Hybrid POMDP Algorithms

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

Abstract

When an agent evolves in a partially observable environment, it has to deal with uncertainties when choosing its actions. An efficient model for such environments is to use partially observable Markov decision processes (POMDPs). Many algorithms have been developed for POMDPs. Some use an offline approach, learning a complete policy before the execution. Others use an online approach, constructing the policy online for the current belief state. In this article, we present three hybrid algorithms that have been developed to combine the strengths of these two extremes approaches (offline and online). We present results showing that hybrid algorithms can often obtained better results than the online or the offline algorithms alone.

1 Introduction

When faced with a partially observable environment, a general model for sequential decision problems is to use partially observable Markov decision processes (POMDPs). A lot of problems can be modeled with POMDPs, but very few can be solved because of their computational complexity (POMDPs are PSPACE-complete [10]). The main problem with POMDPs is that their complexity makes them applicable only on small environments. However, most problems of interest have a huge state space, which motivates the search for approximation methods [6]. This is especially the case for multiagent systems where there is often a huge state space with autonomous agents interacting with each other.

In the last few years, POMDPs have generated a lot of interest in the AI community and many approximation algorithms have been developed recently [2, 12, 14, 17, 18]. They all share in common the fact that they solve the problem offline. This means that they specify, prior to the execution, the action to execute for all possible situations the agent could encounter in the environment. This plan, linking an environment state with the chosen action, is called a *policy*.

On the other hand, there are other algorithms that have adopted an online approach [3, 8, 11, 19]. At each action's choice, these algorithms estimate the value of each possible actions to choose the most promising one. By doing an online search, they avoid the overwhelming complexity of computing a policy for every possible situation the agent could encounter. In fact, the policy is dynamically constructed for the belief states reached during the execution.

In addition, we try to take advantage of both extremes by proposing some hybrid approaches that use the RTBSS online search strategy [11] mixed with approximate offline strategies. We present three new algorithms: RTBSS-QMDP, RTBSS-PBVI-QMDP and RTDPBSS. The RTBSS-QMDP algorithm uses the Q_{MDP} [7] value function as the utility of the belief states at the leaves of the RTBSS search tree. The RTBSS-PBVI-QMDP algorithm uses the PBVI [12] value function as a lower bound at the leaves of the tree and the Q_{MDP} value function as an upper bound for the pruning function. Finally, the RTDPBSS algorithm is exactly like the RTDP-BEL [3] algorithm except that the RTDPBSS algorithm does a deeper search online when choosing the actions. The results of our experiments show that the performances of the hybrid approaches are often better than the performances of the online approach or the offline approach taken alone.

In the first part of this article, we present some POMDP algorithms. Then we describe the formalism of the RTBSS online algorithm and our hybrid algorithms, followed by some results on standard POMDPs.

2 POMDP Model

Partially Observable Markov Decision Processes (POMDPs) provide a general framework for acting in partially observable environments. A POMDP is a model for planning under uncertainty, which gives the agent the ability to effectively estimate the outcome of its actions even though it cannot exactly observe the environment.

One drawback of POMDP is that optimal POMDP models are generally quite useless, because of their complexity (PSPACE-complete [10]). They can only be applied to really small problems of only ten to twenty states. In consequence, many researchers have worked on improving the applicability of POMDP approaches by developing approximation approaches that can be applied to bigger problems [6].

2.1 Offline Approximation Algorithms

One popular approximation approach is to approximate the value function by updating it only for some selected belief states. The idea is that instead of planning over the complete belief space of the agent (which is intractable for large state spaces), planning is carried out only on a limited set of belief states that are chosen or sampled by using the POMDP model. Some of these methods choose the belief states at some given interval, randomly or at the position of the states of the underlying MDP [5, 7]. These methods are called fixed grid-based methods because their grid of chosen belief states over the belief state space stays constant. Some other methods prefer to use variable problem dependant sets of belief states and consequently, they can be seen as variable grid-based methods [1, 20]. This enables them to have more belief states to represent the value function in the most important regions of the belief state space.

