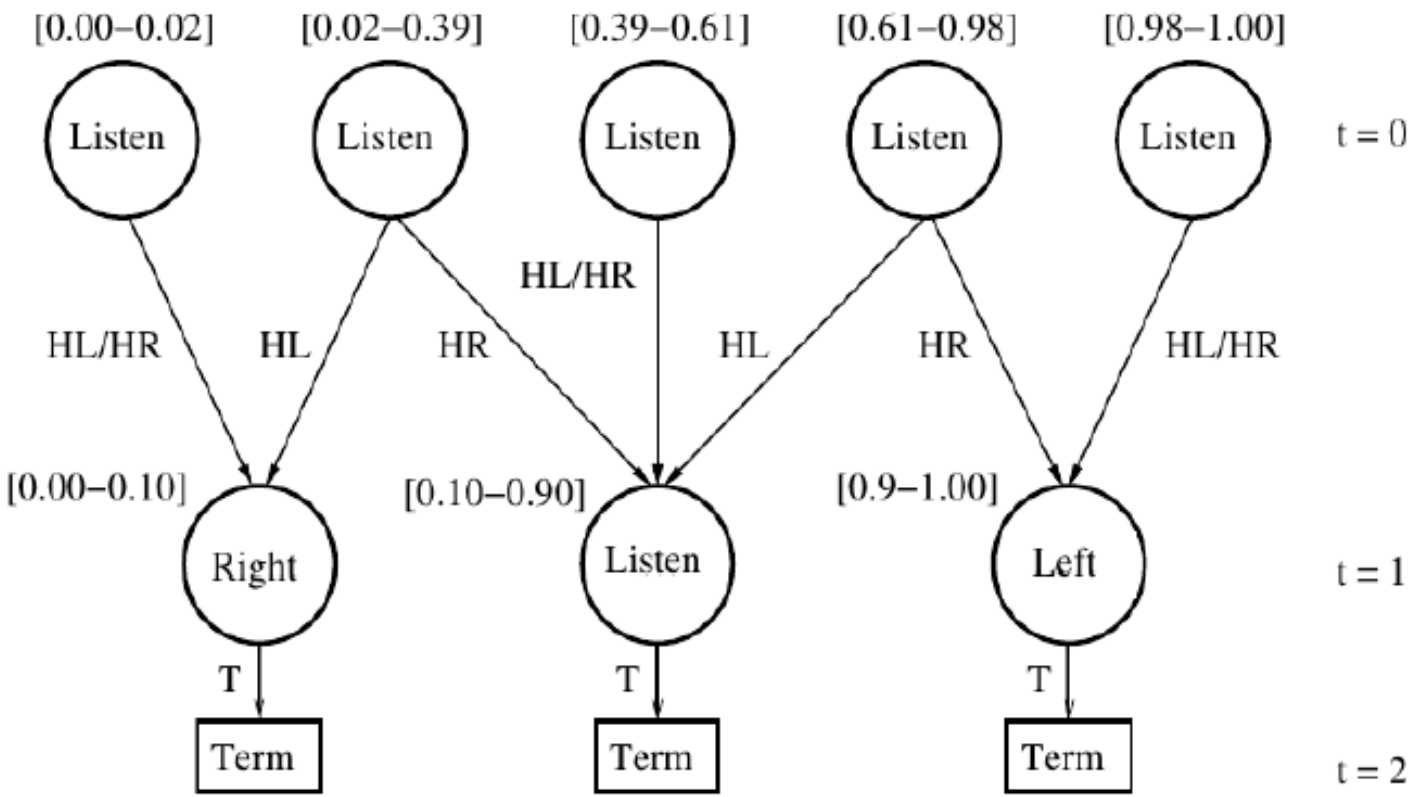
Two-step look ahead: If observation is twice "HearLeft", model-based belief of "Left" is 0.97 and action "Right" appears to be save.

