AEMS: An Anytime Online Search Algorithm for Approximate Policy Refinement in Large POMDPs

Stéphane Ross & Brahim Chaib-draa
Department of Computer Science and Software Engineering
Laval University, Quebec, Canada, G1K 7P4
{ross,chaib}@damas.ift.ulaval.ca

Abstract

Solving large Partially Observable Markov Decision Processes (POMDPs) is a complex task which is often intractable. A lot of effort has been made to develop approximate offline algorithms to solve ever larger POMDPs. However, even state-of-the-art approaches fail to solve large POMDPs in reasonable time. Recent developments in online POMDP search suggest that combining offline computations with online computations is often more efficient and can also considerably reduce the error made by approximate policies computed offline. In the same vein, we propose a new anytime online search algorithm which seeks to minimize, as efficiently as possible, the error made by an approximate value function computed offline. In addition, we show how previous online computations can be reused in following time steps in order to prevent redundant computations. Our preliminary results indicate that our approach is able to tackle large state space and observation space efficiently and under real-time constraints.

1 Introduction

The POMDP framework provides a powerful model for sequential decision making under uncertainty. However, most real world applications have huge state space and observation space, such that exact solving approaches are completely intractable (finite-horizon POMDPs are PSPACE-complete [Papadimitriou and Tsitsiklis, 1987] and infinite-horizon POMDPs are undecidable [Madani *et al.*, 1999]).

Most of the recent research in the area has focused on developing new offline approximate algorithms that can find approximate policies for larger POMDPs [Braziunas and Boutilier, 2004; Pineau *et al.*, 2003; Poupart, 2005; Smith and Simmons, 2005; Spaan and Vlassis, 2005]. Still, successful application of POMDPs to real world problems has been limited due to the fact that even these approximate algorithms are intractable in the huge state space of real world applications. One of the main drawbacks of these offline approaches is that they need to compute a policy over the whole belief state space. In fact, a lot of these computations are generally not necessary since the agent will only visit a small subset of belief states when acting in the environment. This is the strategy online POMDP algorithms tries to exploit [Satia and Lave, 1973; Washington, 1997; Geffner and Bonet, 1998; McAllester and Singh, 1999; Paquet *et al.*, 2005]. Since we only need to plan for the current belief state when acting online, one needs only to compute the best action to do in this belief state, considering the subset of belief states that can be reached over some finite planning horizon.

One drawback of online planning is that it generally needs to meet *hard* real-time constraints when one is in face of large POMDPs. Nevertheless, recent developments in online POMDP search algorithms [Paquet *et al.*, 2005; 2006] suggest that combining approximate offline and online solving approaches may be the most efficient way to tackle large POMDPs. Effectively, we can generally compute a very approximate policy offline using standard offline value iteration algorithms and then use this approximate value function as a heuristic function in an online search algorithm. Using this combination enables online search algorithm to plan on shorter horizons in order to respect online real-time constraints and retain a good precision. Furthermore, doing an exact online search on a certain horizon also reduces the error made by approximate value functions, and consequently, does not require as much precision in the value function to be efficient.

In this paper, we propose a new anytime online search algorithm which aims to reduce, as efficiently as possible, the error made by approximate offline value iteration algorithms. Our algorithm can be combined with any approximate offline value iteration algorithm to refine and improve the approximate policies computed by such algorithm. It can also be used alone, as a simple online search algorithm that can be applied in stationary or dynamic environments.

We first introduce the POMDP model and some offline and online approximate solving approaches. Then we present our new algorithm and some experimental results which show its efficiency.

2 POMDP Model

In this section we introduce the POMDP model and present different approximate offline and online approaches to solve POMDPs.

2.1 Model

A Partially Observable Markov Decision Process (POMDP) is a model for sequential decision making under uncertainty. Using such a model, an agent can plan an optimal sequence of action according to its belief by taking into account the uncertainty associated with its actions and observations.

A POMDP is generally defined by a tuple $(S, A, \Omega, T, R, O, \gamma)$ where S is the state space, A is the action set, Ω is the observation set, $T(s, a, s') : S \times A \times S \rightarrow [0, 1]$ is the transition function which specifies the probability of ending up in a certain state s', given that we were in state s and did action a, $R(s, a) : S \times A \rightarrow \Re$ is the reward function which specifies the immediate reward obtained by doing action a in state s, $O(o, a, s') : \Omega \times A \times S \rightarrow [0, 1]$ is the observation function which specifies the probability of observing a certain observation o, given that we did action a and ended in state s' and γ is the discount factor.

