Partially Observable MDPs

Alain Dutech

Equipe MAIA - LORIA - INRIA
Nancy, France
Web : `http://maia.loria.fr`
Mail : `Alain.Dutech@loria.fr`

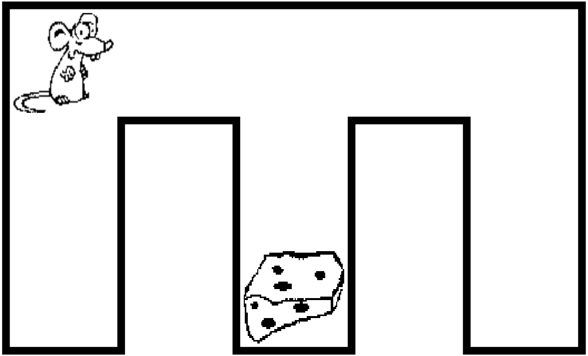
19 février 2010

Outline

Outline

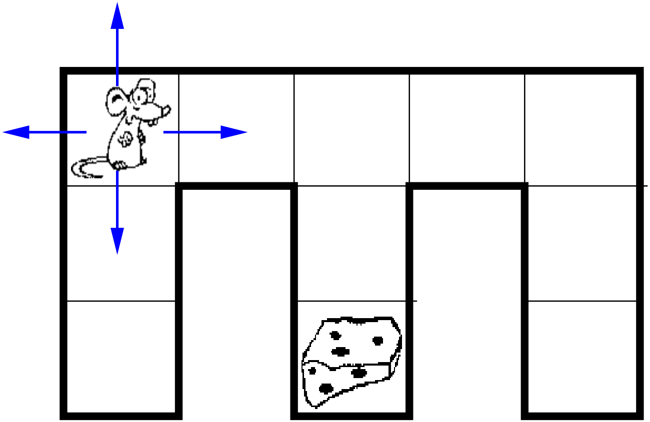
Cheese Maze

Skip



Cheese Maze

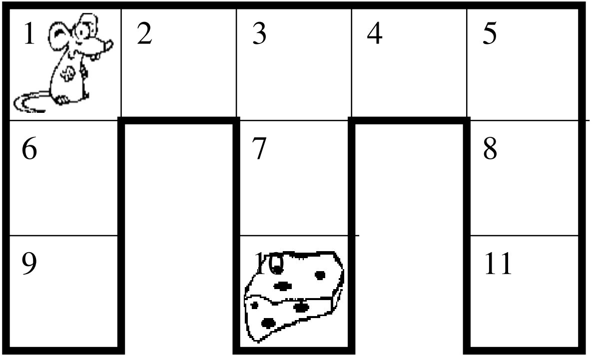
Skip



Cheese Maze

Skip

Observation : in state 1, the mouse only observes upper and left walls.

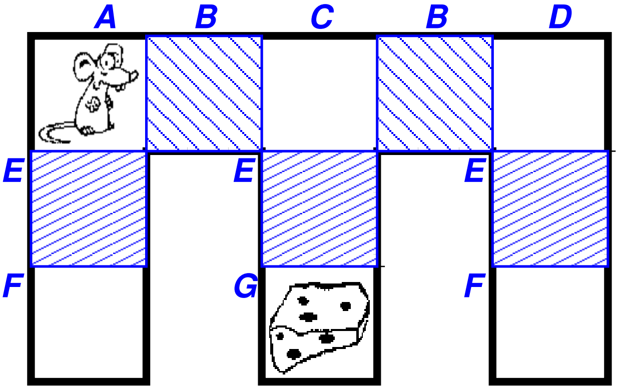


States 1–11

Cheese Maze

Skip

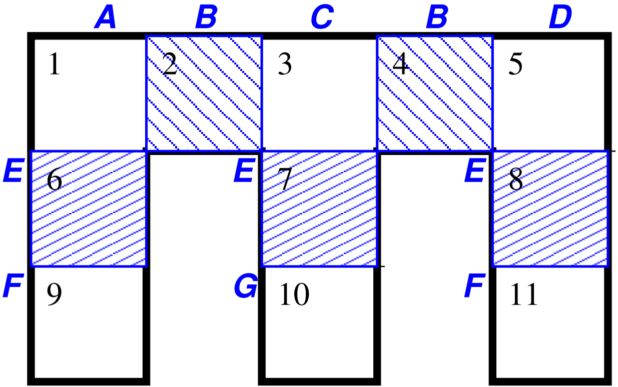
Observation : in state 1, the mouse only observes upper and left walls.



Observations A–G

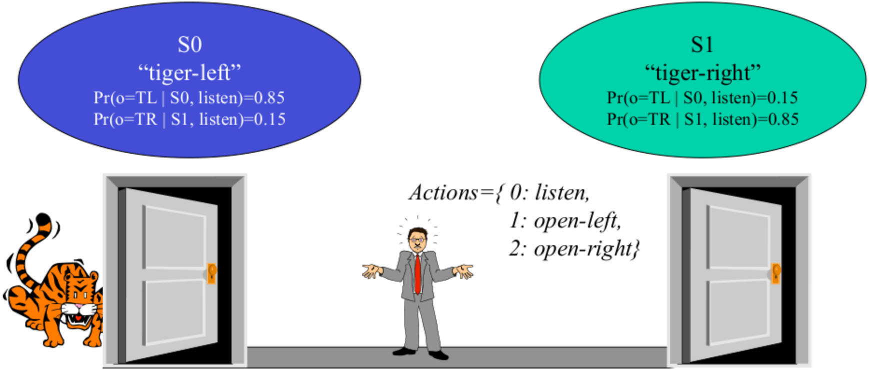
Cheese Maze

Skip



States 1–11 ***Observations A–G***

Tiger problem



Reward Function

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

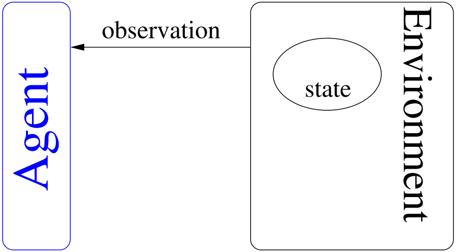
Observations

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

Outline

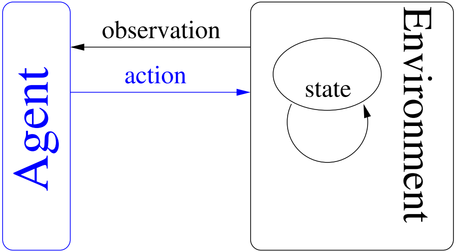
Partially Observable Markov Decision Process

- $\mathcal{S}$: state space
- Ω : observation space
- $\mathcal{A}$: action space
- $T : \mathcal{S} \times \mathcal{A} \longrightarrow \Pi(\mathcal{S})$ transition function
- $O : \mathcal{S}\, [\times \mathcal{A} \times \mathcal{S}] \longrightarrow \Pi(\Omega)$ observation function
- $r : \mathcal{S}\, [\times \mathcal{A} \times \mathcal{S}] \longrightarrow \mathbb{R}$ reward function
- (b_0 : initial state distribution)
  - A POMDP describes a problem, not a solution/behavior/policy



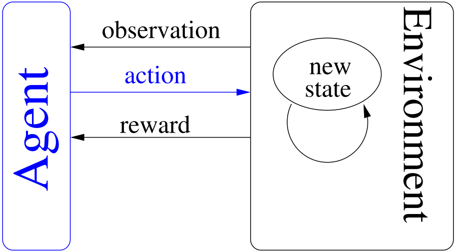
Partially Observable Markov Decision Process