Learning in POMDPs

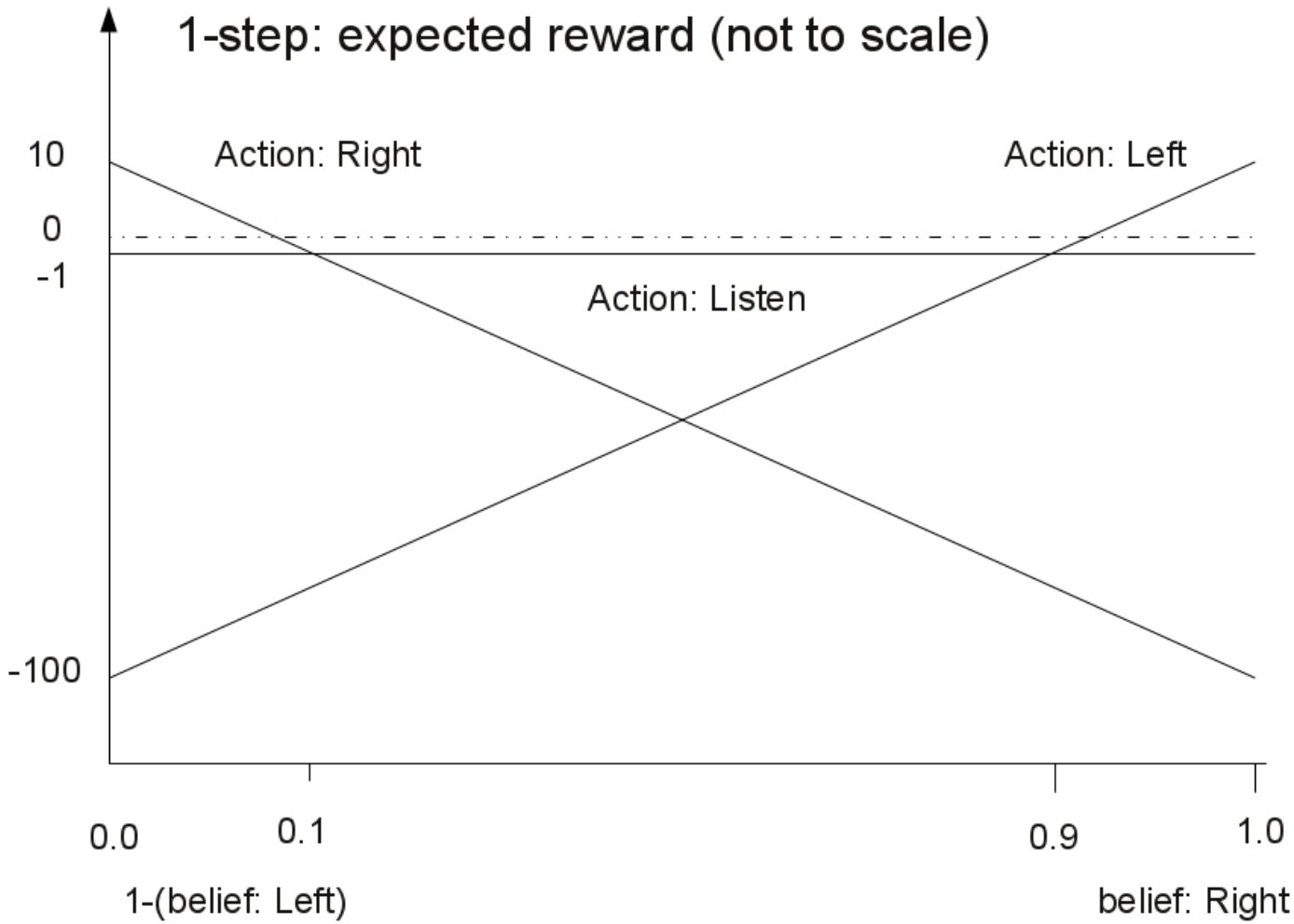
- As always: Choose policy in order to maximise expected reward
- Reward expectation will be based on current belief
- Observations affect belief and thus expectations
- For discrete observations:
  - at each step several possibilities
  - policy branches according to observation
  - policy tree! (grows exponentially $\rightarrow$ finite horizon)

The Tiger problem

Value of the "Listen" action, starting from an arbitrary belief state:

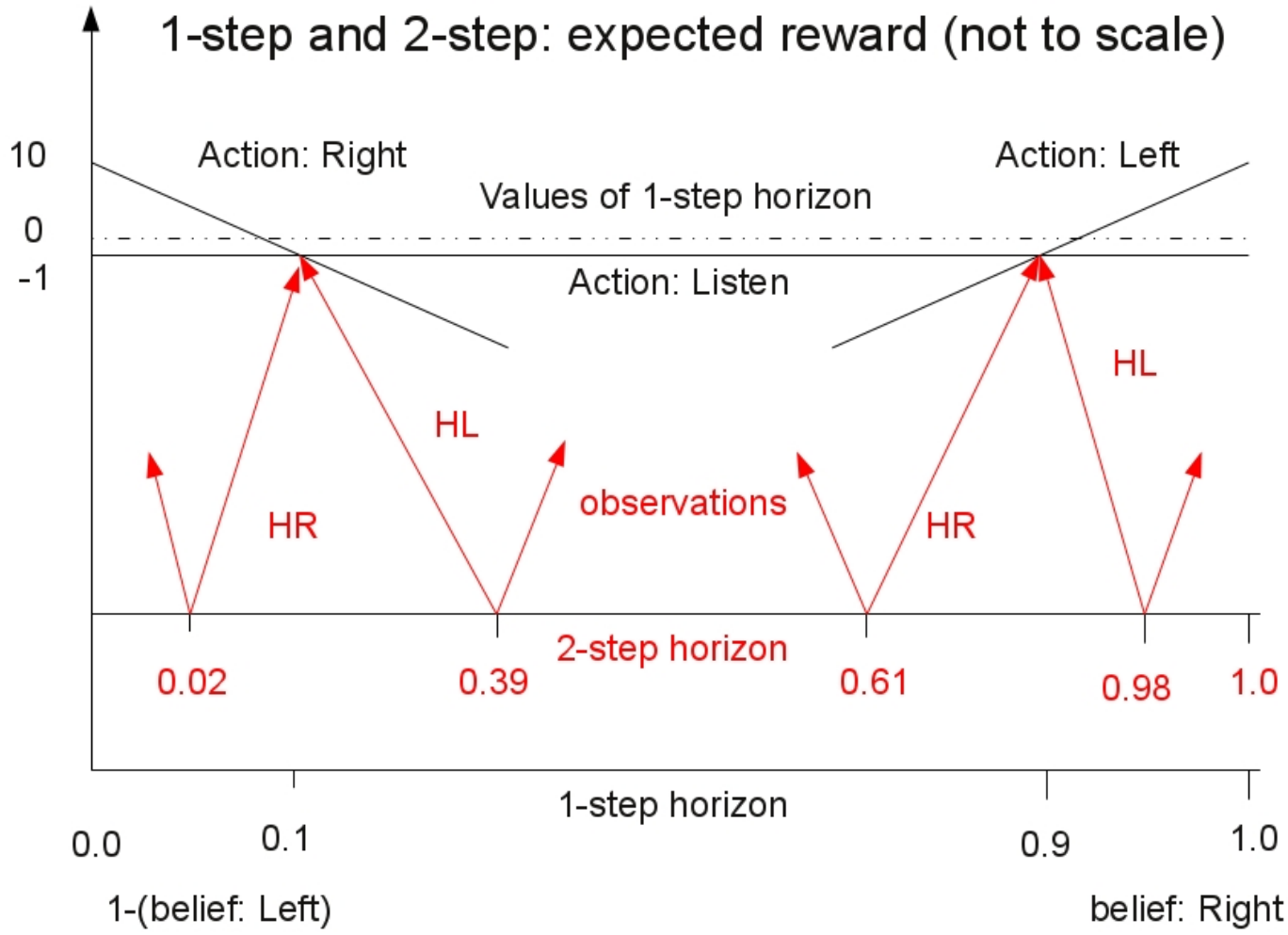


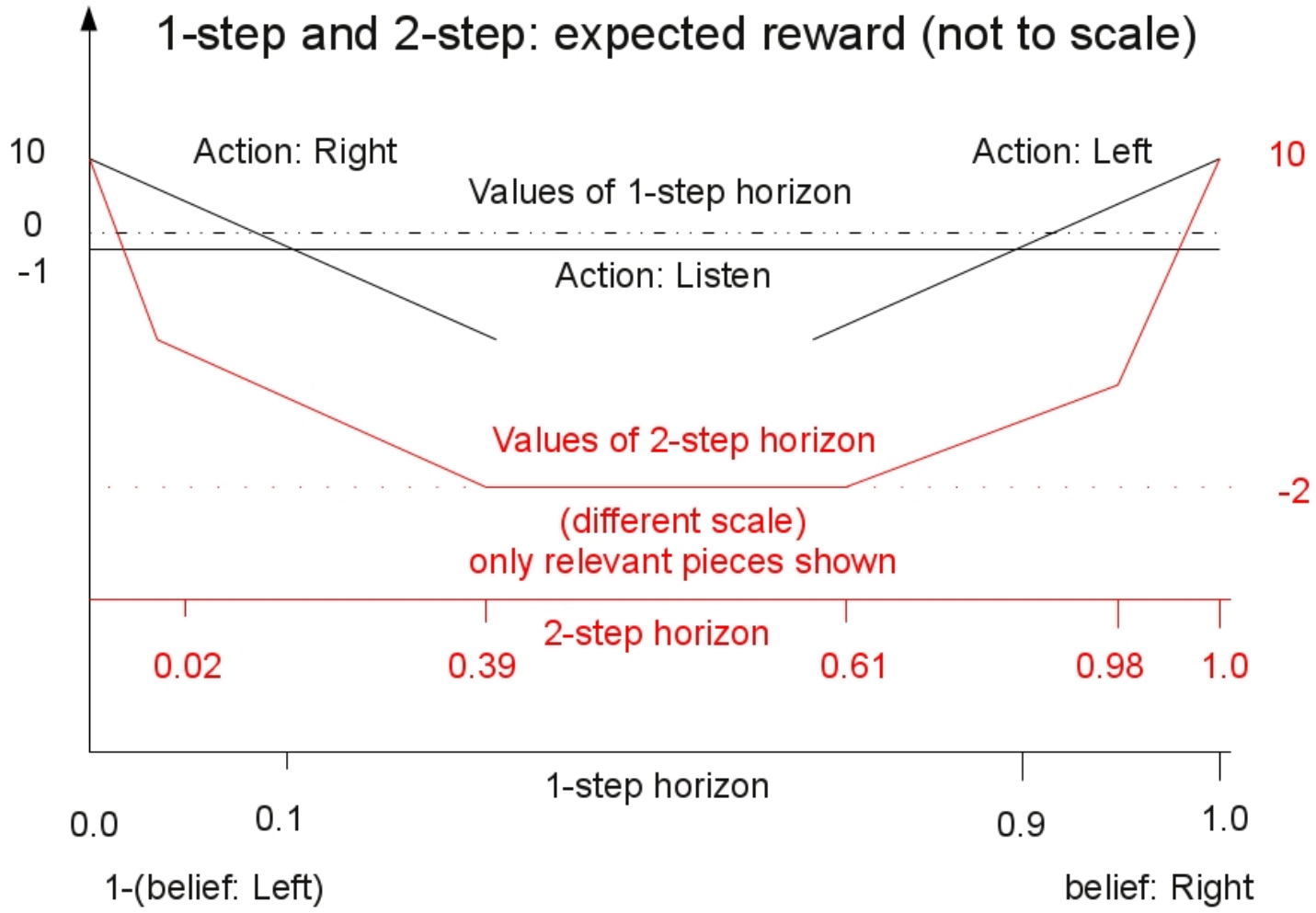
Ranges at $t = 0$ are implied by the reward ratio for the two states: If the belief in "Tiger-Right" is less than $\frac{1}{10}$ then action "open-Right" is better.


