Online Policy Improvement in Large POMDPs via an Error Minimization Search

Stéphane Ross, Brahim Chaib-draa & Joelle Pineau

School of Computer Science
McGill University, Montreal, Canada

April 14th, 2007

McGill

Problem

A POMDP is a model for planning in partially observable stochastic domains.

Many problems can be represented by POMDPs :

- Robot navigation
- Human-Computer speech interface
- Medical diagnosis
- Military defense system
- etc . . .

But few can be solved . . .

McGill

Outline

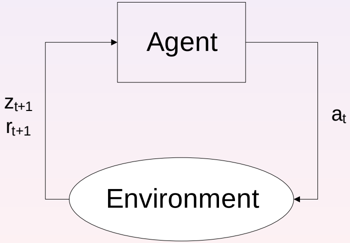
Plan

1. POMDP
2. Online Search Algorithms
3. AEMS : Anytime Error Minimization Search
   - Motivation
   - Error Contribution
   - Heuristic Search
4. Experiments
5. Future Work

Partially Observable Markov Decision Process

A POMDP is defined by a tuple : $< S, A, Z, R, T, O, \gamma, b_0 >$

- States : S
- Actions : A
- Observations : Z
- Rewards : $R(s, a)$
- Transition : $T(s, a, s') = P(s'|s, a)$
- Perception : $O(s', a, z) = P(z|s', a)$
- Discount : $\gamma \in [0, 1)$
- Initial belief : b_0



Belief State

Probability distribution over states.

Sufficient statistic of the complete history :

- $b_t(s) = P(s_t = s | b_0, a_0, z_1, a_1, z_2, \ldots, a_{t-1}, z_t)$

It can be maintained easily after each step :
$b_{t+1} = \tau(b_t, a_t, z_{t+1})$

- $b_{t+1}(s') = \frac{O(s', a_t, z_{t+1}) \sum_{s \in S} T(s, a_t, s') b_t(s)}{P(z_{t+1} | b_t, a_t)}$
- $P(z_{t+1} | b_t, a_t) = \sum_{s' \in S} O(s', a_t, z_{t+1}) \sum_{s \in S} T(s, a_t, s') b_t(s)$

Policy & Value Function

A policy maps belief states to actions.

We seek the optimal policy π^* :

- $\pi^* = \underset{\pi \in \Pi}{\arg\max}\ E(\sum_{t=0}^{\infty} \gamma^t r_t | b_0, \pi)$

V^* defines the expected rewards obtained by π^* from belief b :

- $V^*(b) = \max_{a \in A} \left[R(b,a) + \gamma \sum_{z \in Z} P(z|b,a) V^*(\tau(b,a,z)) \right]$
- $R(b,a) = \sum_{s \in S} b(s) R(s,a)$

McGill

Offline vs. Online Solvers

Offline : Computes π for all beliefs before the execution.

- ✓ Few computations during execution.
- ✗ Takes a lot of computation before execution.

Online : Computes best action in current belief during the execution.

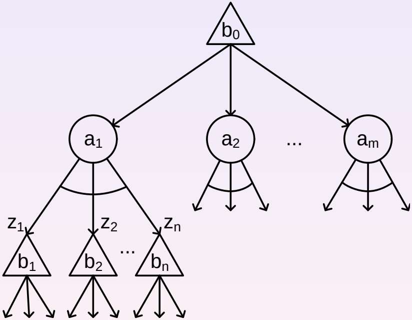
- ✓ Immediatly executable.
- ✗ More computations required during execution.
- ✗ Planning time limited by real-time constraints.

