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

POMDP

Examples
Formalism
Problem
Adapted POMDP
Outline

POMDP
  Examples
  Formalism
  Problem
  Adapted POMDP

Exact resolution
  Belief states
  DP Operator
  Value Iteration
  Policy Iteration

Approx. / Learning

Conclusion
Outline

POMDP

Examples
Formalism
Problem
Adapted POMDP

Exact resolution

Belief states
DP Operator
Value Iteration
Policy Iteration

Approximate solutions and Learning

Approximate solutions
Learning
Predictive State Representation
Outline

POMDP
  Examples
  Formalism
  Problem
  Adapted POMDP

Exact resolution
  Belief states
  DP Operator
  Value Iteration
  Policy Iteration

Approximate solutions and Learning
  Approximate solutions
  Learning
  Predictive State Representation

Conclusion
Outline

POMDP

Examples
Formalism
Problem
Adapted POMDP

Exact resolution
Belief states
DP Operator
Value Iteration
Policy Iteration

Approximate solutions and Learning
Approximate solutions
Learning
Predictive State Representation

Conclusion
Cheese Maze
Cheese Maze
Cheese Maze

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

States 1–11
Cheese Maze

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

States 1–11

Observations A–G
Tiger problem

**S0**
- "tiger-left"
- Pr(o=TL | S0, listen)=0.85
- Pr(o=TR | S1, listen)=0.15

**S1**
- "tiger-right"
- Pr(o=TL | S0, listen)=0.15
- Pr(o=TR | S1, listen)=0.85

**Actions**=
- 0: listen,
- 1: open-left,
- 2: open-right

**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

POMDP

Examples

Formalism

Problem

Adapted POMDP

Exact resolution

Belief states

DP Operator

Value Iteration

Policy Iteration

Approximate solutions and Learning

Approximate solutions

Learning

Predictive State Representation

Conclusion
Partially Observable Markov Decision Process

- $S$: state space
- $\Omega$: observation space
- $A$: action space
- $T: S \times A \rightarrow \Pi(S)$ transition function
- $O: S \times A \times S \rightarrow \Pi(\Omega)$ observation function
- $r: S \times A \times S \rightarrow \mathbb{R}$ reward function
- $(b_0$: initial state distribution)

- A POMDP describes a problem, not a solution/behavior/policy
Partially Observable Markov Decision Process

- $S$: state space
- $\Omega$: observation space
- $A$: action space
- $T: S \times A \rightarrow \Pi(S)$ transition function
- $O: S \times A \times S \rightarrow \Pi(\Omega)$ observation function
- $r: S \times A \times S \rightarrow \mathbb{R}$ reward function
- $(b_0: \text{initial state distribution})$

- A POMDP describes a problem, not a solution/behavior/policy
Partially Observable Markov Decision Process

- $S$: state space
- $\Omega$: observation space
- $A$: action space
- $T: S \times A \rightarrow \Pi(S)$: transition function
- $O: S \times A \times S \rightarrow \Pi(\Omega)$: observation function
- $r: S \times A \times S \rightarrow \mathbb{R}$: reward function
- $(b_0$: initial state distribution)

- A POMDP describes a problem, not a solution/behavior/policy
Partially Observable Markov Decision Process

- $S$: state space
- $\Omega$: observation space
- $A$: action space
- $T: S \times A \rightarrow \Pi(S)$ transition function
- $O: S \times A \times S \rightarrow \Pi(\Omega)$ observation function
- $r: S \times A \times S \rightarrow \mathbb{R}$ 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

### POMDP
- Examples
- Formalism
- **Problem**
- Adapted POMDP

### Exact resolution
- Belief states
- DP Operator
- Value Iteration
- Policy Iteration

### Approximate solutions and Learning
- Approximate solutions
- Learning
- Predictive State Representation

### Conclusion
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
- convergence of “naïve” classical MDP algorithms
- belief state as valid/useful information state
- Planification: when a model is known
  - Witness algorithm
  - Incremental pruning algorithm
- Learning: when the model is unknown
  - learning useful state extensions
  - learning the model
  - using Predictive State Representation
Outline

POMDP

Examples
Formalism
Problem

Adapted POMDP

Exact resolution
Belief states
DP Operator
Value Iteration
Policy Iteration

Approximate solutions and Learning
Approximate solutions
Learning
Predictive State Representation