1-step: expected reward (not to scale)
10
0
-1
-100
Action: Right
Action: Left
Action: Listen
0.0
0.1
0.9
1.0
1-(belief: Left)
belief: Right

Multi-step prediction

- Value is estimated based on next-step values (and immediate reward)
- Present value: Present reward + expected reward
  - to include average over next observations (over a tree of finite length)
  - including the resulting change in belief
- We are talking about an MDP, i.e. all probabilities are known. In practice the agent still has to find it out by sampling.
- Values as a function of belief are piece-wise linear and convex: Linear Programming!


1-step and 2-step: expected reward (not to scale)
10
0
-1
Action: Right
Action: Left
Values of 1-step horizon
Action: Listen
HL
HL
HR
HR
observations
2-step horizon
0.02
0.39
0.61
0.98
1.0
1-step horizon
0.0
0.1
0.9
1.0
1-(belief: Left)
belief: Right


1-step and 2-step: expected reward (not to scale)
10
0
-1
Action: Right
Action: Left
10
Values of 1-step horizon
Action: Listen
Values of 2-step horizon
-2
(different scale)
only relevant pieces shown
2-step horizon
0.02
0.39
0.61
0.98
1.0
1-step horizon
0.0
0.1
0.9
1.0
1-(belief: Left)
belief: Right

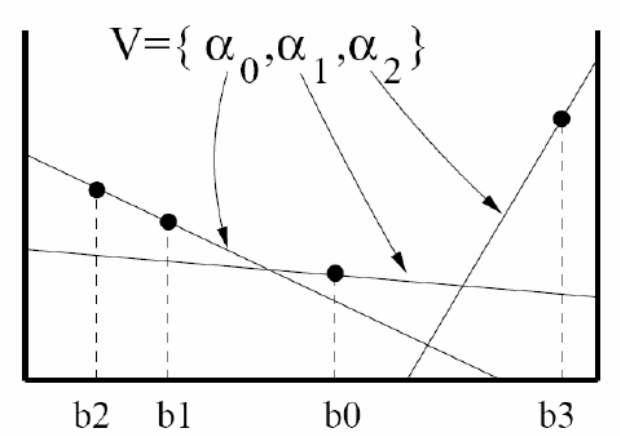
Preliminary Summary on POMDPs

- POMDPs compute the optimal action in partially observable, stochastic domains.
- For finite horizon problems, the resulting value functions are piece-wise linear and convex and can be calculated "easily".
- Based on appropriate assumptions, POMDPs can be applied also to problems of realistic sizes.
- Often simiplifications and approximations are used:
  - $\mathcal{Q}$MDPs
  - AMDPs: Augmented MPDs
  - PBVI: Point-based value iteration
  - Monte Carlo POMDPs

see also: cs.brown.edu/research/ai/pomdp/index.html

Point Based Value Iteration

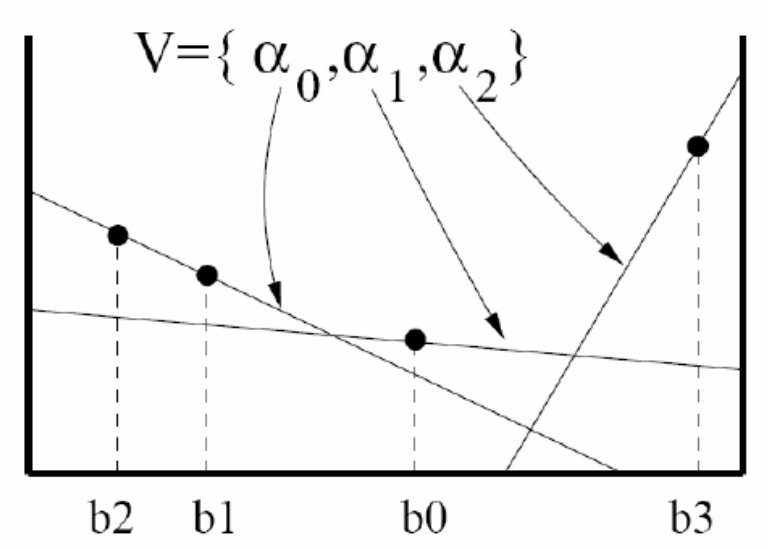
- Maintains a set of example beliefs
- Only considers constraints that maximise value function for at least one of the examples