One variation of the variable grid-based approach is the point-based approach, in which the belief states are sampled by starting in the initial belief state and by simulating some random interactions of the agent with the POMDP environment. By doing so, the belief states sampled have more chance of being reached during the execution of the agent. Another important aspect of this approach is that it does not just update the value at the sampled belief states, but also updates their gradient. This means that the value function is defined for all the belief state space and not just for the sampled belief states. Different algorithms have been developed using the point-based approach: PBVI [12], Perseus [18], HSVI [16] and HSVI2 [17]. These methods are distinguished by the approach they use for choosing belief states and by the method they use to update the value function at these chosen belief states.

Instead of learning a value function and then extracting a policy, some methods directly try to optimize the policy. Most methods in this case restrict the space of policies to policies that can be represented as finite state

controllers (FSCs) [2, 4, 15]. In general, policy iteration methods are quite interesting because they often converge rapidly. However, it might be hard to find the best policy representation for each problem. Also, gradient search methods can get trapped in local minima.

Another approach to improve the tractability of POMDP algorithms is to mixed them with other types of algorithms. For example, Nair and Tambe [9] have developed an hybrid BDI-POMDP approach, where BDI team plans are exploited to improve POMDP tractability and POMDP analysis improves BDI team plan performance. By using the synergy between a BDI algorithm and a distributed POMDP algorithm, these authors have shown that it is possible to solve more complex problems.

2.2 Online Approximation Algorithms

Usually, with offline approaches, the algorithm returns a complete policy defining which action to execute in every possible belief state. On the other hand, an online algorithm takes as input the current belief state and returns the single action that seems to be the best for this particular belief state. In this online view, the online POMDP algorithm is itself simply a policy, but one that may need to perform some non trivial computation at each belief state in order to return the best action.

In this vein, Washington developed the BI-POMDP algorithm [19], which expands an AND/OR tree to find an approximate policy for the POMDP. The BI-POMDP algorithm can be executed online, where to choose each action, it does a search for a given time. However, the BI-POMDP algorithm also needs some time offline to calculate the solution of the underlying MDP, which is used to calculate the bounds during the search.

Another algorithm is the RTDP-BEL algorithm which learns some heuristic values for the belief states visited by successive trials in the environment [3]. At each belief state visited, the agent evaluates all possible actions. The agent then executes the action that returned the greatest expected value. Afterwards, the heuristic value for the current belief state is updated with the value of the best action. Finally, the agent executes the chosen action and it makes the new observation, ending up in a new belief state. This approach is repeated until the goal is reached by the agent. In practice, this algorithm needs a lot of memory to store all the learned belief state values.

Another online algorithm is the one presented by [8], where these authors used an online exploration of the belief state space, similar to the one used

by the BI-POMDP algorithm. The difference is that they do not explore all observations for each possible action in a given belief state, they rather sample C observations from a generative model for each action.

In this article, we have adapted the RTBSS (Real-Time Belief State Search) algorithm, which is a completely online POMDP algorithm, which does a look-ahead search in the belief state space to find the best action to execute at each cycle in the environment [11]. The RTBSS algorithm only explores reachable belief states starting from the agent's current belief state. To accelerate the search, the RTBSS algorithm uses a factored POMDP representation and a branch and bound strategy. By pruning some branches of the search tree, the RTBSS algorithm is able to search deeper, while still respecting the real-time constraints.

In fact, in this article we try to combine the strengths of the online and offline POMDP algorithms. Shortly, in section 4, we present our own hybrid approaches, mixing offline and online POMDP algorithms. But first, we present in more details the RTBSS algorithm, which is at the heart of our hybrid algorithms.

3 RTBSS

The RTBSS algorithm estimates the value of a belief state by using a look-ahead search [11]. The main idea is to construct a tree where the nodes are belief states and where the branches are a combination of actions and observations (see Figure 1). The RTBSS algorithm uses a tree with estimated values at the fringe to estimate the value of the current belief state b_0. As the tree is expanded, the estimates become more accurate, which is guaranteed by the discount factor [19]. In other words, the tree estimates the value of the current belief state b_0, by exploring all reachable belief states for a given horizon.