Conclusion
Stochastic Memoryless Policy

Stochastic memoryless policy can be arbitrarily better than a deterministic memoryless policy.
[Singh et al., 1994]
No memoryless policy leads to an optimal “adapted” value function. [Singh et al., 1994]

$$\psi^{\pi}(o) = \sum_{s \in S} \Pr^{\pi}(s|o)V^{\pi'}(s)$$
Convergence of “classical” algorithms [Jaakkola et al., 1994]

- **TD(0)**

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

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

- **Q-Learning**

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

where $$\text{Pr}^{\pi \text{exp}} (s|o, a)$$ is the asymptotic occupation probability and where $$\text{Pr}^a (s, o') = \sum_{s' \in S} T(s'|s, a) O(o'|s').$$
Finding the best memoryless policy [Jaakkola et al., 1994]

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

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

\( \Rightarrow \) local maximum of scalar value function
Outline

POMDP
  Examples
  Formalism
  Problem
  Adapted POMDP

Exact resolution
  Belief states
  DP Operator
  Value Iteration
  Policy Iteration

Approximate solutions and Learning
  Approximate solutions
  Learning
  Predictive State Representation

Conclusion
Belief States: sufficient statistics of past

Belief States

distribution on states: \( b_t(s) = \Pr(s_t = s) \)
Belief States: sufficient statistics of past

<table>
<thead>
<tr>
<th>Belief States</th>
</tr>
</thead>
<tbody>
<tr>
<td>distribution on states: ( b_t(s) = \Pr(s_t = s) )</td>
</tr>
</tbody>
</table>

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

- Bayesian update:
  \[
  b_o^a(s') = \Pr(s' | b, a, o) = \frac{O(o | s') \sum_{s \in S} T(s' | s, a)b(s)}{\sum_{s \in S} \sum_{s'' \in S} O(o | s'') T(s'' | s, a)b(s)}.
  \]
- defines a (continuous) MDP which can be solved [Aström, 1965].
Policy Tree

For *belief states*

- Deterministic optimal policy
- Kind of Conditionnal Plan.

⇝ tree representation
Piece-Wise Linear Convex Value Function

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

- Finite horizon \( n \)
- Vector = one policy...
Outline

POMDP
- Examples
- Formalism
- Problem
- Adapted POMDP

Exact resolution
- Belief states
- DP Operator
- Value Iteration
- Policy Iteration

Approximate solutions and Learning
- Approximate solutions
- Learning
- Predictive State Representation

Conclusion
Dynamic programming on PWLC value function (1)

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

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

A $\theta$ is mapped to a policy
Dynamic programming on PWLC value function (2)

PWLC $V_n$ at step $n$ for first action $a_1$ and observation $o_1$

$\theta_{n-1,0}^{a_1,o_1}(b,a_1,o_1,n-1,b,a_1,o_1,n) = \frac{r(s,a_1)}{|\Omega|} + \gamma \sum_{s' \in S} T(s,a_1,s')O(s',o_1)\theta_{n-1}^{a_1,o_1}(b^{a_1,o_1},s).$
Dynamic programming on PWLC value function (3)

PWLC $V_n$ at step $n$ for first action $a_1$

$$
\theta_n^{a_1}(b) = \sum_{o \in \Omega} \theta_n^{a_1,o}(b).
$$

At most $|\Theta_{n-1}| |\Omega|$ vectors
Dynamic programming on PWLC value function (4)

PWLC $V_n$ at step $n$

$$\theta_n(b) = \max_{a \in A} \theta_n^a(b)$$
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**
  - \( \epsilon \)-optimal value function is PWLC
  - Optimal only for *transient* POMDP [Sondik, 1971]

\[ \Rightarrow \text{the real problem is the size of vector space.} \]
PWLC Value Function with Belief States

- **Finite Horizon POMDP**
  - Optimal value function is PWLC
    - \[ V_n(b) = \max_{\theta \in \Theta_n} b \cdot \theta \]

- **Infinite Horizon POMDP**
  - \(\epsilon\)-optimal value function is PWLC
  - Optimal only for *transient* POMDP [Sondik, 1971]

\[ \Rightarrow \text{the real problem is the size of vector space.} \]
Parcimonious representation

\[ \theta \text{ from } \Theta \text{ dominated : } b.\theta \leq \max_{\theta' \in \Theta} b.\theta'. \]

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

\[ \theta \text{ from } \Theta \text{ dominated} : \ b.\theta \leq \max_{\theta' \in \Theta} b.\theta'. \]