- Solve POMDP for finite set of belief points
  - Initialise linear segment for each belief point and iterate
- Occasionally add new belief points
  - Add point after a fixed horizon
  - Add points when improvements fall below a threshold
  - Add points implied by belief update if sufficiently different from present set

Point Based Value Iteration

- Solve POMDP for finite set of belief points

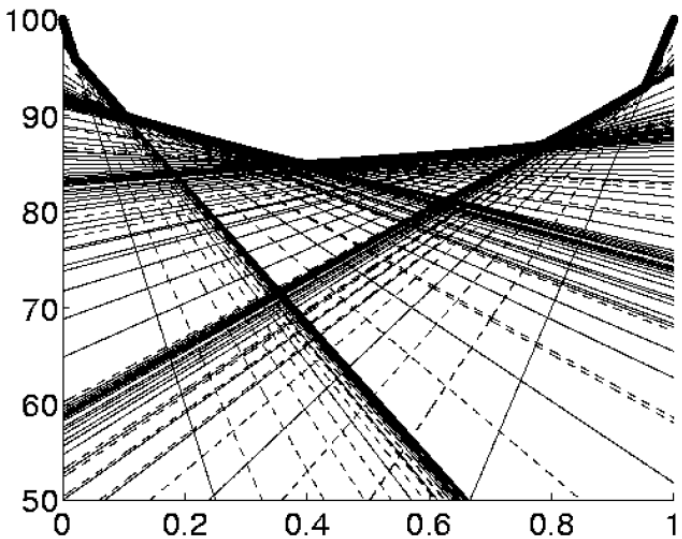


- Can do point updates in polynomial time
  - Modify belief update so that one vector is maintained per point
  - Simplified by finite number of belief points
- Does not require pruning!
  - Only need to check for redundant vectors

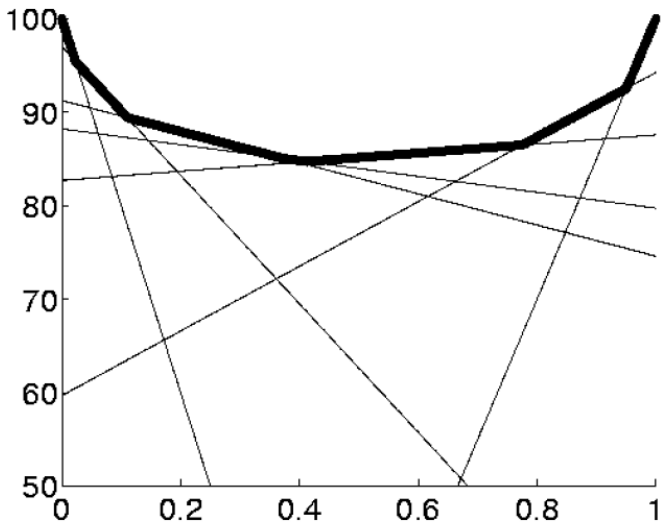
J. Pineau, G. Gordon, and S. Thrun, Point-based value iteration: An anytime algorithm for POMDPs. International joint conference on artificial intelligence. Vol. 18. Lawrence Erlbaum Associates Ltd, 2003.

Point-based Value Iteration

- Value functions for $t = 30$



Exact value function



PBVI

$\mathcal{Q}$MDPs

- $\mathcal{Q}$MDPs only consider state uncertainty in the first step (and, in a sense, similar to $\mathcal{Q}$-learning:)
- After that, the world is assumed to become fully observable.