In a POMDP, the agent does not know exactly in which state it currently is, since its observations on its current state are uncertain. Instead the agent maintains a belief state b which is a probability distribution over all states that specifies the probability that the agent is in each state. After the agent performs an action a and perceives an observation o, the agent can update its current belief state b using the belief update function $\tau(b, a, o)$ specified in equation 1.

$$b'(s') = \eta O(o, a, s') \sum_{s \in S} T(s, a, s') b(s) \tag{1}$$

Here, b' is the new belief state and b is the last belief state of the agent. The summation part specifies the expected probability of transiting in state s', given that we performed action a and belief state b. Afterward, this expected probability is weighted by the probability that the agent observed o in state s' after doing action a. η is a normalization constant such that the new probability distribution over all states sums to 1.

Solving a POMDP consists in finding an optimal policy π^* which specifies the best action to do in every belief state b. This optimal policy depends on the planning horizon and on the discount factor used. In order to find this optimal policy, we need to compute the optimal value of a belief state over the planning horizon. For the infinite horizon, the optimal value function is the fixed point of equation 2.

$$V^*(b) = \max_{a \in A} R(b, a) + \gamma \sum_{o \in \Omega} P(o|b, a) V^*(\tau(b, a, o)) \tag{2}$$

In this equation, $R(b, a)$ is the expected immediate reward of doing action a in belief state b and $P(o|b, a)$ is the probability of observing o after doing action a in belief state b. This probability can be computed using equation 3.

$$P(o|b, a) = \sum_{s' \in S} O(o, a, s') \sum_{s \in S} T(s, a, s') b(s) \tag{3}$$

This equation is very similar to the belief update function, except that it needs to sum over all the possible resulting states s' in order to consider the global probability of observing o over all the state space.

Similarly to the definition of the optimal value function, we can define the optimal policy π^* as in equation 4.

$$\pi^*(b) = \arg\max_{a \in A} R(b, a) + \gamma \sum_{o \in \Omega} P(o|b, a) V^*(\tau(b, a, o)) \tag{4}$$

However, one problem with this formulation is that there is an infinite number of belief states and as a consequence, it would be impossible to compute such a policy for all belief states in a finite amount of time. But, since it has been shown that the optimal value function of a POMDP is piecewise linear and convex, we can define the optimal value function and policy of a finite-horizon POMDP using a finite set of S-dimensional hyper plan, called α-vector, over the belief state space. This is how exact offline value iteration algorithms are able to compute V^* in a finite amount of time. However, exact value iteration algorithms can only be applied to small problems of 10 to 20 states due to their high complexity. For more detail, refer to Littman and Cassandra [Littman, 1996; Cassandra *et al.*, 1997].

2.2 Approximate Offline algorithms

Contrary to exact value iteration algorithms, approximate value iteration algorithms try to keep only a subset of α-vectors after each iteration of the algorithm in order to limit the complexity of the algorithm. Pineau [Pineau *et al.*, 2003; Pineau, 2004] has developed a point based value iteration algorithm (PBVI) which bounds the complexity of exact value iteration to the number of belief points in its set. Instead of keeping all the α-vectors as in exact value iteration, PBVI only keeps a maximum of one α-vector per belief point, that maximizes its value. Therefore, the precision of the algorithm depends on the number of belief points and the location of the chosen belief points. Spaan [Spaan and Vlassis, 2005] has adopted a similar approach (Perseus), but instead of updating all belief points at each iteration, Perseus updates only the belief points which have not been improved by a previous α-vector update in the current iteration. Since Perseus generally updates only a small subset of belief points at each turn, it can converge more rapidly to an approximate policy, or use larger sets of belief points, which improves its precision. Another recent approach which has shown interesting efficiency is HSVI [Smith and Simmons, 2004; 2005], which maintains both an upper bound defined by a set of points and a lower bound defined by α-vectors. HSVI uses an heuristic that approximates the error of the belief points in order to select the belief point on which to do value iteration updates. When it selects a belief to update, it also updates its upper bound using linear programming methods.

2.3 Approximate Online algorithms

Satia & Lave [Satia and Lave, 1973] developed the first online algorithm to solve POMDPs. Their heuristic search algorithm uses upper and lower bounds, computed offline, on the value function to conduct branch-and-bound pruning in the search tree. The POMDP is represented as an AND-OR graph in which belief states are OR-nodes and actions are AND-nodes. The root node (b_0) of such an AND-OR graph represents the

current belief state as it is presented in Figure 1. The authors suggested solving the underlying MDP to get an upper bound and to use the value function of any reasonable policy for the lower bound. The heuristic they proposed to guide their search algorithm will be compared to our proposed heuristic in section 3.1.