Formally, the RTBSS algorithm uses a function that takes as parameters a belief state b and a depth d and returns an estimation of the value of b by performing a search of depth d. For the first call, d is initialized at D, the maximum depth allowed for the search.

$$\delta(b,d) = \begin{cases} U(b) & \text{, if } d = 0 \\ \max\limits_{a \in A} \left[\widehat{R}_B(b,a) + \gamma \sum\limits_{o \in \Omega} \left(\widehat{P}r(o \mid b,a) \times \delta(\widehat{\tau}(b,a,o), d-1) \right) \right] & \text{, if } d > 0 \end{cases} \tag{1}$$

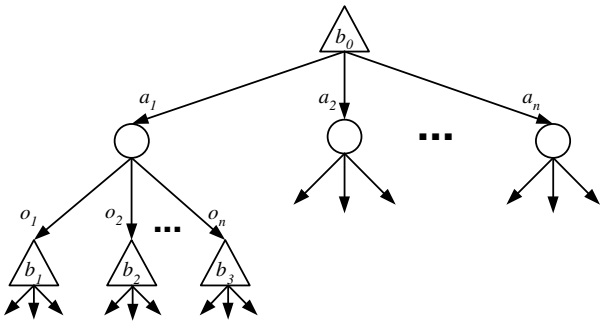

Figure 1: A search tree.

where $\widehat{R}_B(b, a)$ is the reward for executing the action a in the belief state b, $\widehat{P}r(o|b, a)$ is the probability of observing o, if the agent executes a in b and $\widehat{\tau}(b, a, o)$ is the belief update function which returns the resulting belief state.

When $d = 0$, we are at the bottom of the search tree. In this situation, the value of a belief state is given by a utility function $U(b)$. This function gives an idea of the real value of this belief state (if the function $U(b)$ was perfect, there would be no need for a search). This utility function has to be defined for each problem.

When $d > 0$, the value of a belief state at a depth of $D - d$ is simply the immediate reward for being in this belief state added to the maximum discounted reward of the subtrees underneath this belief state.

Finally, the agent's policy is dynamically calculated each time the agent has to choose an action. The action a to perform in a certain belief state b is simply the action that returns the best expected value evaluated with a search of depth D:

$$\pi(b, D) = \underset{a \in A}{\operatorname{argmax}} \left[\widehat{R}_B(b, a) + \gamma \sum_{o \in \Omega} \left(\widehat{P}r(o \mid b, a) \times \delta(\widehat{\tau}(b, a, o), D - 1) \right) \right] \tag{2}$$

It is important to notice that this equation is not used to define the policy for all possible belief states. This function is called only with the current agent's belief state when the agent has to choose an action online.

4 Hybrid Approaches

In this section, we present how we have combine the RTBSS online algorithm with three different offline algorithms: the Q_{MDP} algorithm [7], the PBVI algorithm [12] and the RTDP-BEL algorithm [3]. Using these three offline algorithms with the online RTBSS algorithm, we propose three new hybrid approaches that combine offline and online computation: RTBSS-QMDP, RTBSS-PBVI-QMDP and RTDPBSS.

4.1 RTBSS-QMDP

The RTBSS-QMDP approach uses the Q_{MDP} algorithm to compute an approximation of the value function offline. Then, RTBSS-QMDP algorithm uses the Q_{MDP} approximation as the leaf value function in the search tree of the RTBSS algorithm. In other words, the only modification to the RTBSS algorithm is that the $U(b)$ function at the leaves of the tree returns $V_{MDP}(b)$. Since we know that $V_{MDP}(b)$ introduces a certain error ϵ on the value of the belief state b, the advantage of using the RTBSS algorithm in combination with Q_{MDP} is to reduce this error. Since the RTBSS algorithm returns an exact value up to depth d, it will multiply the error ϵ by a factor of γ^d, where d is the depth of the search done by RTBSS.