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

\[ \theta \text{ from } \Theta \text{ dominated : } b.\theta \leq \max_{\theta' \in \Theta} b.\theta'. \]

- Exists a minimal representation [Littman and Szepesvári, 1996]
  - \( \theta_2 \): entirely dominated
  - \( \theta_4 \): needs PRUNING
Outline

POMDP
  Examples
  Formalism
  Problem
  Adapted POMDP

Exact resolution
  Belief states
  DP Operator
  Value Iteration
  Policy Iteration

Approximate solutions and Learning
  Approximate solutions
  Learning
  Predictive State Representation

Conclusion
Algorithm \textbf{WITNESS}: concepts [Cassandra et al., 1994]

- Incremental build of parcimonious representation.

1. Start from belief state $b$
2. Look for the best vector of its region
3. Add all “neighbors” to the agenda $\Upsilon$
   - Either remove from it (dominated vector)
   - Or add best vector from region to $V$ and its neighbors to $\Upsilon$.
4. Loop
Algorithm **WITNESS**($\Theta_{n-1}, a$)

**Input:** A parsimonious representation $\Theta_{n-1}$ of $V_{n-1}^*$, an action $a$

**Output:** A parsimonious representation $V_{n-1}^{*,a}$

$b \leftarrow$ a belief state of $B$

$\hat{\Theta} \leftarrow \{\theta_n^a(b)\}$

$\Upsilon \leftarrow N(\theta_n^a(b))$

while $\Upsilon \neq \emptyset$ do
  $v \leftarrow$ RemoveElement($\Upsilon$)
  if $v \in \hat{\Theta}$ then
    $b \leftarrow$ null
  else
    $b \leftarrow$ FindVectInRegion($v, \hat{\Theta}$)
  end
  if $b \neq$ null then
    $\hat{\Theta} \leftarrow \hat{\Theta} \cup \{\theta_n^a(b)\}$
    $\Upsilon \leftarrow \Upsilon \cup \{v\}$
    $\Upsilon \leftarrow \Upsilon \cup N(\theta_n^a(b))$
  end
end

$\Theta_n^a \leftarrow \hat{\Theta}$

return $\Theta_n^a$
Incremental Pruning: concept [Zang and Lio, 1996]

- Lots of small pruning vs global final pruning.

1. With the set $\Psi$ of all sets of $\Theta_n^{a,o}$ vectors.
2. Take two sets from it and prune them.
3. Add new prunned set to $\Psi$, loop.
**Algo IncrementalPruning**($\Theta_{n-1}, a$)

**Input:** A parsimonious representation $\Theta_{n-1}$ of $V_{n-1}^*$, an action $a$

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

\[
\psi \leftarrow \bigcup_o \{\Theta_{n,o}^a\}
\]

while $|\psi| > 1$ do

\[
A \leftarrow \text{RemoveElement}(\psi)
\]
\[
B \leftarrow \text{RemoveElement}(\psi)
\]
\[
D \leftarrow \text{PRUNE}(A \oplus B)
\]
\[
\psi \leftarrow \psi \cup \{D\}
\]

end

return $\psi$
Outline

POMDP
- Examples
- Formalism
- Problem
- Adapted POMDP

Exact resolution
- Belief states
- DP Operator
- Value Iteration
  - Policy Iteration

Approximate solutions and Learning
- Approximate solutions
- Learning
- Predictive State Representation

Conclusion
Policy Iteration: concepts [Hansen, 1998]

- Grow an $\epsilon$-optimal FSA controller

1. From a given FSA $\delta$ compute all new vectors
2. For each new vector
   2.1 If exists in $\delta$, 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 vector
Algorithm Policy Iteration($\delta, \epsilon$)

**Input:** A finite state controller $\delta$ and a positive real $\epsilon$

**Output:** A finite state controller $\delta^*$ which is $\epsilon$-optimal

**repeat**

Compute $V^\delta$ from $\delta$ by solving equations (??)