Algorithm $\mathcal{Q}\text{MDP}(b = (p_1, \ldots, p_N))$

$\hat{V} = \text{MPD\_DiscreteValueIteration}()$

for all control actions u do

$\mathcal{Q}(x_i, u) = r(x, u) + \sum_{j=1}^{N} \hat{V}(x_j)\, p(x_j | u, x_i)$

end for

return $u' = \arg\max_u \sum_{i=1}^{N} p_i \mathcal{Q}(x_i, u)$

Augmented MDPs

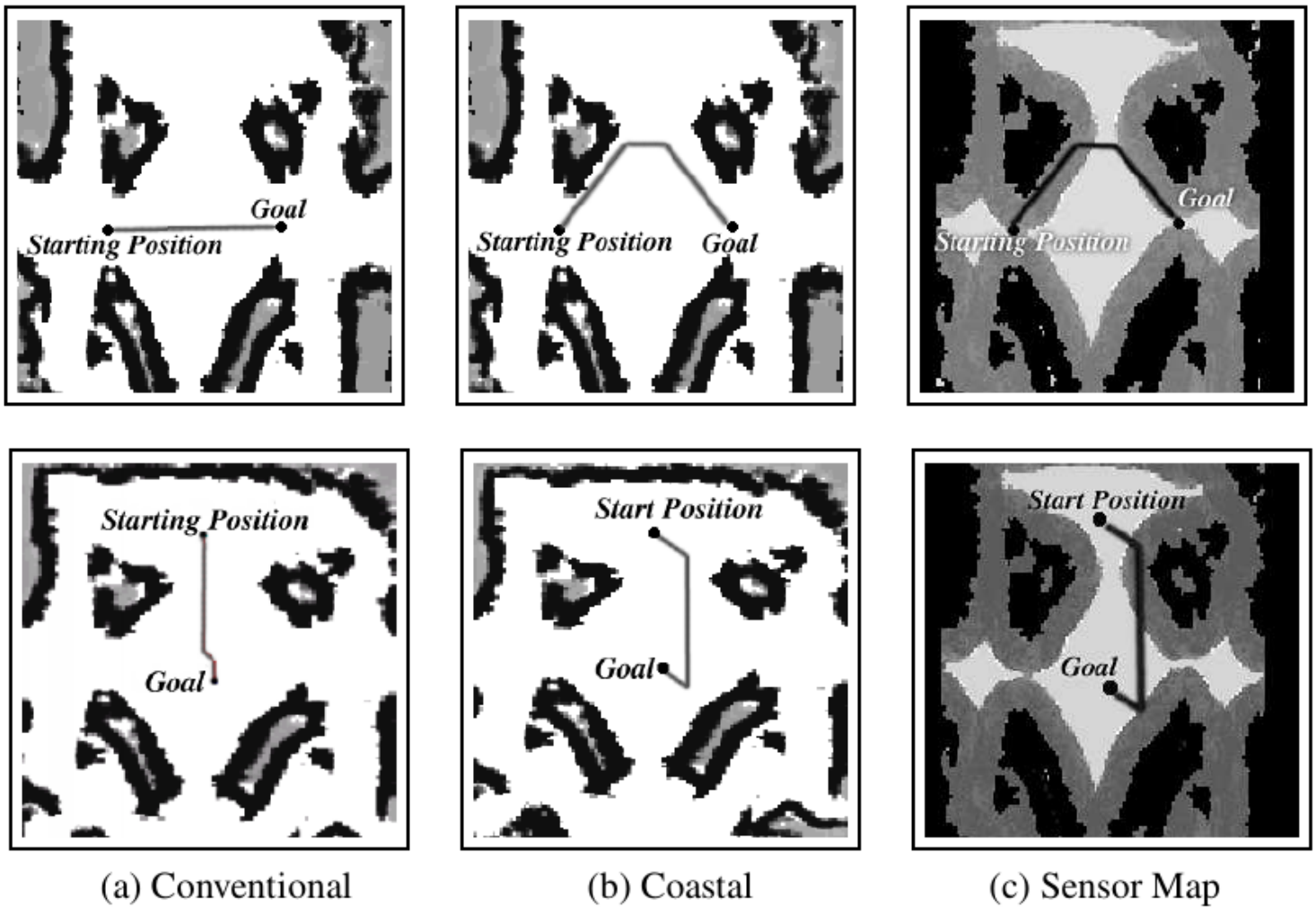
- Augmentation adds uncertainty component to state space, e.g.,

$$\bar{b} = \begin{pmatrix} \arg\max_x b(x) \\ H_b(x) \end{pmatrix} \quad \text{with} \quad H_{b(x)} = -\int b(x) \log b(x)\, dx$$

- Planning is performed by MDP in augmented state space
- Transition, observation and payoff models have to be learnt

N. Roy and S. Thrun, Coastal navigation with mobile robots. In NIPS 12, 1999.

Coastal Navigation by AMDPs (museum environment)



(a) Conventional (b) Coastal (c) Sensor Map

see: Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic robotics. MIT press.