The advantage of using the Q_{MDP} algorithm over the other offline algorithms is that it rapidly gives a good estimate of the belief state values, even in large state spaces. However, the inconvenient is that by using Q_{MDP} as the RTBSS leaf values, we cannot do any action pruning in the tree since Q_{MDP} is an upper bound on the exact value function V^*. We recall that the action pruning can only be done if the leaf value is a lower bound on the exact value function $V^*(b)$ and if the pruning HEURISTIC function returns an upper bound on the exact value $V^*(b)$. Nevertheless, the experiments show that only a search depth of 3 or 4 can give very good results and improve the results of both algorithms when used alone.

4.2 RTBSS-PBVI-QMDP

In order to reintroduce the action pruning into the RTBSS-QMDP algorithm, we propose the RTBSS-PBVI-QMDP algorithm, where PBVI is used to compute a lower bound on the exact value function $V^*(b)$ and Q_{MDP} is

used to compute an upper bound on the exact value function $V^*(b)$. Therefore, the only modification to the RTBSS algorithm is that the $U(b)$ function at the leaves of the tree returns $V_{PBVI}(b)$ and the pruning HEURISTIC function returns $V_{MDP}(b)$. What we want to show with this approach is that it might be possible to compute a quick policy with PBVI, then use this policy as a lower bound and use the Q_{MDP} algorithm to do some action pruning in the search tree of the RTBSS algorithm. By doing so, it might be possible to search deeper and thus reduce the error introduced by the approximate PBVI policy used at the leaves of the search tree.

4.3 RTDPBSS

The RTDPBSS algorithm is basically a combination of the RTDP-BEL algorithm and the RTBSS algorithm, without the action pruning. Instead of choosing an action based directly on the value of the approximation function, as done in RTDP-BEL, the RTDPBSS algorithm adds a depth search of depth d, where the leaf values are given by the approximate value function of the RTDP-BEL approach, but where parent nodes compute the exact values with the immediate rewards. This approach has the advantage of reducing the error of the approximate value function by a factor of γ^d.

The only inconvenient of this approach is that it necessitates more time online to choose actions than in RTDP-BEL, since we do a full depth search. Therefore, the RTDPBSS algorithm requires more time to run a certain number of simulations during the learning phase. However, the results show that the RTDPBSS algorithm tends to converge much more rapidly toward a very good policy than the RTDP-BEL algorithm.

In the next section, we present a proof that the hybrid approach can always reduce the error of an approximate offline approach.

4.4 Proof of the Usefulness of an Hybrid Approach

The intuition behind a hybrid approach is that for any approximate value function $V(b)$ with error bounds $[\inf_b \epsilon(b), \sup_b \epsilon(b)]$ (where $\epsilon(b) = |V(b) - V^*(b)|$) on the exact value function $V^*(b)$, we can always get a lower error bound by computing an exact value function up to a certain horizon d and use the approximation $V(b)$ at the last horizon. This should in fact multiply the error bound of $V(b)$ by a factor of γ^d, as demonstrated in the following proof.

Theorem 4.1. *For any algorithm that computes an approximate value function $V(b)$ with error function $\epsilon(b)$, defining the error on the exact value function $V^*(b)$ such that $\epsilon(b) = |V(b) - V^*(b)|$, the error bounds of the RTBSS algorithm doing a search of depth d with $U(b) = V(b)$ will have $\sup_b \epsilon_{RTBSS}(b) <= \gamma^d \sup_b \epsilon(b)$ and $\inf_b \epsilon_{RTBSS}(b) >= \gamma^d \inf_b \epsilon(b)$ on the exact value function $V^*(b)$.*

Proof. The following proof uses many terms, which are defined as:

- $P(b^d) = P(o^0|b^0, a^0_{\max})P(o^1|b^1, a^1_{\max}) \cdots P(o^{d-1}|b^{d-1}, a^{d-1}_{\max})$
- $a^i_{\max} = \text{argmax}_a R(b^i, a) + \gamma \sum_{o^i} P(o^i|b^i, a) V_{RTBSS(d-i-1)}(b^{i+1})$
- $b^d = \tau(b^{d-1}, a^{d-1}_{\max}, o^{d-1})$
- $s(b) = -1$ when $V(b) < V^*(b)$
- $s(b) = 1$ when $V(b) \geq V^*(b)$
- $\epsilon_{\max} = \sup_b \epsilon(b)$
- $\epsilon_{\min} = \inf_b \epsilon(b)$

Where, $P(b^d)$ represents the probability of reaching the leaf belief state b^d at search depth d. In the search tree, $P(b^d)$ corresponds to the product of the observations probability encountered by following the path from the root b to the leaf b^d.