Plan

1. POMDP
2. Online Search Algorithms
3. AEMS : Anytime Error Minimization Search
   - Motivation
   - Error Contribution
   - Heuristic Search
4. Experiments
5. Future Work

Online Search Algorithms

Online search algorithms proceed by constructing an AND/OR tree of the reachable belief states, from the curent belief b_0 :



Online Search Algorithms

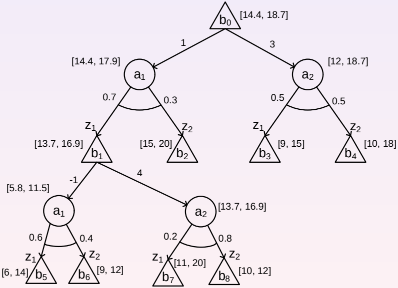
Approximate value functions are used at the fringe nodes :

- Lower Bounds :
  - Blind policy
  - PBVI style algorithms
- Upper Bounds :
  - MDP
  - QMDP
  - FIB
  - Grid based algorithms

Online Search Algorithms

Values of parent nodes are obtained from their children values :

- Lower Bounds :
  - $L_T(b) = \max_{a \in A} L_T(b,a)$
  - $L_T(b,a) = R(b,a) + \gamma \sum_{z \in Z} P(z|b,a) L_T(\tau(b,a,z))$
- Upper Bounds :
  - $U_T(b) = \max_{a \in A} U_T(b,a)$
  - $U_T(b,a) = R(b,a) + \gamma \sum_{z \in Z} P(z|b,a) U_T(\tau(b,a,z))$



McGill

Online Search Algorithms

Once the search has terminated for b_0 :

- Execute the action $\widehat{a} = \arg\max_{a \in A} L_T(b_0, a)$
- Get a new observation z.
- Update the root of tree T.
- Resume the search in this new tree.

Plan

Motivation

Online search is useful to improve the offline policy.

How should we search to improve it the most ?

Can we do better than just a *k*-step lookahead ?

- Might explore paths with small probabilities.
- Might explore paths with small error.

➥ Variable depth search allows to get more precision where needed.

Motivation

Improve policy = Reduce its error.

What is the error of a policy ?

- The error in b_0 : $e_T(b_0) = V^*(b_0) - L_T(b_0)$
- This error comes from the fringe nodes.

How to reduce the error as quickly as possible ?

➥ Expand the fringe node that contributes the most to the error in b_0

McGill

Error contribution

Error contribution of fringe node b : $\gamma^{d(b,b_0)} P(h^b_{b_0} | b_0, \pi^*) e(b)$.

Problem : We cannot compute $P(h^b_{b_0} | b_0, \pi^*)$ and $e(b)$.

We can approximate them :

- $\hat{e}(b) = U(b) - L(b) \geq e(b)$
- $\hat{\pi}_T(b,a) = \begin{cases} 1 & \text{if } a = \arg\max_{a' \in A} U_T(b,a') \\ 0 & \text{otherwise} \end{cases}$

McGill

Heuristic Search

Heuristic : $\widetilde{b}(T) = \underset{b \in fringe(T)}{\arg\max} \, \gamma^{d(b,b_0)} P(h^b_{b_0} | b_0, \hat{\pi}_T) \hat{e}(b)$

Is this a good heuristic ?

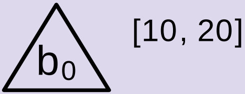
- Favors nodes reached sooner.
- Favors nodes reached by promising actions with high probabilities.
- Favors nodes with large error on their values.

AEMS : Best-first-search using $\widetilde{b}(T)$ as heuristic.

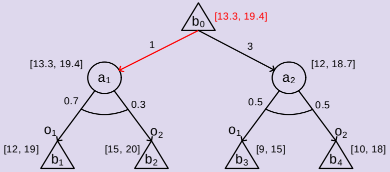
Guaranteed to find an ϵ-optimal action within finite time if $\hat{\pi}_T(b, a)$ is non-zero for $a = \arg\max_{a' \in A} U_T(b, a')$.