- $\mathcal{S}$: state space
- Ω : observation space
- $\mathcal{A}$: action space
- $T : \mathcal{S} \times \mathcal{A} \longrightarrow \Pi(\mathcal{S})$ transition function
- $O : \mathcal{S} [\times \mathcal{A} \times \mathcal{S}] \longrightarrow \Pi(\Omega)$ observation function
- $r : \mathcal{S} [\times \mathcal{A} \times \mathcal{S}] \longrightarrow \mathbb{R}$ reward function
- (b_0 : initial state distribution)
  - A POMDP describes a problem, not a solution/behavior/policy



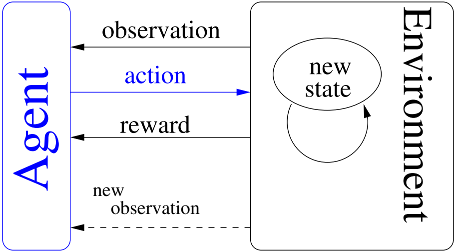
Partially Observable Markov Decision Process

- $\mathcal{S}$: state space
- Ω : observation space
- $\mathcal{A}$: action space
- $T : \mathcal{S} \times \mathcal{A} \longrightarrow \Pi(\mathcal{S})$ transition function
- $O : \mathcal{S} \, [\times \mathcal{A} \times \mathcal{S}] \longrightarrow \Pi(\Omega)$ observation function
- $r : \mathcal{S} \, [\times \mathcal{A} \times \mathcal{S}] \longrightarrow \mathbb{R}$ reward function
- (b_0 : initial state distribution)
  - A POMDP describes a problem, not a solution/behavior/policy

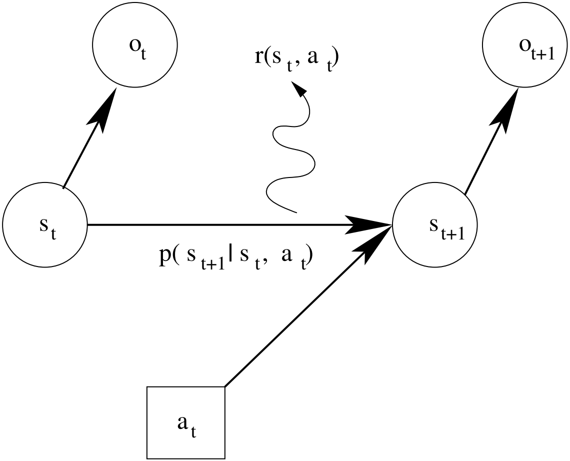


Partially Observable Markov Decision Process

- $\mathcal{S}$: state space
- Ω : observation space
- $\mathcal{A}$: action space
- $T : \mathcal{S} \times \mathcal{A} \longrightarrow \Pi(\mathcal{S})$ transition function
- $O : \mathcal{S}\,[\times \mathcal{A} \times \mathcal{S}] \longrightarrow \Pi(\Omega)$ observation function
- $r : \mathcal{S}\,[\times \mathcal{A} \times \mathcal{S}] \longrightarrow \mathbb{R}$ reward function
- (b_0 : initial state distribution)
  - A POMDP describes a problem, not a solution/behavior/policy



POMDP: a dynamical view



POMDP as an Influence Diagram

Outline

Solving a POMDP

Problem

find an optimal policy, *i.e.* that maximises a function of the reward. (*eg.* cumulative reward).

- non-markovian: existence of a *value function* ?
- information state: the policy is a function of what ?

Solving a POMDP

Problem

find an optimal policy, *i.e.* that maximises a function of the reward. (*eg.* cumulative reward).

- non-markovian: existence of a *value function* ?
- information state: the policy is a function of what ?

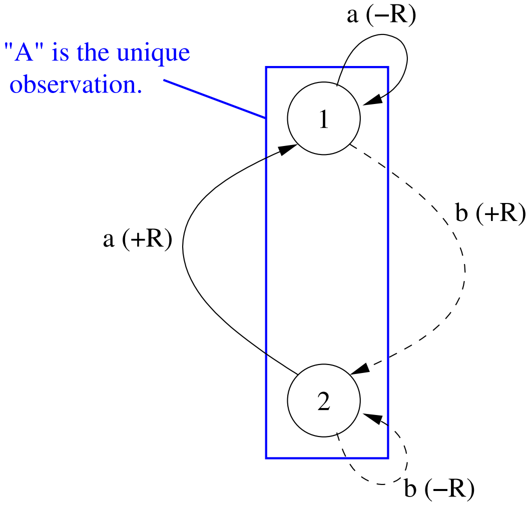
Elements of solution

- classical results of MDP (⇝ **see O. Sigaud)**
- convergence of “naïve” classical MDP algorithms
- *belief state* as valid/useful information state go direct
- Planification: when a model is known
  - Witness algorithm
  - Incremental prunning algorithm
- Learning: when the model is unknown
  - learning useful state extensions
  - learning the model
  - using Predictive State Representation

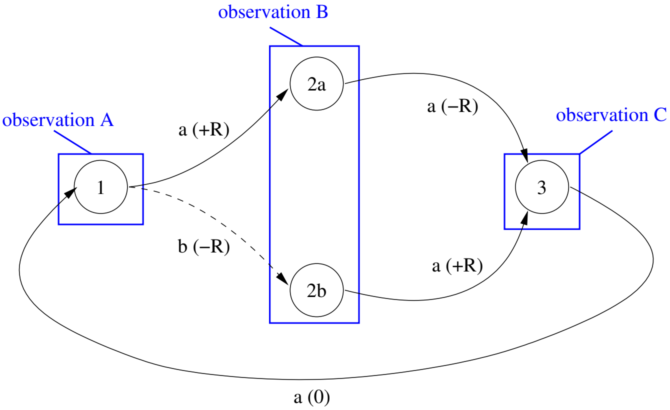
Outline

Stochastic Memoryless Policy

Stochastic memoryless policy can be arbitraty better than a deterministic memoryless policy.
[Singh et al., 1994]



No optimal memoryless policy



No memoryless policy leads to an optimal “adapted” value function. [Singh et al., 1994]

$$\vartheta^{\pi}(o) = \sum_{s \in \mathcal{S}} \Pr^{\pi}(s|o)\mathcal{V}_{\pi'}(s)$$

Convergence of "classical" algorithms [Jaakkola et al., 1994]

- TD(0)

$$\forall o \in \Omega,\ \vartheta^\pi(o) = \sum_{s \in \mathcal{S}} \Pr^\pi(s|o) \left[r(s) + \gamma \sum_{o' \in \Omega} \Pr^\pi(s, o') \vartheta^\pi(o') \right],$$

where $\Pr^\pi(s, o') = \sum_{s' \in \mathcal{S}} \Pr^\pi(s'|s) O(o'|s')$.

- Q-Learning

$$Q(o, a) = \sum_{s \in \mathcal{S}} \Pr^{\pi_{\text{exp}}}(s|o, a) \left[r(s, a) + \gamma \sum_{o' \in \Omega} \Pr^a(s, o') max_{a' \in \mathcal{A}} Q(o', a') \right],$$

where $\Pr^{\pi_{\text{exp}}}(s|o, a)$ is the asymptotic occupation probability and where $\Pr^a(s, o') = \sum_{s' \in \mathcal{S}} T(s'|s, a) O(o'|s')$.

Finding the best memoryless policy [Jaakkola et al., 1994]

- scalar value function: $\sum_{o\in\Omega} \Pr^{\pi}(o) \sum_{s\in\mathcal{S}} \Pr^{\pi}(s|o) V^{\pi}(s)$

- Monte Carlo evaluation of a policy
- Policy improvement
- loop ...