Proof of Theorem 4.1: if $U(b) = V(b)$, the error of the RTBSS algorithm ($\epsilon_{RTBSS(d)}(b)$) is bounded by:

$$
\begin{aligned}
\epsilon_{RTBSS(d)}(b) &= |V_{RTBSS(d)}(b) - V^*(b)| && \text{Definition of the error} \\
&= |V^*_{d-1}(b) + \gamma^d \sum P(b^d) U(b^d) - V^*(b)| && \text{Definition of } V_{RTBSS(d)}(b) \\
&= |V^*_{d-1}(b) + \gamma^d \sum P(b^d) V(b^d) - V^*(b)| && \text{Definition of } U(b) \\
&= |V^*_{d-1}(b) + \gamma^d \sum P(b^d)(V^*(b^d) + \\
&\quad s(b^d)\epsilon(b^d)) - V^*(b)| && \text{Definition of } V(b) \\
&= |V^*_{d-1}(b) + \gamma^d \sum (P(b^d)V^*(b^d) + \\
&\quad P(b^d)s(b^d)\epsilon(b^d)) - V^*(b)| && \text{Distributivity} \\
&= |V^*_{d-1}(b) + \gamma^d \sum (P(b^d)V^*(b^d)) + \\
&\quad \gamma^d \sum (P(b^d)s(b^d)\epsilon(b^d)) - V^*(b)| && \text{Split sum} \\
&= |V^*(b) + \gamma^d \sum (P(b^d)s(b^d)\epsilon(b^d)) - V^*(b)| && \text{Definition of } V^*(b) \\
&= |\gamma^d \sum (P(b^d)s(b^d)\epsilon(b^d))| && \text{Simplification}
\end{aligned}
$$

In the worst case:

$$
\begin{array}{lcll}
|\gamma^d \sum(P(b^d)s(b^d)\epsilon(b^d))| & \leq & |\gamma^d \sum(P(b^d)s(b^d)\epsilon_{\max})| & \epsilon(b^d) \leq \epsilon_{\max} \\
& = & |\gamma^d \epsilon_{\max} \sum(P(b^d)s(b^d))| & \text{Simplification} \\
& \leq & |\gamma^d \epsilon_{\max}| & \sum(P(b^d)s(b^d)) \leq 1 \\
& = & \gamma^d \epsilon_{\max} &
\end{array}
$$

In the best case:

$$
\begin{array}{lcll}
|\gamma^d \sum(P(b^d)s(b^d)\epsilon(b^d))| & \geq & |\gamma^d \sum(P(b^d)s(b^d)\epsilon_{\min})| & \epsilon(b^d) \geq \epsilon_{\min} \\
& = & |\gamma^d \epsilon_{\min} \sum(P(b^d)s(b^d))| & \text{Simplification} \\
& \geq & |-\gamma^d \epsilon_{\min}| & \sum(P(b^d)s(b^d)) \geq -1 \\
& = & \gamma^d \epsilon_{\min} &
\end{array}
$$

Consequently, $\epsilon_{RTBSS(d)}(b) \in [\gamma^d \epsilon_{\min}, \gamma^d \epsilon_{\max}]$ □

This theorem has a pretty strong implication. It means that for any offline algorithm computing an approximate value function with error bounds $[\epsilon_{\min}, \epsilon_{\max}]$, we can always get a better approximation by combining this offline algorithm with our RTBSS algorithm online. In the next section, we present results for the hybrid approaches that confirm this claim.