Build $\hat{V}^\delta \leftarrow \text{DynamicProgOperator}(V^\delta)$

$\hat{\delta} \leftarrow \emptyset$

foreach $\hat{\theta}^j \in \hat{V}^\delta$ do

if there exists a node $i$ of $\delta$ 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 $\theta^i$ 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 $\delta$ 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

POMDP
- Examples
- Formalism
- Problem
- Adapted POMDP

Exact resolution
- Belief states
- DP Operator
- Value Iteration
- Policy Iteration

Approximate solutions and Learning
- Approximate solutions
- Learning
- Predictive State Representation

Conclusion
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

POMDP
  Examples
  Formalism
  Problem
  Adapted POMDP

Exact resolution
  Belief states
  DP Operator
  Value Iteration
  Policy Iteration

Approximate solutions and Learning
  Approximate solutions
  Learning
  Predictive State Representation

Conclusion
Learn State Extensions

POMPD ≡ 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

\[
\begin{align*}
  z_{t+1} &= \gamma z_t + \frac{\nabla \mu_{a_t}(\theta, o_t)}{\mu_{a_t}(\theta, o_t)} \\
  \Delta_{t+1} &= \Delta_t + \frac{1}{t+1}[r_{t+1}z_{t+1} - \Delta_t]
\end{align*}
\]

- Interlaced with policy improvement with gradient ascent.

\[
\theta_{t+1} = \theta_t + \alpha \Delta_{t+1}
\]

\( \Rightarrow \) local optimum
Outline

POMDP
- Examples
- Formalism
- Problem
- Adapted POMDP

Exact resolution
- Belief states
- DP Operator
- Value Iteration
- Policy Iteration

Approximate solutions and Learning
- Approximate solutions
- Learning
  - Predictive State Representation

Conclusion
Predictive State Representation

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

\[
(1 \times q) \text{ prediction vector } p(h) = \{\text{Pr}(t_1|h), \text{Pr}(t_2|h), \ldots, \text{Pr}(t_q|h)\} \text{ iff } \\
\forall h, \text{Pr}(t|h) = f_t(p(h))
\]
Predictive State Representation

- a test: \( t_i = o_1a_1o_2 \ldots o_n \)
- prediction. history \( h: \Pr(o_1, \ldots, o_n|h, a_1, \ldots, a_{n-1}) \)
- set of tests: \( Q = \{t_i\}_{i=1, \ldots, q} \)

**Predictive State Representation**

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

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

**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.
Learning PSRs [Singh et al., 2003]

- How to maintain correct predictions for the tests
  $\sim m_{aot_i}$ 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 $\leadsto$ core-tests $Q_{T_1}$
- look for independant rows $\leadsto$ core-histories $Q_{H_1}$
- build new $\mathcal{D} = (Q_{T_1} \cup Q_{T_1}^{+ao}) \bowtie (Q_{H_1} \cup Q_{H_1}^{+ao})$
- 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 $\leadsto (|A||\Omega|)^n$
- POMDP, HMM $\leadsto < |S|$
- POMDP $\subset$ PSR

\[
\begin{align*}
D &= \begin{bmatrix}
D(Q) \\
\vdots \\
h_1 = \emptyset \\
h_2 \\
h_i \\
\vdots 
\end{bmatrix}
\end{align*}
\]

\[
Q = \{q_1, q_2, \ldots, q_k\}
\]

core tests $t_1 \ldots \ldots \ldots \ldots t_j \ldots$
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 published)

http://www.pomdp.org/
Aström, K. (1965).
Optimal control of Markov decision processes with incomplete state estimation.

Reinforcement learning in POMDP’s via direct gradient ascent.
In Proc. 17th International Conf. on Machine Learning (ICML’00).

Acting optimally in partially observable stochastic domains.
In Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI).
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. (Édité par Olivier Buffet et Olivier Sigaud)*, volume 1 & 2.
Lavoisier - Hermes Science Publications.

Solving POMDPs by searching in policy space.
In *Proc. of the Fourteenth Conf. on Uncertainty in Artificial Intelligence (UAI’98)*.
Bibliography III

Reinforcement learning algorithm for partially observable markov decision problems.

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).

A generalized reinforcement-learning model: Convergence and applications.
In Proc. of the Thirteenth Int. Conf. on Machine Learning (ICML'96).
Bibliography IV

Reinforcement learning with selective perception and hidden state.

Point-based value iteration: An anytime algorithm for POMDPs.
In Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI'03), pages 1025 – 1032.

Markov Decision Processes: discrete stochastic dynamic programming.
John Wiley & Sons, Inc. New York, NY.
Memory-bounded dynamic programming for DEC-POMDPs.
In Proc. of the Twentieth Int. Joint Conf. on Artificial Intelligence (IJCAI’07).

Learning without state estimation in partially observable markovian decision processes.

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

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

Reinforcement Learning.