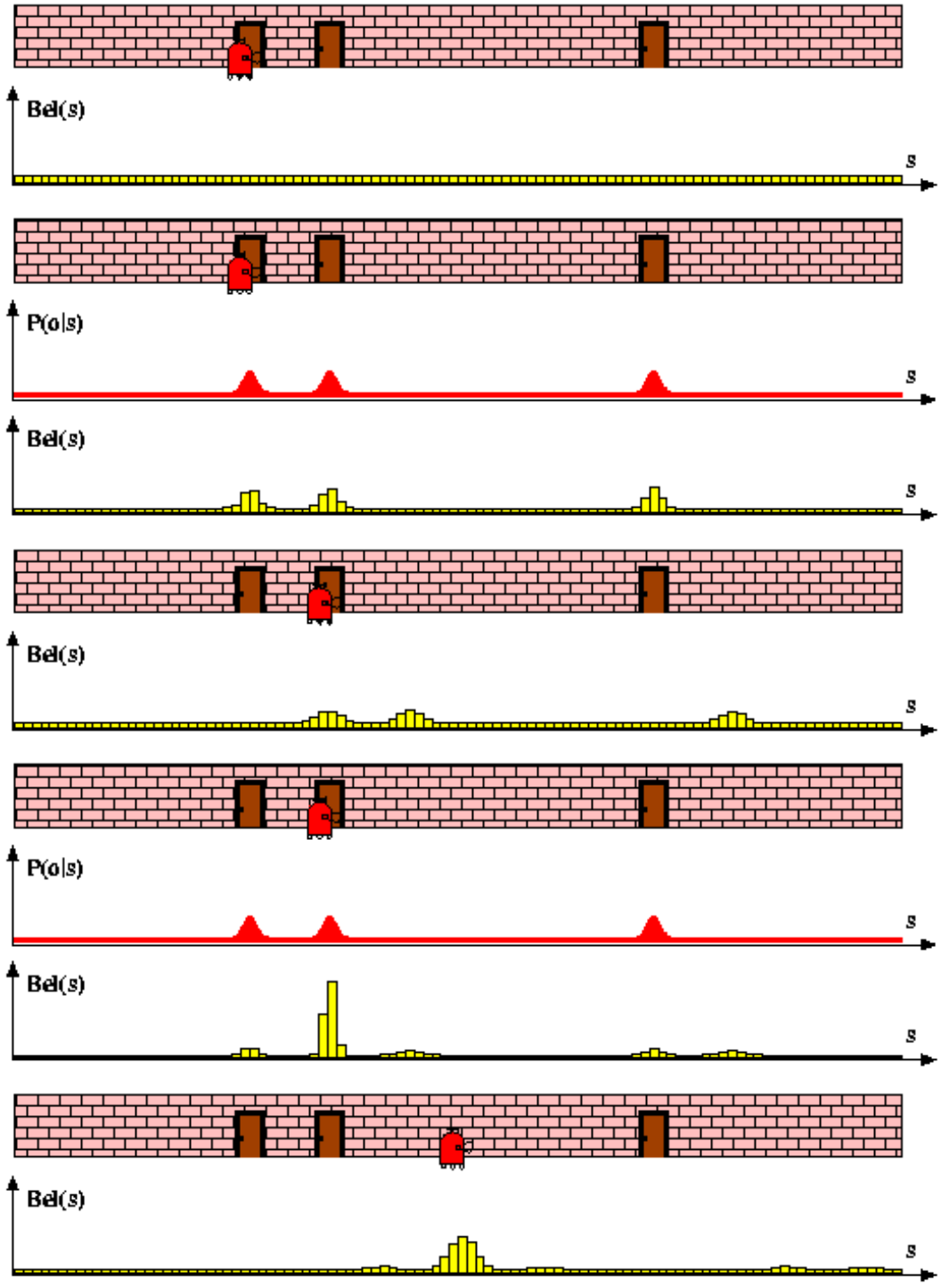
Monte Carlo POMDPs

- Represent beliefs by samples
- Estimate value function on sample sets
- Simulate control and observation transitions between beliefs

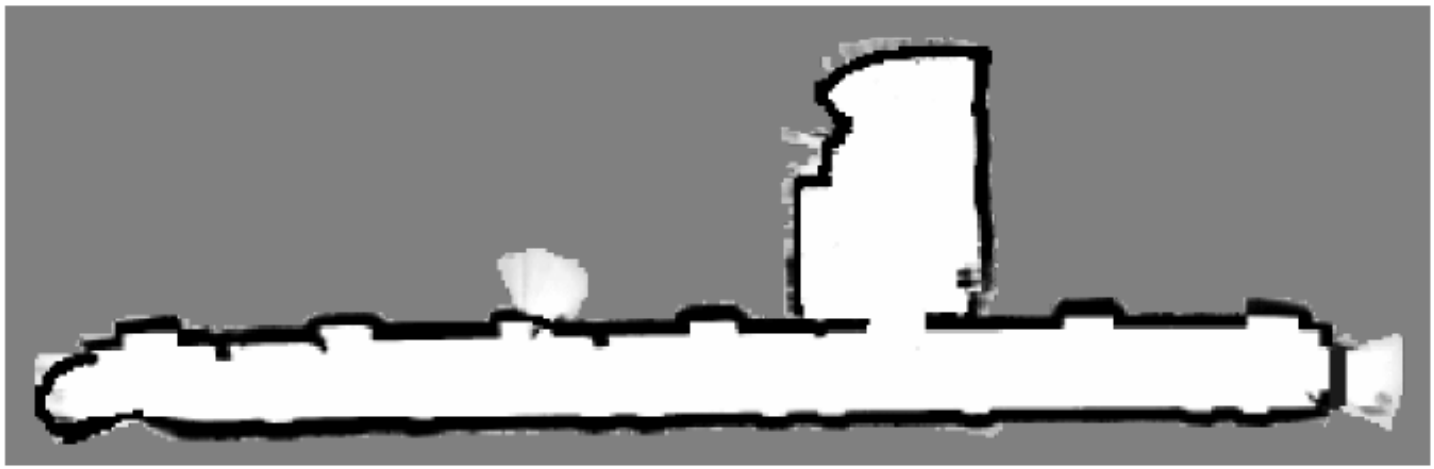
S. Thrun, Monte carlo pomdps. NIPS 12 (2000) 1064-1070.