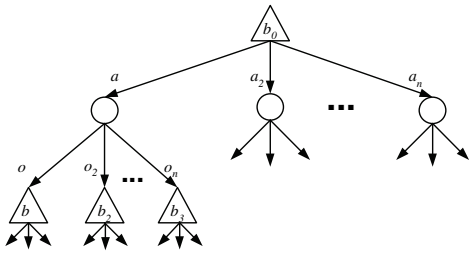

Figure 1: A search tree.

The BI-POMDP algorithm [Washington, 1997] uses the classic AO* algorithm [Nilsson, 1980] online to search the AND-OR graph. The author slightly modified AO* to use lower and upper bounds on the value function. For the lower bound, BI-POMDP pre-computes offline the min MDP (lowest possible value in every state), and uses this approximation at the fringe of the tree. For the upper bound, they use the QMDP algorithm [Littman *et al.*, 1995] to solve the underlying MDP and use this bound as an heuristic to direct the search of AO* toward promising actions. They also use the difference between the lower and upper bound to guide the search toward fringe nodes that require more precision.

The RTBSS algorithm [Paquet *et al.*, 2005] is another similar algorithm which uses a branch and bound technique to search the AND-OR graph from the current belief state online. The search in the tree is done in a depth-first-search fashion up to a certain pre-determined fixed depth. When it reaches this depth, it uses a lower bound heuristic to evaluate the long term value of the fringe belief state. RTBSS also uses an upper bound heuristic in order to prune some branches in the tree. Pruning is only possible when the upper bound value of doing an action is lower than the lower bound of another action in the same belief state.

Several other techniques have been proposed to conduct online search in POMDPs. Geffner [Geffner and Bonet, 1998] adapted the RTDP algorithm to POMDPs. This approach requires the belief state space to be discretized and generally needs a lot of learning time before it performs well. Another online search algorithm which uses observation sampling has also been proposed by McAllester [McAllester and Singh, 1999]. Instead of exploring all possible observations, this approach samples a pre-determined number of observations, at each AND-node from a generative model of the environment. A more recent online approach, called SOVI [G. Shani and Shimony, 2005], extended HSVI into an online value iteration algorithm. Its authors also proposed a few improvements to speed up the upper bound updates and evaluations. The main drawback of this approach is that it is hardly applicable online in large environments with real time constraints since it needs to do a value iteration with α-vectors online, and this has a very high complexity.

3 AEMS

We now present our new online search algorithm, called Anytime Error Minimization Search (AEMS). This algorithm aims to determine the best action to do in the current belief state by doing a look-ahead search. This search is done by exploring the tree of reachable belief states from the current belief state, by considering the different sequence of actions and observations. In this tree, belief states are represented as OR-nodes (we must choose an action child node) and actions are represented as AND-nodes (we must consider all belief state child nodes, associated to the different possible observations) as presented before in Figure 1.

This tree structure is used to determine the value of the current belief state b_0 and the best action to do in this belief state. The values of the actions and belief states in the tree are evaluated by backtracking the fringe belief state values, according to equation 2. However, since the search cannot be conducted on an infinite horizon, we use an approximate value function at the fringe of the tree to approximate the infinite-horizon value of these fringe belief states. As the tree is expanded, the estimate we will get for the current belief state b_0 is guaranteed to be more precise by the discount factor.

AEMS conducts the search by using a heuristic that provides an efficient way to minimize the error on the current belief state b_0 and to handle large observation space. AEMS is also able to reuse the computations done at previous time steps in order to prevent the redundant computations. Finally, AEMS is also an anytime algorithm which is able to exploit every bit of time available at each turn.

The key idea of AEMS consists in exploring the search tree by always expanding the fringe node that has the highest expected error contribution to the current belief state b_0. Expanding this belief state will reduce its error and will lead to better precision at the current belief state b_0 for which we are planning.

3.1 Expected Error evaluation

There are three key factors that influence the error introduced by a fringe belief state on the current belief state. The first is the actual error committed by using a lower bound value function instead of the exact value function to evaluate the value of the fringe belief state. In order to evaluate this error, we can compute the difference between our upper and lower bound to get the maximal possible error introduced by our lower bound function on the fringe belief state. We will refer to this approximation as the function $\hat{\epsilon}$ defined by equation 5.

$$\hat{\epsilon}(b) = U(b) - L(b) \tag{5}$$

Here, $U(b)$ is the upper bound on $V^*(b)$ and $L(b)$ is the lower bound on $V^*(b)$. The real error $\epsilon(b)$, which is defined by $\epsilon(b) = V^*(b) - L(b)$, is always lower or equal to our approximation $\hat{\epsilon}(b)$.