$\leadsto$ local maximum of scalar value function

Outline

Belief States : sufficient statistics of past

Belief States

distribution on states : $b_t(s) = \Pr(s_t = s)$

Belief States : sufficient statistics of past

Belief States

distribution on states : $b_t(s) = \Pr(s_t = s)$

- Sufficient statistics : $b_t(s) = \Pr(s_t | a_t, s_{t-1}, \ldots, s_0)$

⇝ Complete information state.

- Bayesian update :

$$\begin{aligned} b^a_o(s') &= \Pr(s' | b, a, o) \\ &= \frac{O(o|s') \sum_{s \in \mathcal{S}} T(s'|s,a) b(s)}{\sum_{s \in \mathcal{S}} \sum_{s'' \in \mathcal{S}} O(o|s'') T(s''|s,a) b(s)}. \end{aligned}$$

⇝ defines a (continuous) MDP which can be solved [Aström, 1965].

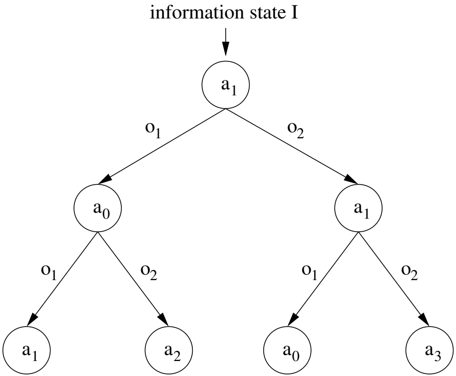
Policy Tree

For *belief states*

- Deterministic optimal policy
- Kind of Conditionnal Plan.

⇝ tree representation

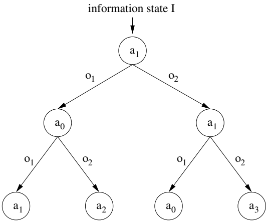
⇝ **see Tiger Example**

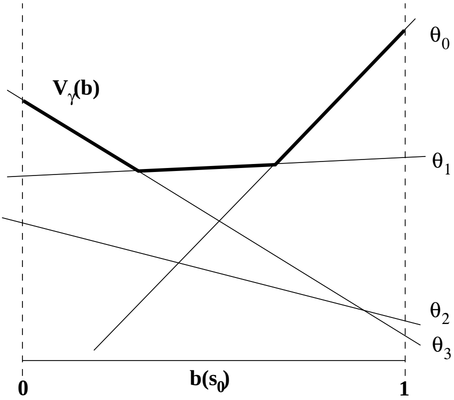


Piece-Wise Linear Convex Value Function

$$V_n^*(b) = \max_{a \in \mathcal{A}} \left[r(b,a) + \gamma \sum_{o \in \Omega} \Pr(o|b,a) V_{n-1}^*(T(b,o,a)) \right].$$

- Finite horizon n
- θ-Vector = one policy...





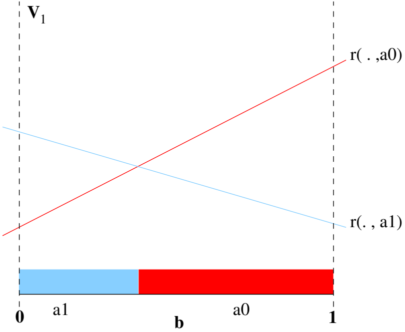
Skip details

Outline

Dynamic programming on PWLC value function (0)

$$V_{n-1}(b) = \max_{a \in \mathcal{A}} [r(b,a)]$$
$$= \max_{a \in \mathcal{A}} \left[\sum_s r(s,a).b(s) \right]$$

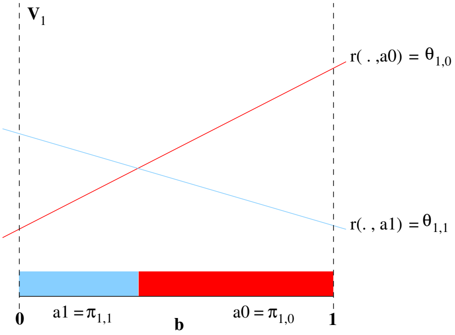
a θ is mapped to a policy of one action.



Dynamic programming on PWLC value function (0)

$$V_{n-1}(b) = \max_{a \in \mathcal{A}} [r(b,a)]$$
$$= \max_{a \in \mathcal{A}} \left[\sum_s r(s,a).b(s) \right]$$

a θ is mapped to a policy of one action.

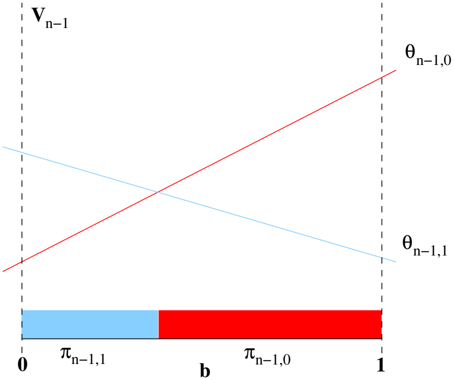


Dynamic programming on PWLC value function (1)

PWLC V_{n-1} at step $n-1$

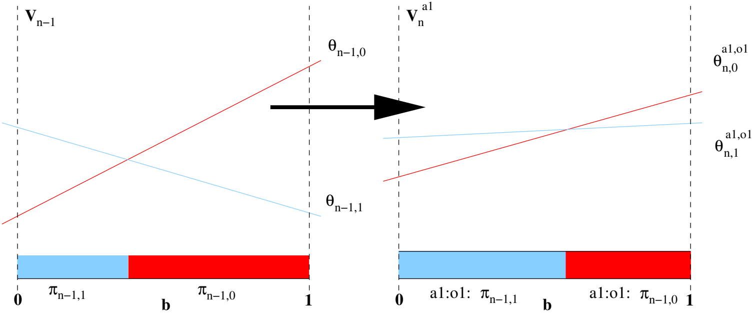
$$V_{n-1}(b) = \max_{\theta \in \Theta_{n-1}} b.\theta$$

a θ is mapped to a policy



Dynamic programming on PWLC value function (2)

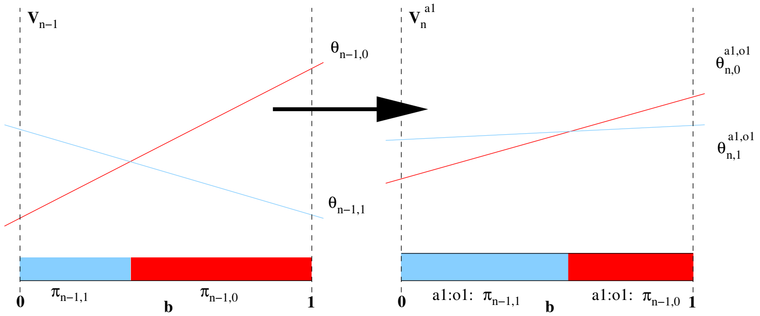
PWLC V_n at step n for first action $a1$ and observation $o1$



$$V_n^*(b) = \max_{a \in \mathcal{A}} \left[r(b,a) + \gamma \sum_{o \in \Omega} \Pr(o|b,a) V_{n-1}^*(T(b,o,a)) \right].$$

Dynamic programming on PWLC value function (2)