Exemple

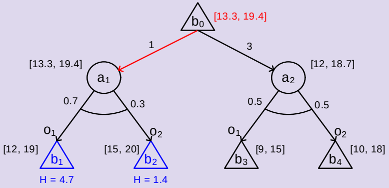
Choice 1^{st} iteration



Expand 1^{st} iteration

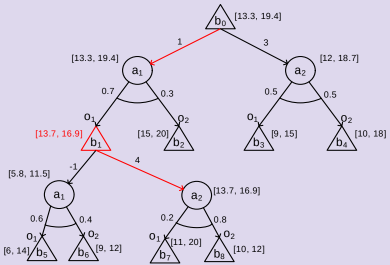


Choice 2^{nd} iteration

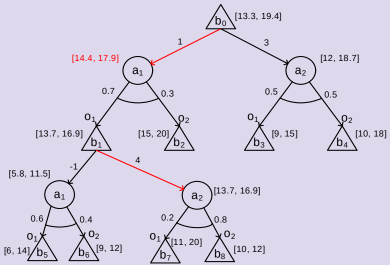


Example

Expand 2^{nd} iteration



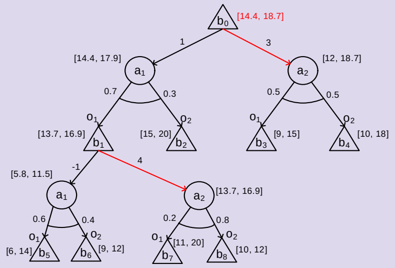
Update 2^{nd} iteration



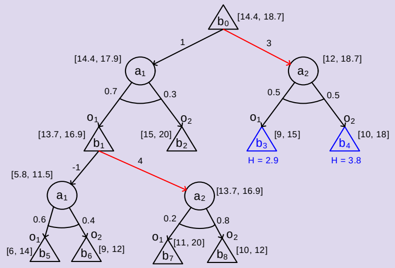
McGill

Example

Update 2^{nd} iteration



Choice 3^{rd} iteration



Plan

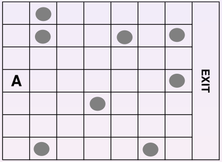
1. POMDP
2. Online Search Algorithms
3. AEMS : Anytime Error Minimization Search
   - Motivation
   - Error Contribution
   - Heuristic Search
4. Experiments
5. Future Work

RockSample[7,8]

A robot that must sample good rocks. The state of each rock (good or bad) can be observed through a noisy sensor.

$|S| = 12545, |A| = 13, |Z| = 2$

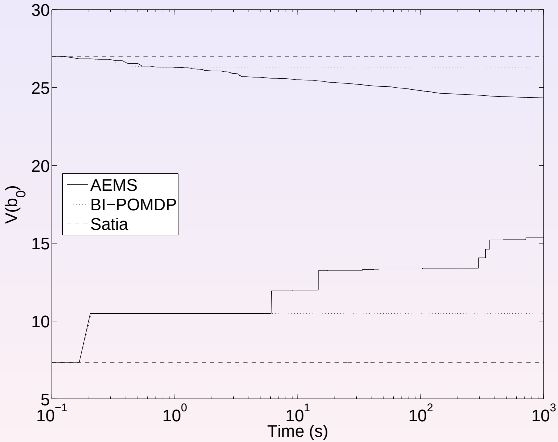
Method	Reward	Offline Time (s)	Online Time (s)
Blind	7.4	4	-
$\text{Satia}^{QMDP}_{Blind}$	7.4	29	0.889
PBVI	7.7	2418	-
Perseus	8.3	36000	-
RTDP-BEL	8.7	8362	-
$\text{RTBSS(2)}^{QMDP}_{Blind}$	10.3	29	0.896
HSVI	15.1	10266	-
QMDP	15.5	25	-
$\text{BI-POMDP}^{QMDP}_{Blind}$	18.4	29	0.955
RTBSS(2)^{QMDP}	20.3	25	0.320
HSVI2	20.6	1003	-
$\textbf{AEMS}^{QMDP}_{Blind}$	**20.8**	**29**	**0.884**



McGill

Convergence

Convergence of the lower and upper bounds with different online search algorithms in RockSample[7,8] :



Plan

McGill

Future Work

- Explore different variants of $\hat{\pi}_T$
  - ➥ We could try several exploration policies already used in RL, e.g. Boltzmann, ϵ-greedy, etc.
  - Learn $\hat{\pi}_T$ from previous search ?
- Update the bounds computed offline after every search ?

McGill

Questions

?