However, this error is multiplied by different factors, when it is backtracked into the search tree, that must be taken into account to get a good evaluation of its impact. By looking

back at equation 2, we notice that the value of a child belief state is always multiplied by the discount factor γ and the probability $P(o|b,a)$ of reaching this child belief state given the action taken in the parent belief state. Since these factors have a value in interval $[0,1]$, they reduce the contribution of this error on the parent belief state's value.

Another implicit factor that must be considered is the $\max$ operator, since it indicates that we need to consider only the values of belief states that can be reached by doing a sequence of optimal actions. In other words, if we know the optimal action in a certain belief state, than we would only need to pursue the search in this action's subtree, because the other action values will not be considered in the value of b_0. In our case, since we only have an approximate value function, we are generally not sure whether a certain action is optimal or not. Nevertheless, we can take this uncertainty into account by considering the probability that an action becomes the optimal action, given its current bounds. In particular, if the upper bound value of a certain action a is lower than the lower bound value of another action a' in a belief state, then we are sure that a is not the optimal action, (i.e., its probability of becoming the optimal action is 0). However, most of the time we might encounter cases where the upper bound is higher than the highest lower bound. To handle such case, we will assume the other actions lower bound fixed and we will assume that the exact value of the parent belief state is evenly distributed between its current lower and upper bounds. We could also consider other types of distributions, in particular if we know that a certain bound is more precise than the other. Using these assumptions, we can evaluate the probability that a certain action can still become the best action in the future using equation 6.

$$P(a|b) = \frac{U(a,b) - L(b)}{U(b) - L(b)} \tag{6}$$

Here $U(a,b)$ is the upper bound on the value of action a. $L(b)$ and $U(b)$ corresponds to the current lower and upper bound of belief state b, i.e. which can be obtained from the maximum lower and upper bound of the actions in belief state b. So basically, what we are computing is $P(U(a,b) \geq V^*(b) | V^*(b) \sim Uniform(L(b), U(b)))$, i.e. the probability that $U(a,b)$ is greater than $V^*(b)$, assuming that $V^*(b)$ follows a uniform distribution between $L(b)$ and $U(b)$. This formula is valid when $U(a,b) > L(b)$ and, as we mentioned earlier, if this is not the case, then $P(a|b) = 0$ because we are sure that action a will not be the optimal action. This probability can also be interpreted as the probability that we cannot prune action a in belief state b if $V^*(b) \sim Uniform(L(b), U(b))$. While $P(a|b)$, is not a probability distribution, it still gives a measure of how likely a certain action will not be pruned in the future and remain as the optimal action.

An alternative way to approximate the $\max$ operator would be to consider the current action with the highest upper bound as the optimal action. In such a case, we can use the alternative definition for $P(a|b)$ presented in equation 7.

$$P(a|b) = \begin{cases} 1 & \text{if } a = \arg\max_{a' \in A} U(a', b) \\ 0 & \text{otherwize} \end{cases} \tag{7}$$

Furthermore, if we want to know what the probability is, that a certain fringe belief state can be reached by a sequence of optimal actions, we can use the product rule to combine the probabilities of optimal action at each depth.

Combining all these factors, we find that the probability of reaching a certain fringe belief state b^d at depth d, denoted $P(b^d)$, can be computed using equation 8.

$$P(b^d) = \prod_{i=0}^{d-1} P(o^i|b^i, a^i) P(a^i|b^i) \tag{8}$$

In this equation, o^i, a^i and b^i denote the observation, action and belief state encountered at depth i that leads to belief state b^d at depth d.

Consequently, we can compute the expected error introduced by a certain fringe belief state b^d at depth d on the current belief state b_0 by using equation 9.

$$E(b^d) = \gamma^d P(b^d) \hat{\epsilon}(b^d) \tag{9}$$

Therefore, we can use equation 9 as an heuristic to choose the fringe node that contributes the most to the error in b_0. Since we propose two different definitions for the term $P(a|b)$, we will refer to $E(b^d)$ using equation 6 as the heuristic AEMS1 and $E(b^d)$ using equation 7 as the heuristic AEMS2.