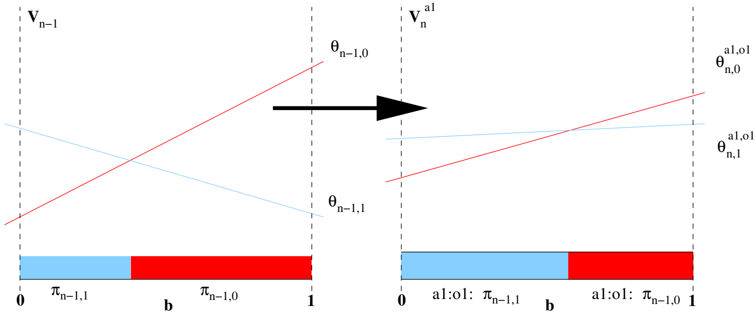
PWLC V_n at step n for first action $a1$ and observation $o1$



$$V_n^*(b) = \max_{a\in\mathcal{A}} \left[r(b,a) + \gamma \sum_{o\in\Omega} \Pr(o|b,a) V_{n-1}^*(T(b,o,a)) \right].$$

Dynamic programming on PWLC value function (2)

PWLC V_n at step n for first action $a1$ and observation $o1$



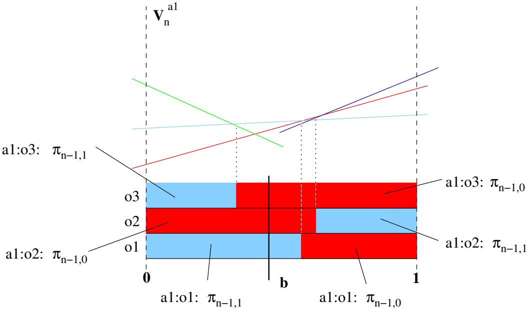
$$\theta_n^{a1,o1}(b,s) = \frac{r(s,a1)}{|\Omega|} + \gamma \sum_{s' \in \mathcal{S}} T(s,a1,s')O(s',o1)\theta_{n-1}^{a1,o1}(b^{a1,o1},s).$$

Dynamic programming on PWLC value function (3)

PWLC V_n at step n for first action $a1$

$$\theta_n^{a1}(b) = \sum_{o \in \Omega} \theta_n^{a1,o}(b).$$

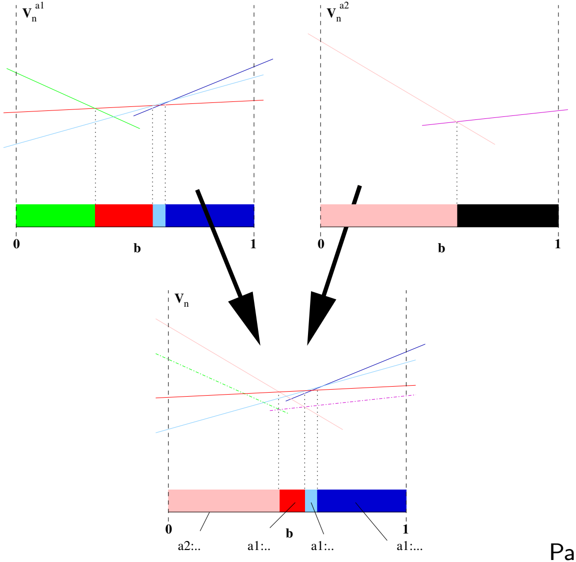
At most $|\Theta_{n-1}|^{|\Omega|}$ θ-vectors



Dynamic programming on PWLC value function (4)

PWLC V_n at step n

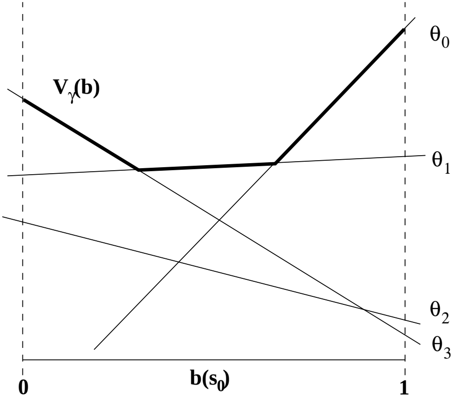
$$\theta_n(b) = \max_{a \in \mathcal{A}} \theta_n^a(b)$$



back to PI

PWLC Value Function with Belief States

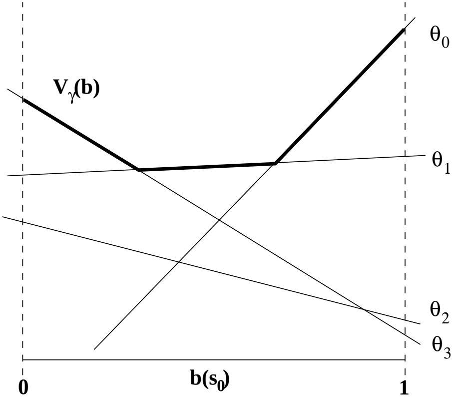
- Finite Horizon POMDP
  - Optimal value function is PWLC [Smallwood and Sondik, 1973]
  - $$V_n(b) = \max_{\theta \in \Theta_n} b.\theta$$
- Infinite Horizon POMDP
  - ϵ-optimal value function is PWLC
  - Optimal only for *transient* POMDP [Sondik, 1971]



$\rightsquigarrow$ the real problem is the size of θ-vector space.

PWLC Value Function with Belief States

- Finite Horizon POMDP
  - Optimal value function is PWLC [Smallwood and Sondik, 1973]
  - $$V_n(b) = \max_{\theta \in \Theta_n} b.\theta$$
- Infinite Horizon POMDP
  - ϵ-optimal value function is PWLC
  - Optimal only for *transient* POMDP [Sondik, 1971]

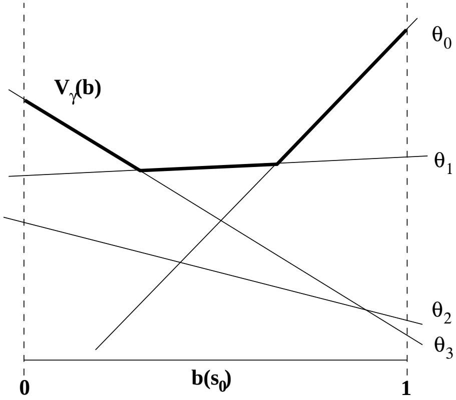


$\rightsquigarrow$ the real problem is the size of θ-vector space.

Parcimonious representation

θ from Θ dominated : $b.\theta \leq \max_{\theta' \in \Theta} b.\theta'$.

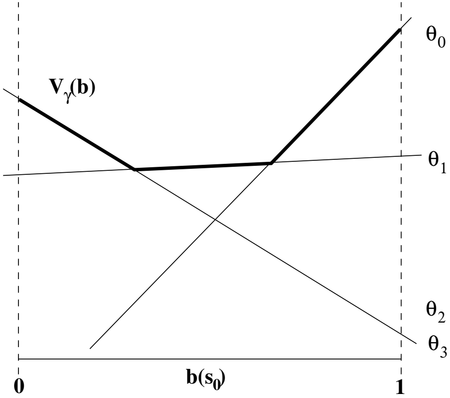
- Exists a minimal representation [Littman and Szepesvári, 1996]



Parcimonious representation

θ from Θ dominated : $b.\theta \leq \max_{\theta' \in \Theta} b.\theta'$.