Planning in stochastic domains: Problem characteristics and approximation.
Neighbor 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 $\theta_{n-1}$ instead of THE BEST $\theta_{n-1}^{a,o}(b^{a,o})$

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

$\rightsquigarrow$ Family of 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” vector iff it is also the case for one of its neighbor.

$\text{Back}$ to WITNESS
Find WITNESS vectors

Algorithm 1: FindVecInRegion(θ, Θ)

Input: A representation Θ, a vector \( \theta \in \Theta \)

Output: A point of the region or null

\( LP \leftarrow \text{SetUpLinearProgram}(\theta, \Theta) \)

\( \text{SolveLinearProg}(LP) \)

if NoSolution \((LP)\) then
  return null
endif

if \( \text{val}(LP) \leq 0 \) then
  return null
endif

return Solution \((LP)\)

Algorithm 2: SetUpLinearProgram(θ, Θ)

Input: A representation Θ, a vector \( \theta \in \Theta \)

Output: A Linear Program Problem

solve

\[
\max_{\mathbb{R}} \epsilon \\
\text{with}
\]

\[
x.(\theta - \tilde{\theta}) \geq \epsilon, \quad \forall \tilde{\theta} \in \Theta, \quad \tilde{\theta} \neq \theta
\]

\( x \in \Pi(S) \)
Find dominated vectors

**Algorithm 3: CheckDomination(Θ)**

**Input:** A representation Θ  
**Output:** A representation without any entirely dominated vector

\[
\text{if } |Θ| \leq 2 \text{ then}  \\
\quad \text{return } Θ  \\
\text{end}  \\
\tilde{Θ} ← ∅  \\
\text{repeat}  \\
\quad θ ← \text{RemoveElement}(Θ)  \\
\quad \text{if } \exists θ' ∈ \tilde{Θ} \text{ t.q. } θ' ≥ θ \text{ then}  \\
\quad \quad \tilde{Θ} ← \{θ'|θ' ∈ \tilde{Θ}, θ ≥ θ'\}  \\
\quad \quad \tilde{Θ} ← \tilde{Θ} ∪ \{θ\}  \\
\quad \text{end}  \\
\text{until } Θ = ∅  \\
\text{return } \tilde{Θ}
\]

**Algorithm 4: Pruning(̂Θ)**

**Input:** A representation ̂Θ of V  
**Output:** A parsimonious representation ̂Θ of V

\[
\hat{Θ} ← ∅  \\
\text{while } \tilde{Θ} ≠ ∅ \text{ do}  \\
\quad θ ← \text{RemoveElement}(\tilde{Θ})  \\
\quad b ← \text{FindVectInRegion}(θ, ̂Θ)  \\
\quad \text{if } b ≠ \text{null then}  \\
\quad \quad ̂Θ ← ̂Θ ∪ \{θ\}  \\
\quad \quad θ^* ← \text{BestVector}(\tilde{Θ}, b)  \\
\quad \quad ̂Θ ← ̂Θ − \{θ\}  \\
\quad \quad ̂Θ ← ̂Θ ∪ \{θ^*\}  \\
\quad \text{end}  \\
\text{end}  \\
\text{return } ̂Θ
\]

*Back to Incremental Prunning*
Check one vector

Algorithm 5: BestVector(Θ, b)

Input: A representation Θ, a belief state b
Output: The best vector of Θ for this state

\[ v^* \leftarrow -\infty \]

\textbf{foreach} θ ∈ Θ \textbf{do}
\[ v \leftarrow b.\theta \]
\textbf{if} \ v = v^* \textbf{then}
\[ v^* \leftarrow \]
\textbf{LexicographicMaximum}(\theta^*, \theta)
\textbf{end}
\textbf{if} \ v > v^* \textbf{then}
\[ v^* \leftarrow v \]
\[ \theta^* \leftarrow \theta \]
\textbf{end}
\textbf{end}

\textbf{return} \ θ^*

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

Input: Two vectors θ and \tilde{\theta} from Θ
Output: The lexicographic maximum of the two vectors

\textbf{foreach} s ∈ S \textbf{do}
\textbf{if} \ θ(s) > \tilde{\theta}(s) \textbf{then}
\textbf{return} \ θ
\textbf{end}
\textbf{if} \ θ(s) < \tilde{\theta}(s) \textbf{then}
\textbf{return} \ \tilde{\theta}
\textbf{end}
\textbf{end}

\textbf{return} \ θ