Intuitively, $E(b^d)$ seems a sound heuristic to guide the search since it has several desired properties. First, it will favor exploration of nodes with loose bounds. Loose bounds generally indicate that at least one of them is flawed and therefore, exploring such node is generally important to get more precision and make better decisions afterwards. In addition, if we have very tight bounds on the value of a belief state then we do not need to search this belief state any longer since it would have a very low impact on the quality of our solution in belief state b_0. Moreover, $E(b^d)$ favors the exploration of the most probable belief states we will encounter in the future. This is good for two reasons. Firstly, if a belief state has a really low probability of occurring in the future, then we do not need a high precision on its value because better precision on this belief state would only have a small impact on the value of the actions in b^0, and consequently on our action choice in b^0. Secondly, exploring the most probable belief states also increases the chance that we will be able to reuse the computations done for this belief state in the future, which will improve our precision in the future. Finally $E(b^d)$, favors the exploration of actions that look promising. This behavior is desired for multiple reasons. Generally, we will only hesitate between a few actions for the best action choice. These actions will have the highest probability of being optimal, and by concentrating the search to these actions, we should be in a better position to decide which one is the best. If for some reason the promising actions were not optimal, then we should find it pretty quickly when we get better precisions on their upper bounds.

We can also compare how this heuristic differs from other heuristics that have been proposed to guide best-first-search in POMPDs. Satia & Lave actually proposed a similar heuristic, i.e. they suggested exploring at each iteration the k fringe

nodes that maximize the term $\gamma^{d(b)}\hat{\epsilon}(b)\prod_{i=0}^{d(b)-1}P(o^i|b^i,a^i)$. This term differs from our heuristic by the fact that they do not consider the term $P(a|b)$. We will actually see that this makes a big difference in terms of performance in practice. On the other hand, BI-POMDP always explore the fringe node, reached by a sequence of actions that maximizes the upper bound, that maximizes $\hat{\epsilon}(b)$, i.e. it is equivalent to choosing the fringe node that maximizes $\hat{\epsilon}(b)\prod_{i=0}^{d(b)-1}P(a^i|b^i)$ using equation 7 for $P(a|b)$. This heuristic does not take into account the probability that a certain belief state is going to be reached, or the discount factor that applies to the value of this belief state, such that it may explore belief nodes in the tree that do not have a lot of impact on the bounds in b_0. Again, our experiments will show that this affects the performance of the BI-POMDP approach.

3.2 Anytime Error Minimization Search

As mentioned before, the expected error $E(b)$ will be the term we will seek to minimize. This can be done efficiently in an anytime fashion by always exploring the fringe belief state that has the highest $E(b)$. Since this term includes the probability $P(o|b,a)$, we can possibly handle large observation space because generally in such an environment, only a few observations will have high probabilities and therefore, the search will be conducted only in the most probable part of the tree. Furthermore, the probability $P(a|b)$ in $E(b)$ implicitly does pruning of the non optimal actions and limits the search to the parts of the search tree where the actions have small probabilities of being optimal. A detailed description of our algorithm is presented in algorithm 1.

Algorithm 1 $AEMS$: Anytime Error Minimization Search

Function $AEMS(t)$
Static : G: an AND-OR graph representing the current search tree.
$t_0 \leftarrow$ CURRENTTIME()
while CURRENTTIME() $- t_0 \leq t$ **do**
  $b^* \leftarrow \arg\max_{b \in \text{FRINGE}(G)} E(b)$
  EXPAND(b^*)
  BACKTRACK(b^*)
end while
return BESTACT(ROOT(G)))

The AEMS algorithm takes the time allowed to search the tree in parameter and returns the best action to do in the current belief state. This current belief state is stored in the root node of the AND-OR graph G. The graph G is also kept in memory to resume the search at the fringe in the next time steps. After an action is executed in the environment, the graph G is updated such that our new belief state will be the root of G. This is simply done by setting the root to the node we reach by following the action-observation path from the old root node. The value $E(b)$ can be computed quickly since all information needed to compute its value can be stored in the belief state nodes when they are created or updated.

The EXPAND function simply does a one-step look-ahead, from the fringe belief state given in parameter, by constructing the action AND-nodes and the next belief state OR-nodes resulting from all possible action and observation combinations. It also computes the lower and upper bounds for all the next belief states using the lower bound L and upper bound U functions. For the actions, the lower and upper bounds are simply computed using equation 2 in which V^* is replaced by L and U respectively. Notice that if we reach a belief state that is already somewhere else in the tree, it will be duplicated, since our current algorithm does not handle cyclic graph structure. We could possibly try to use a technique proposed for AO* (LAO* algorithm [Hansen and Zilberstein, 2001]) to handle cycle, but we have not investigated this further and how it affects the heuristic value.