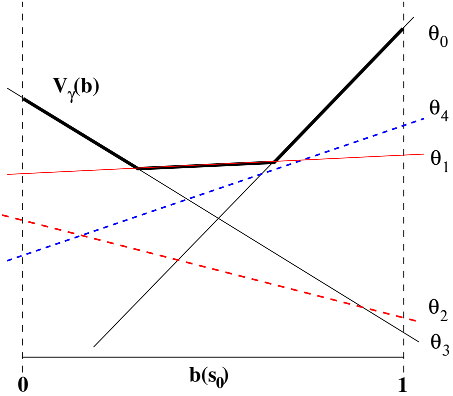
- Exists a minimal representation [Littman and Szepesvári, 1996]



Parcimonious representation

θ from Θ dominated : $b.\theta \leq \max_{\theta' \in \Theta} b.\theta'$.

- Exists a minimal representation [Littman and Szepesvári, 1996]
- θ_2 : entirely dominated
- θ_4 : needs PRUNING



Outline

Algorithm WITNESS: concepts [Cassandra et al., 1994]

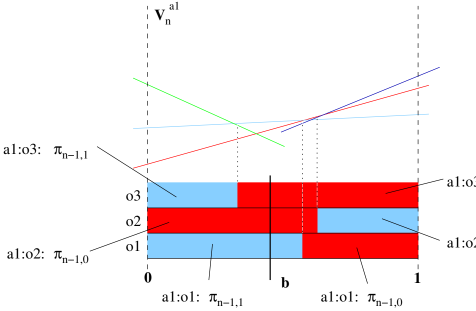
- Incremental build of parcimonious representation.

1. Start from belief state b
2. Look for the best θ-vector of its region
3. Add all "neighbors" to the agenda Υ
   - Either remove from it (dominated θ-vector)
   - Or add best θ-vector from region to V and its neihbors to Υ.
4. Loop

Algorithm $\textsc{Witness}(\Theta_{n-1}, a)$

Input: A parsimonious representation Θ_{n-1} of V^*_{n-1}, an action a
Output: A parsimonious representation $V_n^{*,a}$

$b \leftarrow$ a belief state of $\mathcal{B}$
$\hat{\Theta} \leftarrow \{\theta_n^a(b)\}$
$\Upsilon \leftarrow \mathcal{N}(\theta_n^a(b))$
while $\Upsilon \neq \emptyset$ **do**
  $v \leftarrow$ `RemoveElement`(Υ)
  if $v \in \hat{\Theta}$ **then**
    $b \leftarrow$ **null**
  else
    $b \leftarrow$ `FindVectInRegion`$(v, \hat{\Theta})$
  end
  if $b \neq$ **null then**
    $\hat{\Theta} \leftarrow \hat{\Theta} \cup \{\theta_n^a(b)\}$
    $\Upsilon \leftarrow \Upsilon \cup \{v\}$
    $\Upsilon \leftarrow \Upsilon \cup \mathcal{N}(\theta_n^a(b))$
  end
end
$\Theta_n^a \leftarrow \hat{\Theta}$
return Θ_n^a

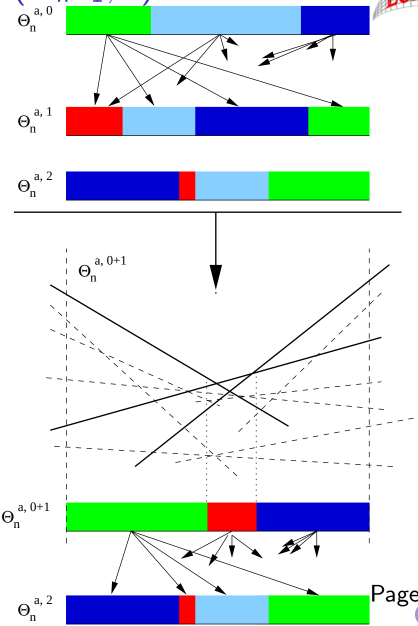


IncrementalPruning: concept [Zang and Lio, 1996]

- Lots of small pruning vs global final pruning.

1. With the set Ψ of all sets of $\Theta_n^{a,o}$ θ-vectors.
2. Take two sets from it and prune them
3. Add new prunned set to Ψ, loop.

Algo IncrementalPruning(Θ_{n-1}, a)



Input: A parsimonious representation Θ_{n-1} of V^*_{n-1}, an action a

Output: A parsimonious representation $V_n^{*,a}$

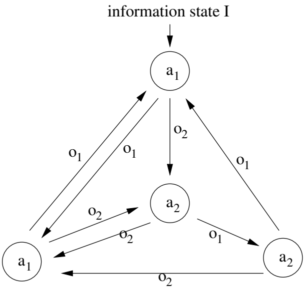
$\Psi \leftarrow \bigcup_o \{\Theta_n^{a,o}\}$
while $|\Psi| > 1$ **do**
  $A \leftarrow \mathtt{RemoveElement}(\Psi)$
  $B \leftarrow \mathtt{RemoveElement}(\Psi)$
  $D \leftarrow \textsc{Prune}(A \oplus B)$
  $\Psi \leftarrow \Psi \cup \{D\}$
end
return Ψ

Outline

Policy Iteration: concepts [Hansen, 1998]

- Grow an ϵ-optimal FSA controler

1. From a given FSA δ compute all new θ-vectors
2. For each new θ-vector
   2.1 If exists in δ, added to $\hat{\delta}$
   2.2 Else modify same but dominated node *i* to $\hat{\delta}$
   2.3 Else add new node to $\hat{\delta}$
3. Loop



- A FSA policy has PWLC Value Function
- One node = One θ-vector

Algorithm Policy Iteration(δ, ϵ)

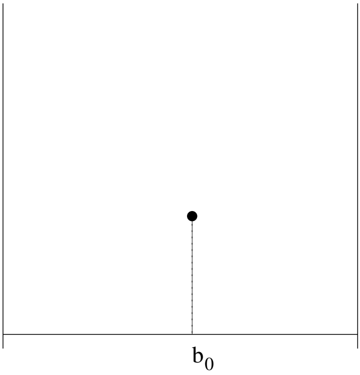
Input: A finite state controller δ and a positive real ϵ
Output: A finite state controller δ^* which is ϵ-optimal

repeat
  Compute V^δ from δ by solving equations (**??**)
  Build $\hat{V}^\delta \leftarrow \texttt{DynamicProgOperator}(V^\delta)$ details on DP
  $\hat{\delta} \leftarrow \emptyset$
  foreach $\hat{\theta}^j \in \hat{V}^\delta$ **do**
    if *there exists a node i of δ associated with $\hat{\theta}^j$ with identical action and links* **then**
      add i to $\hat{\delta}$
    else if *there exists a node i such that $\hat{\theta}^j$ dominates θ^i* **then**
      add i to $\hat{\delta}$, with the action and the links of $\hat{\theta}^j$
    else
      add a *new* node to $\hat{\delta}$ with the actions and links of $\hat{\theta}^j$
  end
  Add to $\hat{\delta}$ all the other nodes of δ that are reachable from $\hat{\delta}$
  $\delta \leftarrow \hat{\delta}$
until $\|\hat{V}^\delta - V^\delta\| \leq \epsilon(1-\gamma)/\gamma$
return $\hat{\delta}$

Outline

Point Based Value Iteration [Pineau et al., 2003]

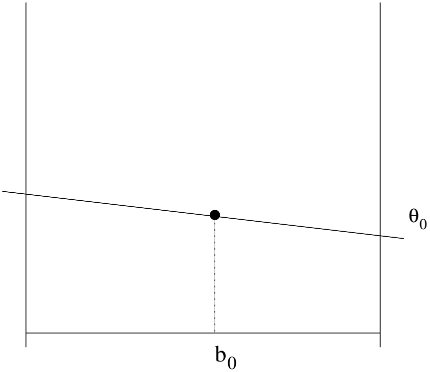
- Start: set of belief states



See also [Spaan and Vlassis, 2005], [Seuken and Zilberstein, 2007]...