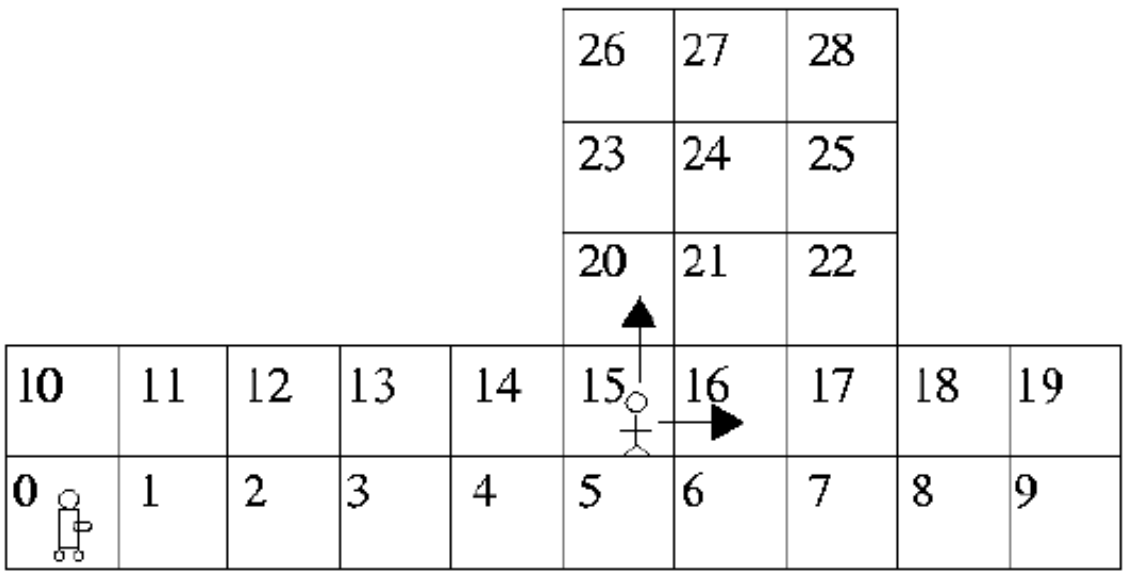
Bayes Filter Implementations in Probabilistic Robotics

(piece-wise constant representation)



PBVI: Example Application ("tag")





Dimensionality Reduction on Beliefs

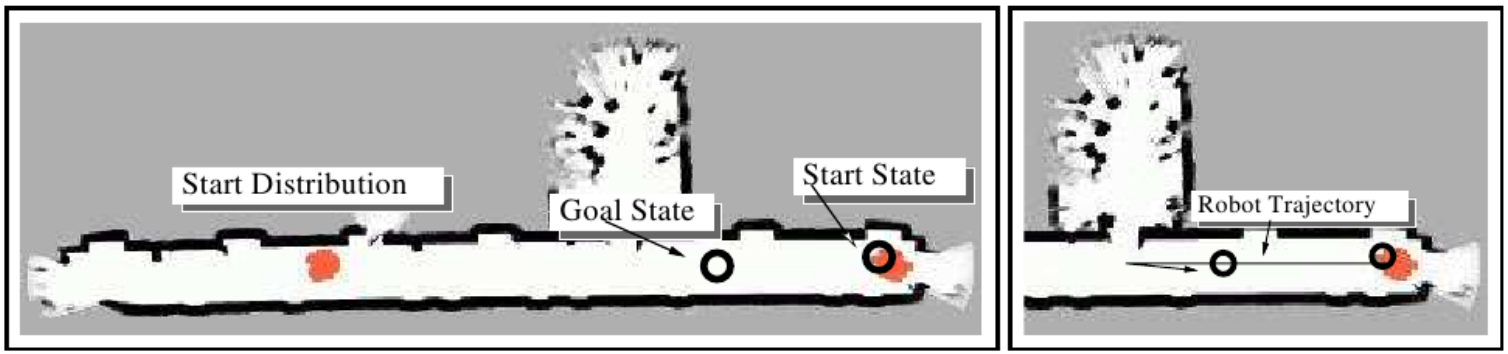


Figure 4: An example robot trajectory, using the policy learned using 5 basis functions. On the left are the start conditions and the goal. On the right is the robot trajectory. Notice that the robot drives past the goal to the lab door to localise itself, before returning to the goal.

N. Roy, and G. Gordon. Exponential family PCA for belief compression in POMDPs. NIPS 15 (2002): 1635-1642.

More particle filters

- Represent belief by random samples
- Estimation of non-Gaussian, nonlinear processes
- Monte Carlo filter, Survival of the fittest, Condensation, Bootstrap filter, Particle filter
- Filtering: [Rubin, 88], [Gordon et al., 93], [Kitagawa 96]
- Computer vision: [Isard and Blake 96, 98]
- Dynamic Bayesian Networks: [Kanazawa et al., 95]

Summary on POMDPs

- POMDPs compute the optimal action in partially observable, stochastic domains.
- For finite horizon problems, the resulting value functions are piece-wise linear and convex, but very complicated
- In each iteration the number of linear constraints grows exponentially
- A number of heuristic and stochastic approaches are available to reduce the complexity.
- (more or less) Heuristic versions of POMDPs have been applied successfully to realistic problem is robotics, media access control in ad-hoc networks, language-based communication systems, medical image-based diagnosis

What is Missing in POMDPs?

- POMDPs do not describe natural metrics in environment
  - When driving, we know both global and local distances
- POMDPs do not natively recognise differences between scales
  - Uncertainty in control is entirely different from uncertainty in routing
- POMDPs conflate properties of the environment with properties of the agent
  - Roads and buildings behave differently from cars and pedestrians: we need to generalise over them differently
- POMDPs are defined in a global coordinate frame, often discrete
  - We may need many different representations in real problems