Once a node has been expanded, we need to backtrack its new upper and lower bounds in the tree, in order to update the probabilities that each action is optimal and reconsider our best action choices. This is done by the BACKTRACK function which recursively recomputes the bounds (using equation 2 in which V^* is replaced by L and U), the probabilities $P(a|b)$ and the best actions of each ancestor nodes that leads to the expanded node b^*. Notice that if the bounds of a certain ancestor node do not change, then we do not need to pursue the backtracking process because all the subsequent ancestor node bounds will remain the same.

4 Empirical results

We now present empirical results with our AEMS algorithm. We have tested our algorithm in the RockSample environment [Smith and Simmons, 2004] and a modified version of this environment, called FieldVisionRockSample (FVRS). FVRS is a new environment that we introduce to test our algorithm in an environment with a big observation space; its observation space size is exponential in the number of rocks, as presented below.

In each of these environments, we first computed a very approximate policy with PBVI by limiting its computation time and number of belief points to a very small value. Then we evaluated this policy empirically and we compared the improvement yielded by different online approaches using this policy as a lower bound on V^*. For the upper bound, we used the QMDP algorithm and solved the underlying MDP, and provided the resulting value function to every online algorithm. The online time available for decision making was constrained to 1 second per action and all the different online heuristics presented (Satia, BI-POMDP, AEMS1, AEMS2) were implemented in the same best-first-search algorithm, such that only the search heuristic could affect the performance and not the different implementations. We also compared our heuristic search to the performance of the depth-first search algorithm RTBSS, using the same PBVI and QMDP value functions for the lower and upper bounds.

4.1 RockSample

In RockSample (RS) environment, a robot has to explore the environment and sample good rocks. Each rock can be either good or bad (no scientific value) and the robot receives rewards accordingly. The robot also receives rewards by leaving the environment (by going to the extreme right of the environment). At the beginning, the agent knows the position

of each rock, but not their scientific value (good or bad). The robot has an noisy sensor to check if a rock is good or not before choosing to go to this rock and sample it. The accuracy of this sensor depends on the distance between the robot and the rock checked.

In Table 1, we compare the performance of the different algorithms in this environment. To get our results, we ran 20 runs on each of the possible starting rock states and the initial belief state for each of these runs was the uniform distribution over all rock states. We present the average reward, the average online time per action, the average error reduction[1] per time step, the average number of belief nodes in the search tree and the average percentage of belief nodes reused in the next time step. For PBVI, the time column shows the offline time used to compute its value function and the Belief Nodes column shows the number of α-vectors representing its value function. The number of states, actions and observations of each of the different RockSample environments are shown in parenthesis.

Table 1: Results in RockSample.

Heuristic / Algorithm	Reward	Time (s)	Error Reduction (%)	Belief Nodes	Nodes Reused (%)
RS(4,4) $(257s, 9a, 2o)$					
PBVI	7.7	136	-	65	-
Satia	7.7	0.990	43.45	19134	1.02
BI-POMDP	7.7	0.991	46.18	10547	4.93
RTBSS(4)	14.5	1.247	14.78	15267	0
AEMS1	**17.9**	**0.883**	**50.29**	**13692**	**24.71**
AEMS2	**17.9**	**0.846**	**64.05**	**14168**	**33.21**
RS(5,5) $(801s, 10a, 2o)$					
PBVI	5.4	104	-	57	-
Satia	17.3	0.947	24.52	11676	9.89
AEMS1	**17.5**	**0.942**	**35.19**	**11174**	**21.43**
RTBSS(3)	17.6	0.367	24.31	2373	0
AEMS2	**18.4**	**0.926**	**62.11**	**17500**	**36.36**
BI-POMDP	19.2	0.920	47.90	13626	37.40
RS(5,7) $(3201s, 12a, 2o)$					
PBVI	11.9	1345	-	114	-
Satia	17.6	0.963	10.68	3032	7.49
RTBSS(2)	21.4	0.172	22.89	318	0
AEMS1	**22.9**	**0.954**	**22.03**	**4857**	**21.74**
BI-POMDP	24.2	0.947	47.80	5749	34.12
AEMS2	**24.2**	**0.938**	**53.13**	**5577**	**38.03**
RS(7,8) $(12545s, 13a, 2o)$					
PBVI	7.7	2418	-	54	-
Satia	17.6	0.963	10.68	3032	7.49
BI-POMDP	19.0	0.974	39.68	2705	27.02
RTBSS(2)	19.4	1.029	23.10	448	0
AEMS1	**19.6**	**0.965**	**27.11**	**3094**	**27.9**
AEMS2	**21.3**	**0.950**	**50.32**	**3553**	**38.73**