Point Based Value Iteration [Pineau et al., 2003]

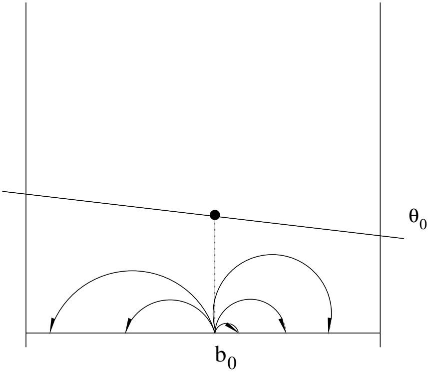
- Start: set of belief states
- Alternate : update / expand



See also [Spaan and Vlassis, 2005], [Seuken and Zilberstein, 2007]...

Point Based Value Iteration [Pineau et al., 2003]

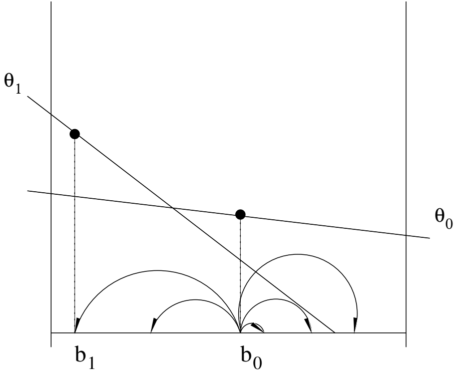
- Start: set of belief states
- Alternate : update / expand



See also [Spaan and Vlassis, 2005], [Seuken and Zilberstein, 2007]...

Point Based Value Iteration [Pineau et al., 2003]

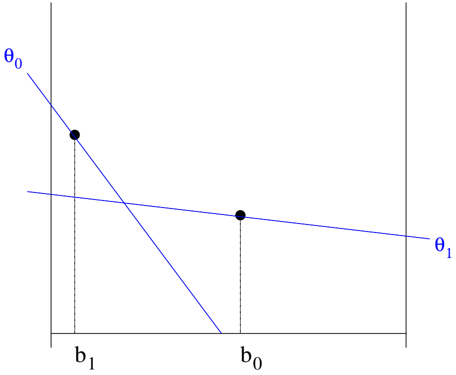
- Start: set of belief states
- Alternate : update / expand



See also [Spaan and Vlassis, 2005], [Seuken and Zilberstein, 2007]...

Point Based Value Iteration [Pineau et al., 2003]

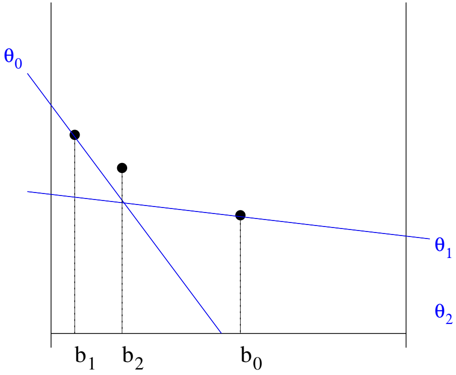
- Start: set of belief states
- Alternate : update / expand



See also [Spaan and Vlassis, 2005], [Seuken and Zilberstein, 2007]...

Point Based Value Iteration [Pineau et al., 2003]

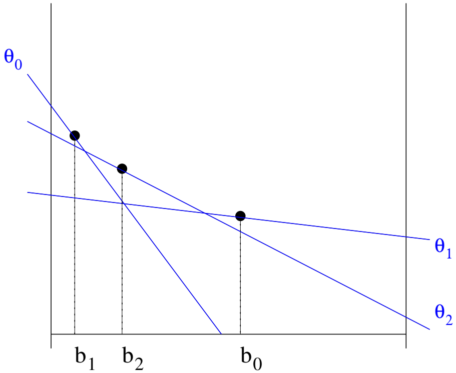
- Start: set of belief states
- Alternate : update / expand



See also [Spaan and Vlassis, 2005], [Seuken and Zilberstein, 2007]...

Point Based Value Iteration [Pineau et al., 2003]

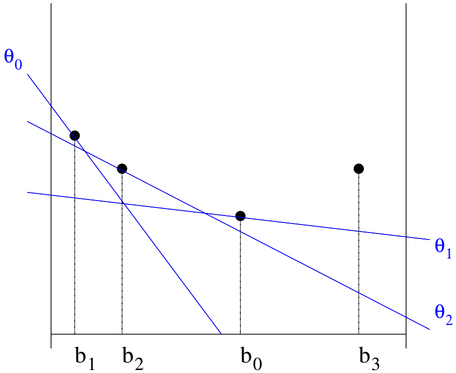
- Start: set of belief states
- Alternate : update / expand



See also [Spaan and Vlassis, 2005], [Seuken and Zilberstein, 2007]...

Point Based Value Iteration [Pineau et al., 2003]

- Start: set of belief states
- Alternate : update / expand
- Only approximation



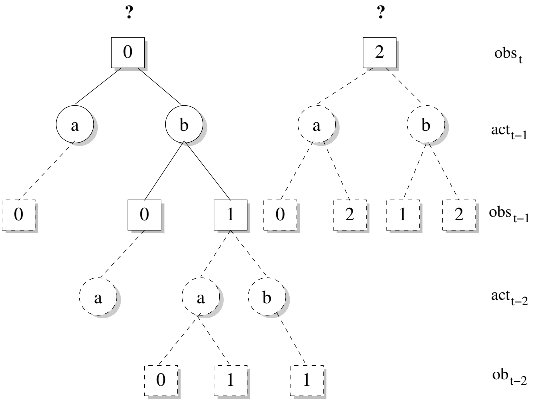
See also [Spaan and Vlassis, 2005], [Seuken and Zilberstein, 2007]...

Outline

Learn State Extensions

POMPD $\equiv$ variable n-Markov Decision Process

- ext. states = (o, a) histories
- Start with 'obs' as "histories"
- Extend *ambiguous* states
- heuristics (Q variations)
- statistical diff. in probability distributions.
- See [McCallum, 1995], [Dutech, 2000]



GPOMDP algorithm [Baxter and Bartlett, 2000]

- randomized policy: $\{\mu(\theta, .)\}_{\theta \in \mathbb{R}^k}$
- Gradient estimate

$$z_{t+1} = \gamma z_t + \frac{\nabla \mu_{a_t}(\theta, o_t)}{\mu_{a_t}(\theta, o_t)}$$

$$\Delta_{t+1} = \Delta_t + \frac{1}{t+1}[r_{t+1} z_{t+1} - \Delta_t]$$

- Interlaced with policy improvement with gradient ascent.

$$\theta_{t+1} = \theta_t + \alpha \Delta_{t+1}$$

$\leadsto$ local optimum

Outline

Predictive State Representation

- a test: $t_i = o_1 a_1 o_2 \ldots o_n$
- prediction. history h: $\Pr(o_1, \ldots, o_n | h, a_1, \ldots, a_{n-1})$
- set of tests: $\mathcal{Q} = \{t_i\}_{i=1,\ldots,q}$

Predictive State Representation

$(1 \times q)$ prediction vector $p(h) = \{\Pr(t_1|h), \Pr(t_2|h), \ldots, \Pr(t_q|h)\}$ iff $\forall h,\ \Pr(t|h) = f_t(p(h))$

Predictive State Representation

- a test: $t_i = o_1 a_1 o_2 \dots o_n$
- prediction. history h: $\Pr(o_1, \dots, o_n | h, a_1, \dots, a_{n-1})$
- set of tests: $\mathcal{Q} = \{t_i\}_{i=1,\dots,q}$

Predictive State Representation

$(1 \times q)$ prediction vector $p(h) = \{\Pr(t_1|h), \Pr(t_2|h), \dots, \Pr(t_q|h)\}$ iff $\forall h,\ \Pr(t|h) = f_t(p(h))$

- linear PSR : $\Pr(t|h) = p(h) m_t^T$
- Update : $p_i(hao) = \Pr(t_i|hao) = \frac{\Pr(aot_i|h}{\Pr ao|h} = \frac{p(h) m_{aot_i}^T}{p(h) m_{ao}^T}$

Theorem

For any environment that can be represented by a finite POMDP model, there exists a linear PSR with number of tests no larger than the number of states in the minimal POMDP model.