5 Experimentations

In this section, we present the results of the hybrid algorithms on two problems found in the POMDP literature: *Tag* [12] (Figure 2) and *Rock-Sample* [16] (Figure 3).

5.1 *Tag*

We tested our algorithm in *Tag*, introduced for the first time by [12]. This environment has also been used recently in [2, 13, 15, 16, 17, 18]. The *Tag* environment consists in an agent A that has to catch another agent B. The agent A cannot see the agent B, except if both agents are at the same position. To find B, the agent A has to move in the environment where it thinks B should be. If it finds B, it can tag it and its goal has been achieved.

In Table 1, we can see that the RTBSS-PBVI-QMDP hybrid algorithm obtains good results. For our tests, we generated one policy for the PBVI algorithm in 10 hours for the starting position 0. This policy obtained an average reward of -10.61. On the other hand, the hybrid approach obtained an average reward of -5.40. This shows that the hybrid approach is quite efficient to improve a weak offline policy. Another example of that is the

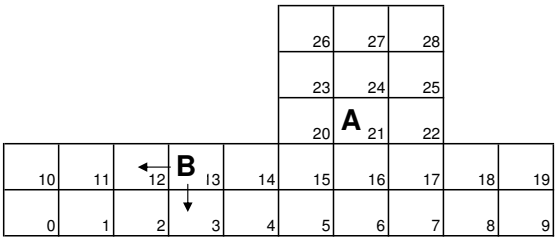

Figure 2: The *Tag* problem.

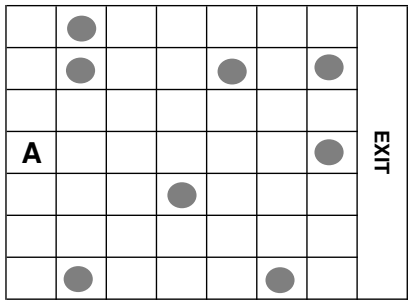

Figure 3: *RockSample[7,8]*.

RTBSS-QMDP approach that obtained a much better result (-6.11) than the Q_{MDP} approach alone (-16,75).

However, the RTDP-BEL and the RTDPBSS algorithms are not good on Tag. They were not able to converge, because they were not guided enough. The reward can be quite far in Tag, and they had some trouble finding the other agent. These methods act almost randomly at the beginning, thus they were not able to catch the agent B often enough to learn how to catch it. These methods need a lot of memory to store all the belief states encountered, thus they often ran out of memory, even with one Gigabyte of memory.

5.2 *RockSample*

We have also tested our hybrid algorithms in a bigger environment, the *RockSample* problem which was presented for the first time by [16]. In the *RockSample* problem, an agent has to explore the environment and sample some rocks. The agent receives rewards by sampling rocks and by leaving the environment (by going to the extreme right of the environment). A rock can have a scientific value or not and the agent has to sample only good rocks. At the beginning, the agent knows the position of each rock, but not their scientific value. In order to verify if a rock is good, the agent can use an imperfect sensor. This sensor helps the agent to see if a rock is good or not before choosing to go to this rock and sample it.

If we look at the results obtained by our hybrid algorithms, we can see that the RTBSS-QMDP algorithm got significantly better results than Q_{MDP}. This shows that it is possible to significantly improve a bad offline policy with a search only. In fact, if we look at all our results, the RTBSS-QMDP

Method	Reward	Offline Time (s)	Online Time (s)
Q_{MDP}	-16,75	0.875	-
RTDP-BEL	-12.15	3645	0,001
RTDPBSS	**-9.60**	**24540**	**0,556**
PBVI [12]	-9,18	180880	-
BBSLS [2]	$\backsim$ -8.3	$\backsim$ 100000	-
BPI [14]	-6,65	250	-
HSVI1 [16]	-6,37	10113	-
HSVI2 [17]	-6,36	24	-
PERSEUS [18]	-6,17	1670	-
PBVI [13]	$\backsim$ -6,12	$\backsim$ 900000	-
RTBSS-QMDP(5)	**-6.11**	**0.875**	**0,311**
RTBSS-PBVI-QMDP(4)	**-5.40***	**36000**	**0,220**
RTBSS(12)	**-5.03**	**0**	**0,750**

*: For this result, the agent A always starts in position 0.