The results obtained shows that our AEMS2 heuristic obtains a significantly better performance than the other heuristics, obtaining the highest error reduction percentage in all environments. It also obtains the best average reward in all environments except for RS(5,5) where BI-POMDP obtained surprisingly good rewards. On the other hand, AEMS1 obtains average performance in most environments. This may be due to the fact that equation 6 does not correctly approximate the probability that the action is optimal. Still, AEMS1 provided interesting performance compared to Satia, BI-POMDP and RTBSS in RS(4,4) and RS(7,8). We also observe that AEMS2 generally had a greater reuse percentage than all other approaches which also indicates that the heuristic is really guiding the search toward belief nodes that are more likely to be encountered in the future.

[1]We define the error reduction as $1 - \hat{\epsilon}(b_0)_{after}/\hat{\epsilon}(b_0)_{offline}$, where $\hat{\epsilon}(b_0)_{after}$ represent the difference between the bounds in the search tree after the search and $\hat{\epsilon}(b_0)_{offline}$ represent $U(b_0) - L(b_0)$ from the offline value functions computed offline.

4.2 FieldVisionRockSample

FieldVisionRockSample (FVRS) differs from the original RockSample by the fact that the robot's noisy sensor perceives all rocks at each turn. So basically, instead of having only 2 observations (Good or Bad), there are 2^R observations, where R is the number of rocks in the environment. As in RockSample, the probability that the sensor is accurate for a certain rock at distance d is computed according to $\eta/2 + 0.5$ where $\eta = 2^{-d/d_0}$, d_0 being the half efficiency distance, but since the robot perceives all rocks after each action, we had to decrease the accuracy of its sensor in order for the problem to remain partially observable. Thus, we have chosen to set the half efficiency distance d_0 of the sensor to $d_0 = (n-1)\sqrt{(2)}/4$, where n is the width of the square grid. The probabilities of the joint observations are obtained by simply taking the joint probability of each rock observations using the product rule (we consider that each rock observation is independant). The action space size is reduced to 5 (4 *Move* action and a *Sample* action) since the robot has no need to do *Check* actions.

In Table 2, we compare the same statistics that we presented for RockSample in the FVRS environments.

Table 2: Results in FieldVisionRockSample.

Heuristic / Algorithm	Reward	Time (s)	Error Reduction (%)	Belief Nodes	Nodes Reused (%)
FVRS(4,4) $(257s, 5a, 16o)$					
PBVI	10.1	158	-	52	-
BI-POMDP	11.3	0.942	12.21	17478	1.81
Satia	12.3	0.940	20.11	12800	1.75
AEMS1	**18.9**	**0.864**	**24.19**	**12388**	**3.22**
AEMS2	**20.0**	**0.857**	**30.83**	**13861**	**5.54**
RTBSS(2)	20.8	0.349	30.96	3271	0
FVRS(5,5) $(801s, 5a, 32o)$					
PBVI	9.2	534	-	28	-
RTBSS(2)	20.7	2.446	34.40	13996	0
BI-POMDP	21.7	0.914	19.00	11847	1.43
Satia	21.8	0.922	19.80	9725	1.79
AEMS1	**21.8**	**0.923**	**25.61**	**9799**	**1.94**
AEMS2	**22.1**	**0.917**	**26.74**	**10785**	**4.66**
FVRS(5,7) $(3201s, 5a, 128o)$					
PBVI	14.1	7802	-	31	-
RTBSS(1)	26.4	0.183	19.53	475	0
BI-POMDP	26.7	0.966	11.52	5487	0.25
Satia	26.8	1.015	12.06	5225	0.67
AEMS1	**27.0**	**0.977**	**12.67**	**5352**	**1.05**
AEMS2	**27.2**	**0.966**	**14.60**	**5667**	**2.20**

As in RS, we observe that our AEMS2 heuristic performs substantially better than all other heuristics in the FVRS environments. While RTBSS obtained the best result on FVRS(4,4), we see that it does not scale as well when the branching factor increases significantly, in FVRS(5,5) and FVRS(5,7), since it explores all action/observation combinations. FVRS has a much greater branching factor than

the standard RS and the fact that we succeeded in significantly improving the policy obtained with PBVI shows that our heuristic scales well even in environments with big observation space.