▸ matrix dyn. sys. ▸ skip details

Planning with PSRs

Good news

- Value function for PSRs is PWLC. [James et al., 2004]

But

- Exact algorithms exists but not parcimonious
  $\rightsquigarrow$ DP operator does not necessarily give *valid* PSRs.

- Approximate algorithms with good results (PBVI-PSR [Izadi and Precup, 2008])

Learning PSRs [Singh et al., 2003]

- How to maintain correct predictions for the tests
  $\rightsquigarrow$ m_{aot_i} and m_{ao}

- Gradient of the error
- $E(t) = \sum_{x \in X_t} [p(x|h_{t-1}) - \hat{(}p)(x|h_{t-1})]^2$
  where X_t is the set of all extension tests possible from time t

- indirect solution, local optimum, huge iterations

Discovering PSRs [James and Singh, 2004]

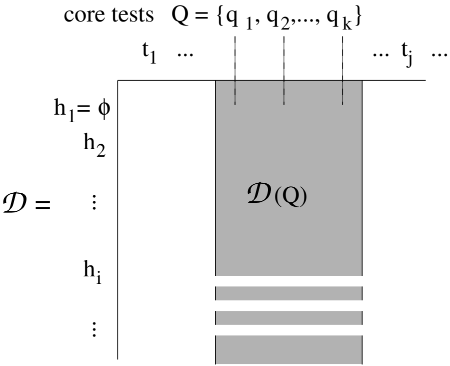
- for histories and tests of size 1
- build the empirical system-dynamics matrix $\mathcal{D}$
- look for independant columns $\rightsquigarrow$ core-tests $\mathcal{Q}_{\mathcal{T}_1}$
- look for independant rows $\rightsquigarrow$ core-histories $\mathcal{Q}_{\mathcal{H}_1}$
- build new $\mathcal{D} = (\mathcal{Q}_{\mathcal{T}_1} \bigcup \mathcal{Q}_{\mathcal{T}_1}^{+ao}) \bigotimes (\mathcal{Q}_{\mathcal{H}_1} \bigcup \mathcal{Q}_{\mathcal{H}_1}^{+ao})$
- loop

- (uses rank estimation of unknown matrix, need reset action, can learn PSR in parallel)

System-dynamics Matrix [Singh et al., 2004]

PSRs : set of k columns for syst-dyn of linear dimension k.

- n-MDP $\rightsquigarrow (|\mathcal{A}||\Omega|)^n$
- POMDP,HMM $\rightsquigarrow$ $< |\mathcal{S}|$
- POMDP $\subset$ PSR



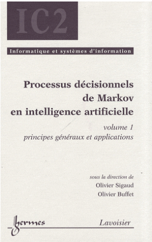
Conclusion

- What was here
  - formalization of POMDPs
  - memoryless policies
  - belief states and PWLC value function
  - value iteration: WITNESS, INCREMENTAL PRUNING
  - policy iteration
  - others: state extension, GPOMDP, PSR

Conclusion

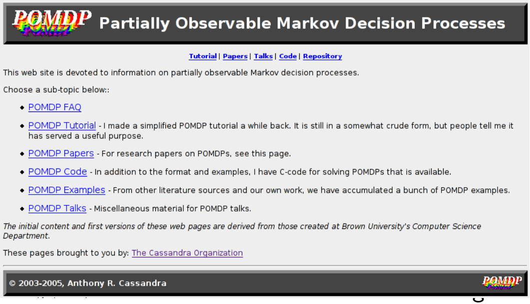
- What was here
  - formalization of POMDPs
  - memoryless policies
  - belief states and PWLC value function
  - value iteration: WITNESS, INCREMENTAL PRUNING
  - policy iteration
  - others: state extension, GPOMDP, PSR
- What was left
  - complexity results (from *bad* to *worst*)
  - applications (robotics, H/C dialog, H/R interactions, ??)
  - cognitive aspects (how *good* representations are build)

Some starting references



Groupe PDMIA (2008).
Processus Décisionnels de Markov en Intelligence Artificielle. (Edité par Olivier Buffet et Olivier Sigaud), volume 1 & 2.
Lavoisier - Hermes Science Publications.
(a translation is about to be puvlished)

http://www.pomdp.org/


POMDP Partially Observable Markov Decision Processes

Tutorial | Papers | Talks | Code | Repository

This web site is devoted to information on partially observable Markov decision processes.

Choose a sub-topic below::

- POMDP FAQ
- POMDP Tutorial - I made a simplified POMDP tutorial a while back. It is still in a somewhat crude form, but people tell me it has served a useful purpose.
- POMDP Papers - For research papers on POMDPs, see this page.
- POMDP Code - In addition to the format and examples, I have C-code for solving POMDPs that is available.
- POMDP Examples - From other literature sources and our own work, we have accumulated a bunch of POMDP examples.
- POMDP Talks - Miscellaneous material for POMDP talks.

The initial content and first versions of these web pages are derived from those created at Brown University's Computer Science Department.

These pages brought to you by: The Cassandra Organization

© 2003-2005, Anthony R. Cassandra POMDP

Bibliography I

Aström, K. (1965).
Optimal control of Markov decision processes with incomplete state estimation.
Journal of Mathematical Analysis and Applications, 10:174–205.

Baxter, J. and Bartlett, P. (2000).
Reinforcement learning in POMDP's via direct gradient ascent.
In *Proc. 17th International Conf. on Machine Learning (ICML'00)*.

Cassandra, A., Kaelbling, L., and Littman, M. (1994).
Acting optimally in partially observable stochastic domains.
In *Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI)*.

Bibliography II

Dutech, A. (2000).
Solving POMDP using selected past-events.
In *Proceedings of the 14th European Conference on Artificial Intelligence, ECAI2000.*

Groupe PDMIA (2008).
Processus Décisionnels de Markov en Intelligence Artificielle. (Edité par Olivier Buffet et Olivier Sigaud), volume 1 & 2.
Lavoisier - Hermes Science Publications.

Hansen, E. (1998).
Solving POMDPs by searching in policy space.
In *Proc. of the Fourteenth Conf. on Uncertainty in Artificial Intelligence (UAI'98.*

Bibliography III

Izadi, M. T. and Precup, D. (2008).
Point-based planning for predictive state representations.
In *Proc. of Canadian AI'08*, volume 5032 of *Lectures Notes in Computer Sciences*, pages 126–137.

Jaakkola, T., Singh, S., and Jordan, M. (1994).
Reinforcement learning algorithm for partially observable markov decision problems.
In Tesauro, G., Touretsky, D., and Leen, T., editors, *Advances in neural information processing systems*, volume 7. MIT Press, Cambridge, Massachusetts.

Bibliography IV

James, M. and Singh, S. (2004).
Learning and discovery of predictive state representations in dynamical systems with reset.
In *Proc. of the Twenty-first Int. Conf. of Machine Learning (ICML'04)*.