Table 1: Comparison of our approach on the *Tag* problem. For RTBSS, the reward and the computation time are averages over 5000 simulations. The number between parenthesis besides the RTBSS based methods is the depth of the search D used to obtain these results.

approach is the one that obtained the more constant performances. It offers a good balance between a fast offline approximation and a search online to improve this approximation for the belief state reached.

Generally, our RTDPBSS algorithm showed quite an improvement over the RTDP-BEL algorithm. The RTDPBSS algorithm obtained good results on the three smallest problems. However, on biggest problems, the RTDPBSS showed its limitations, mainly due to the amount of memory necessary to store all the belief state value estimations.

In our tests, the RTBSS-PBVI-QMDP algorithm showed less effective behaviors. The time needed online to calculate two complex bounds with the PBVI and the QMDP algorithms was just too important. However, we can see that the hybrid algorithms still generally obtains better results than the offline algorithms separately.

Problem	Algorithm	Reward	Offline Time(s)	Online Time(s)
RockSample[4,4] (257s,9a,2o)	Q_{MDP}	8.6	0.188	0
	Perseus	16.1	2104	0
	RTBSS(4)	**16.5**	**0**	**0.001**
	RTBSS-PBVI-QMDP(2)	**17.4**	**8229**	**0.014**
	PBVI	17.9	10200	0
	RTDP-BEL	18.0	486	0.002
	RTBSS-QMDP(5)	**18.0**	**0.188**	**0.036**
	HSVI [16]	18.0	577	-
	HSVI2 [17]	18.0	0.75	-
	RTDPBSS(2)	**18.1**	**1272**	**0.022**
RockSample[5,5] (801s,10a,2o)	RTDP-BEL	7.3	1444	0.003
	Q_{MDP}	13.9	0.625	0.002
	Perseus	14.0	36000	0
	RTBSS(6)	**18.5**	**0**	**0.131**
	HSVI [16]	19.0	10208	-
	PBVI	19.1	36000	0
	RTBSS-PBVI-QMDP(2)	**19.2**	**36000**	**0.020**
	RTBSS-QMDP(4)	**19.3**	**0.625**	**0.627**
	RTDPBSS(2)	**19.4**	**18636**	**0.054**
RockSample[5,7] (3201s,12a,2o)	Perseus	12.5	36000	0.001
	Q_{MDP}	17.3	4	0.015
	PBVI	18.9	36000	0
	RTBSS-PBVI-QMDP(2)	**20.3**	**36000**	**0.177**
	RTBSS(5)	**22.7**	**0**	**0.070**
	HSVI [16]	23.1	10263	-
	RTBSS-QMDP(2)	**23.7**	**4**	**0.104**
	RTDPBSS(2)	**24.5**	**41117**	**0.234**
	RTDP-BEL	24.7	8773	0.006
RockSample[7,8] (12545s,13a,2o)	PBVI	4.3	36000	0
	Perseus	8.3	36000	0.001
	RTDP-BEL	8.7	8362	0.029
	RTBSS-PBVI-QMDP(2)	**13.1**	**36000**	**0.559**
	HSVI [16]	15.1	10266	-
	Q_{MDP}	15.5	24	0.048
	RTDPBSS(2)	**17.2**	**47007**	**0.356**
	RTBSS(5)	**19.0**	**0**	**0.022**
	RTBSS-QMDP(2)	**20.3**	**24**	**0.320**
	HSVI2 [17]	20.6	1003	-
RockSample[10,10] (102401s,19a,2o)	Q_{MDP}	11.2	208	0.029
	RTBSS-QMDP(2)	**19.2**	**208**	**1.234**
	RTBSS(7)	**20.0**	**0**	**1.441**
	HSVI2 [17]	20.4	10014	-

Table 2: Comparison of many algorithms in different RockSample environments. The number between parenthesis beside the RTBSS based methods is the depth of the search D used to obtain those results.