5 Conclusion

The online search algorithm we proposed has shown very good performance in large POMDPs and can tackle big observation space more efficiently than other previous online search techniques. It has shown the best overall performance both in terms of rewards and error reduction compared to the other existing online algorithm. While our search heuristic seems to work very well in practice, much more theoretical work needs to be done in order to bound the error yielded by our algorithm. We also want to investigate further improvement to the algorithm. As we mentioned, our current algorithm does not handle cycle which makes it do redundant computations when belief states are duplicated. We would also like to investigate further different variants of the term $P(a|b)$ and see whether we can come up with better approximations of the true probability that a certain action is optimal. Finally, we have only tried to combine our approach with the PBVI algorithm, since it is fairly easy to implement. However combining our algorithm with HSVI would be promising since HSVI computes quickly lower and upper bounds that we could use online with our algorithm.

References

[Braziunas and Boutilier, 2004] Darius Braziunas and Craig Boutilier. Stochastic Local Search for POMDP Controllers. In *Proc. of AAAI-04*, 2004.

[Cassandra *et al.*, 1997] A. Cassandra, M. L. Littman, and N. L. Zhang. Incremental Pruning: A Simple, Fast, Exact Method for Partially Observable Markov Decision Processes. In *Proc. of UAI-97*, pages 54–61, 1997.

[G. Shani and Shimony, 2005] R. Brafman G. Shani and S. Shimony. Adaptation for Changing Stochastic Environments through Online POMDP Policy Learning. In *Proc. of ECML-05*, 2005.

[Geffner and Bonet, 1998] H. Geffner and B. Bonet. Solving Large POMDPs Using Real Time Dynamic Programming, 1998. Working notes. Fall AAAI symp. on POMDPs.

[Hansen and Zilberstein, 2001] Eric A. Hansen and Shlomo Zilberstein. LAO * : A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1-2):35–62, 2001.

[Littman *et al.*, 1995] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning Policies for Partially Observable Environments: Scaling Up. In *Proc. of ICML-95*, 1995.

[Littman, 1996] M. L. Littman. *Algorithms for Sequential Decision Making*. PhD thesis, Brown University, 1996.

[Madani *et al.*, 1999] Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence. (AAAI-99)*, pages 541–548. The MIT Press, 1999.

[McAllester and Singh, 1999] David McAllester and Satinder Singh. Approximate Planning for Factored POMDPs using Belief State Simplification. In *Proc. of UAI-99*, pages 409–416, San Francisco, CA, 1999. Morgan Kaufmann Publishers.

[Nilsson, 1980] N.J. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishing, 1980.

[Papadimitriou and Tsitsiklis, 1987] Christos Papadimitriou and John N. Tsitsiklis. The Complexity of Markov Decision Processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[Paquet *et al.*, 2005] Sébastien Paquet, Ludovic Tobin, and Brahim Chaib-draa. An Online POMDP Algorithm for Complex Multiagent Environments. In *Proc. of AAMAS-05*, Utrecht, The Netherlands, 2005.

[Paquet *et al.*, 2006] Sébastien Paquet, Brahim Chaib-draa, and Stéphane Ross. Hybrid POMDP algorithms. In *Proceedings of The Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM-2006)*, Hakodate, Hokkaido, Japan, 2006.

[Pineau *et al.*, 2003] J. Pineau, G. Gordon, and S. Thrun. Point-based Value Iteration: An Anytime Algorithm for POMDPs. In *Proc. of IJCAI-03*, pages 1025–1032, Acapulco, Mexico, 2003.

[Pineau, 2004] Joelle Pineau. *Tractable Planning Under Uncertainty: Exploiting Structure*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 2004.

[Poupart, 2005] Pascal Poupart. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes*. PhD thesis, University of Toronto, 2005.

[Satia and Lave, 1973] J. K. Satia and R. E. Lave. Markovian decision processes with probabilistic observation of states. *Management Science*, 20(1):1–13, 1973.

[Smith and Simmons, 2004] Trey Smith and Reid Simmons. Heuristic Search Value Iteration for POMDPs. In *Proc. of UAI-04*, Banff, Canada, 2004.

[Smith and Simmons, 2005] Trey Smith and Reid Simmons. Point-based POMDP Algorithms: Improved Analysis and Implementation. In *Proc. of UAI-05*, Edinburgh, Scotland, 2005.

[Spaan and Vlassis, 2005] Matthijs T. J. Spaan and Nikos Vlassis. Perseus: Randomized Point-based Value Iteration for POMDPs. *J. of AI Research (JAIR)*, 24:195–220, 2005.

[Washington, 1997] Richard Washington. BI-POMDP: Bounded, Incremental Partially-Observable Markov-Model Planning. In *Proc. of the 4th Eur. Conf. on Planning*, volume 1348 of *Lecture Notes in Computer Science*, pages 440–451, Toulouse, France, 1997. Springer.