James, M. R., Singh, S., and Littman, M. (2004).
Planning with predictive state representations.
In *Proc. of the Int. Conf. on Machine Learning and Applications (ICMLA'04)*.

Littman, M. and Szepesvári, C. (1996).
A generalized reinforcement-learning model: Convergence and applications.
In *Proc. of the Thirteenth Int. Conf. on Machine Learning (ICML'96)*.

Bibliography V

McCallum, A. (1995).
Reinforcement learning with selective perception and hidden state.
PhD thesis, Dept. of Computer Science, University of Rochester, Rochester, New York.

Pineau, J., Gordon, G., and Thrun, S. (2003).
Point-based value iteration: An anytime algorithm for POMDPs.
In *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI'03)*, pages 1025 – 1032.

Puterman, M. (1994).
Markov Decision Processes: discrete stochastic dynamic programming.
John Wiley & Sons, Inc. New York, NY.

Bibliography VI

Seuken, S. and Zilberstein, S. (2007).
Memory-bounded dynamic programming for DEC-POMDPs.
In *Proc. of the Twentieth Int. Joint Conf. on Artificial Intelligence (IJCAI'07)*.

Singh, S., Jaakkola, T., and Jordan, M. (1994).
Learning without state estimation in partially observable markovian decision processes.
In *Proceedings of the Eleventh International Conference on Machine Learning*.

Singh, S., James, M. R., and Rudary, M. R. (2004).
Predictive state representations: A new theory for modeling dynamical systems.
In *Proc. of the twentieth Conf. on Uncertainty in Artificial Intelligence (UAI'04)*.

Bibliography VII

Singh, S., Littman, M., Jong, N., Pardoe, D., and Stone, P. (2003).
Learning predictive state representations.
In *Proc. of the Twentieth Int. Conf. of Machine Learning (ICML'03)*.

Smallwood, R. D. and Sondik, E. J. (1973).
The optimal control of partially observable Markov processes over a finite horizon.
Operations Research, 21:1071–1088.

Sondik, E. (1971).
The optimal control of partially observable markov decision processes.
PhD thesis, Stanford University, California.

Bibliography VIII

Spaan, M. and Vlassis, N. (2005).
Perseus: Randomized point-based value iteration for POMDPs.
Journal of Artificial Intelligence Research (JAIR), 24:195–220.

Sutton, R. and Barto, A. (1998).
Reinforcement Learning.
Bradford Book, MIT Press, Cambridge, MA.

Zang, N. and Lio, W. (1996).
Planning in stochastic domains: Problem characteristics and approximation.
Technical report, Tech. report HKUST-CS96-31, Honk-Kong University of Science and Technology.

Neighbor θ-vectors

Step (3) of DP: $\theta_n^{a,o} = \frac{r(a)}{|\Omega|} + \gamma P^{a,o} \theta_{n-1}^{a,o}(b^{a,o})$
With *any* θ_{n-1} instead of *THE BEST* $\theta_{n-1}^{a,o}(b^{a,o})$

$$\widetilde{\theta}^{a,o} = \frac{r(a)}{|\Omega|} + \gamma P^{a,o} \theta_{n-1},$$

$\rightsquigarrow$ Family of θ-vector

Neighbor of $\theta_n^a = \sum_{o \in \Omega} \theta_n^{a,o}$

$\nu = \tilde{\theta}_n^{a,o'} + \sum_{o \neq o'} \theta_n^{a,o}$ where $\tilde{\theta}_n^{a,o'} \neq \theta_n^{a,o'}$

Theorem

For a belief state b, there exists a "best" θ-vector iff it is also the case for one of its neighbor.

Back to WITNESS

Find WITNESS θ-vectors

Algorithm 1: FindVecInRegion(θ, Θ)

Input: A representation Θ, a θ-vector $\theta \in \Theta$

Output: A point of the region or **null**

LP $\leftarrow$ `SetUpLinearProgram` (θ, Θ)
`SolveLinearProg` (LP)
if `NoSolution` *(*LP*)* **then**
    return null
end
if *val(*LP*)* ≤ 0 **then**
    return null
end
return `Solution` *(*LP*)*

Back to WITNESS

Algorithm 2: SetUpLinearProgram(θ, Θ)

Input: A representation Θ, a θ-vector $\theta \in \Theta$

Output: A Linear Program Problem

solve

$$\max_{\mathbb{R}} \epsilon$$

with

$$x.(\theta - \tilde{\theta}) \geq \epsilon,\ \forall \tilde{\theta} \in \Theta,\ \tilde{\theta} \neq \theta$$
$$x \in \Pi(\mathcal{S})$$

Find dominated θ-vectors

Algorithm 3: CheckDomination(Θ)

Input: A representation Θ
Output: A representation without any entirely dominated θ-vector

if $|\Theta|$ ¡ 2 **then**
  return Θ
end
$\tilde{\Theta} \leftarrow \emptyset$
repeat
  $\theta \leftarrow$ `RemoveElement`(Θ)
  if $\nexists \theta' \in \tilde{\Theta}$ *t.q.* $\theta' \geq \theta$ **then**
    $\tilde{\Theta} \leftarrow \{\theta' | \theta' \in \tilde{\Theta}, \ \theta \not\geq \theta'\}$
    $\tilde{\Theta} \leftarrow \tilde{\Theta} \cup \{\theta\}$
  end
until $\Theta = \emptyset$
return $\tilde{\Theta}$

Back to INCREMENTAL PRUNNING

Algorithm 4: Pruning($\tilde{\Theta}$)

Input: A representation $\tilde{\Theta}$ of V
Output: A parsimonious representation Θ of V

$\hat{\Theta} \leftarrow \emptyset$
while $\tilde{\Theta} \neq \emptyset$ **do**
  $\theta \leftarrow$ `RemoveElement`$(\tilde{\Theta})$
  $b \leftarrow$ `FindVectInRegion`$(\theta, \hat{\Theta})$
  if $b \neq$ **null then**
    $\tilde{\Theta} \leftarrow \tilde{\Theta} \cup \{\theta\}$
    $\theta^* \leftarrow$ `BestVector`$(\tilde{\Theta}, b)$
    $\tilde{\Theta} \leftarrow \tilde{\Theta} - \{\theta\}$
    $\hat{\Theta} \leftarrow \hat{\Theta} \cup \{\theta^*\}$
  end
end
$\Theta \leftarrow \hat{\Theta}$
return Θ

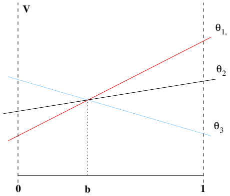
Check one θ-vector

Algorithm 5: BestVector(Θ, b)

Input: A representation Θ, a belief state b

Output: The best θ-vector of Θ for this state

$v^* \leftarrow -\infty$
foreach $\theta \in \Theta$ **do**
  $v \leftarrow b.\theta$
  if $v = v^*$ **then**
    $v^* \leftarrow$ `LexicographicMaximum`(θ^*, θ)
  end
  if $v > v^*$ **then**
    $v^* \leftarrow v$
    $\theta^* \leftarrow \theta$
  end
end
return θ^*



Algorithm 6: LexicographicMaximum(θ, $\tilde{\theta}$)

Input: Two θ-vectors θ and $\tilde{\theta}$ from Θ

Output: The lexicographic maximum of the two θ-vectors

foreach $s \in \mathcal{S}$ **do**
  if $\theta(s) > \tilde{\theta}(s)$ **then**
    return θ
  end
  if $\theta(s) < \tilde{\theta}(s)$ **then**
    return $\tilde{\theta}$
  end
end
return θ