6 Conclusion

In this article, we have presented some hybrid approaches that use the RTBSS online search strategy mixed with approximate offline strategies. We presented three new algorithms: RTBSS-QMDP, RTBSS-PBVI-QMDP and RTDPBSS. The RTBSS-QMDP algorithm uses the Q_{MDP} value function as the utility of the belief states at the leaves of the RTBSS search tree. The RTBSS-PBVI-QMDP algorithm uses the PBVI value function as a lower bound at the leaves of the tree and the Q_{MDP} value function as an upper bound for the pruning function. Finally, the RTDPBSS algorithm is exactly like the RTDP-BEL algorithm except that the RTDPBSS algorithm does a deeper search online when choosing the actions. The results of our experiments showed that the performances of the hybrid approaches are often better than the performances of the online approach or the offline approach taken alone. Moreover, our results have shown that RTBSS-QMDP is the most consistent approach over all the test environments.

Future work could be to look at other hybrid approaches. In our experiments, the hybrid approaches have often shown some improvements compared to completely online or offline approaches. Therefore, we think that the hybrid approaches have the most potential to deal with complex POMDP environments.

References

[1] B. Bonet. An Epsilon-Optimal Grid-Based Algorithm for Partially Observable Markov Decision Processes. In *Proceedings of The Nineteenth International Conference on Machine Learning (ICML-2002)*, pages 51–58, 2002.

[2] Darius Braziunas and Craig Boutilier. Stochastic Local Search for POMDP Controllers. In *The Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, 2004.

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

[4] Eric A. Hansen. An Improved Policy Iteration Algorithm for Partially Observable MDPs. In *Tenth Neural Information Processing Systems Conference (NIPS-97)*, Denver, Colorado, 1997.

[5] Milos Hauskrecht. Incremental Methods for Computing Bounds in Partially Observable Markov Decision Processes. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 734–739, 1997.

[6] Milos Hauskrecht. Value-Function Approximations for Partially Observable Markov Decision Processes. *Journal of Artificial Intelligence Research*, 13:33–94, 2000.

[7] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning Policies for Partially Observable Environments: Scaling Up. In *Proceedings of the 12th International Conference on Machine Learning (ICML-95)*, 1995.

[8] David McAllester and Satinder Singh. Approximate Planning for Factored POMDPs using Belief State Simplification. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 409–416, San Francisco, CA, 1999. Morgan Kaufmann Publishers.

[9] Ranjit Nair and Milind Tambe. Hybrid BDI-POMDP Framework for Multiagent Teaming. *Journal of Artificial Intelligence Research (JAIR)*, 23:367–420, 2005.

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

[11] Sébastien Paquet, Ludovic Tobin, and Brahim Chaib-draa. An Online POMDP Algorithm for Complex Multiagent Environments. In *Proceedings of The fourth International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS-05)*, Utrecht, The Netherlands, 2005.

[12] J. Pineau, G. Gordon, and S. Thrun. Point-based Value Iteration: An Anytime Algorithm for POMDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 1025–1032, Acapulco, Mexico, 2003.

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

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

[15] Pascal Poupart and Craig Boutilier. Bounded Finite State Controllers. In *Advances in Neural Information Processing Systems 16 (NIPS-2003)*, Vancouver, Canada, 2003.

[16] Trey Smith and Reid Simmons. Heuristic Search Value Iteration for POMDPs. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence(UAI-04)*, Banff, Canada, 2004.

[17] Trey Smith and Reid Simmons. Point-based POMDP Algorithms: Improved Analysis and Implementation. In *Proceedings of the 21th Conference on Uncertainty in Artificial Intelligence(UAI-05)*, Edinburgh, Scotland, 2005.

[18] Matthijs T. J. Spaan and Nikos Vlassis. Perseus: Randomized Point-based Value Iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

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

[20] Rong Zhou and Eric A. Hansen. An Improved Grid-Based Approximation Algorithm for POMDPs. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001)*, pages 707–716, 2001.