Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

A Fast Pairwise Heuristic for Planning under Uncertainty

Koosha Khalvati and **Alan K. Mackworth**
{kooshakh, mack}@cs.ubc.ca
Department of Computer Science
University of British Columbia
Vancouver, B.C. V6T 1Z4 Canada

Abstract

POMDP (Partially Observable Markov Decision Process) is a mathematical framework that models planning under uncertainty. Solving a POMDP is an intractable problem and even the state of the art POMDP solvers are too computationally expensive for large domains. This is a major bottleneck. In this paper, we propose a new heuristic, called the pairwise heuristic, that can be used in a one-step greedy strategy to find a near optimal solution for POMDP problems very quickly. This approach is a good candidate for large problems where real-time solution is a necessity but exact optimality of the solution is not vital. The pairwise heuristic uses the optimal solutions for pairs of states. For each pair of states in the POMDP, we find the optimal sequence of actions to resolve the uncertainty and to maximize the reward, given that the agent is uncertain about which state of the pair it is in. Then we use these sequences as a heuristic and find the optimal action in each step of the greedy strategy using this heuristic. We have tested our method on the available large classical test benchmarks in various domains. The resulting total reward is close to, if not greater than, the total reward obtained by other state of the art POMDP solvers, while the time required to find the solution is always much less.

Introduction

Planning under uncertainty has numerous applications in areas such as target tracking, robot navigation and assistive technology (Du et al. 2010; Viswanathan et al. 2011). POMDP is a mathematical framework for planning under uncertainty. There has been significant progress in solving POMDPs more efficiently in recent years. However, even the state of the art POMDP solvers are still too computationally expensive for large problems. The solution time is an important factor and in many practical applications only real-time approaches are acceptable. For example, in a target tracking problem, if the robot does not respond quickly enough, the target would soon go out of its visual range. On the other hand, in many problems, a near-optimal solution is good enough. In the target tracking problem, catching the target quickly and minimizing power consumption are often the two most important criteria. However, as long as the robot can catch the target, the exact optimality of the time required and power usage may not be crucial. As another example, in a robot navigation problem, as long as the robot does not hurt itself or others or wander aimlessly around the room, its path may be acceptable. In general, in many practical applications, the time required for decision making is more important than the exact optimality of the solution. In these situations, a POMDP solver that can find the solution very quickly is a good candidate even if the solution is not perfectly optimal. In this paper we propose a fast online approach for solving POMDPs. Our method achieves total reward close to or better than the total reward gained by previously state of the art POMDP solvers in much less time. This approach is a one step greedy strategy that uses a pairwise heuristic.

The pairwise heuristic is an approximation of the optimal plan for pairs of states. For each pair of states, we calculate the sequence of actions that resolves the uncertainty and gain the maximum reward, if the agent is uncertain about which one of the two states it is in. This calculation is done by running the value iteration method on an MDP (Markov Decision Process) whose states are pairs of states of the original problem. The whole process is independent of the initial belief state and so need only be done once for each domain. After obtaining the pairwise heuristic values, we use an online greedy strategy using those values to choose the optimal action at each step.

We have tested our method on large classical POMDP problems in various domains: robot navigation, target tracking and scientific exploration. The results are very promising. The resultant solution is near optimal while the time required is extremely low. This is the first time that an online algorithm gains near optimal reward with only one step look ahead search. Other online approaches need to go deep in the search tree to get a good total reward and that increases their computational cost (Ross et al. 2008). In addition to computational efficiency, unlike most of the POMDP solvers, the pairwise heuristic is simple to understand and implement.

Previous Work

There have been many results on planning under uncertainty in the last two decades. QMDP is one of the early methods for solving POMDP (Littman, Cassandra, and Kaelbling 1995). It tries to find the optimal strategy assuming that the

uncertainty is eliminated in the next step. This assumption, however, is not realistic especially in large problems. Therefore, QMDP does not usually work well. Cassandra et al. studied planning under uncertainty in the domain of robot navigation (Cassandra, Kaelbling, and Kurien 1996). They proposed a strategy, named entropy-weighting (EW), that carries out localization and reward maximization simultaneously. The problem with this strategy is that it only considers single step actions. Localization, however, is sometimes not possible with only one action. One of the successful early methods is the grid based approach (Brafman 1997; Poon 2001).

The point-based value iteration method is probably the most famous approach in the POMDP literature (Pineau, Gordon, and Thrun 2006). HSVI (Smith and Simmons 2004) and SARSOP (Kurniawati, Hsu, and Lee 2008) are also algorithms that are based on point-based value iteration. These methods produce significantly better results in comparison with earlier methods. However, they are computationally expensive and not suitable for large problems. This problem of scalability led some researchers to consider reducing the size of the problem by abstracting away some details and solving the reduced problem with a point-based solver. Using PCA to reduce the size of the state space and then solving the resulted POMDP is one of these methods (Roy and Gordon 2003). A group of researchers did this reduction in state space with variable resolution decomposition (Kaplow, Atrash, and Pineau 2010). Furthermore, two proposed methods ignore some observations and solve the POMDP by macro actions (He, Brunskill, and Roy 2010; Kurniawati et al. 2011).

Point-based approaches are all offline, in that all the planning is done before execution of the strategy. Beside offline strategies, some researchers have worked on online approaches where planning and execution are interleaved. As online methods focus only on the current belief state, they scale better than offline approaches. However, the search tree in these methods has to be expanded enough to produce a good solution. This is problematic in domains with large action and observation spaces. There is a comprehensive survey on online POMDP in the literature (Ross et al. 2008).

Among the approaches mentioned above the work of Cassandra et al. is the work most relevant to ours (Cassandra, Kaelbling, and Kurien 1996). Like this work, we use a single step greedy strategy with a heuristic that considers both revealing uncertainty and reward maximization. However, our heuristic can find the paths that reveal the uncertainty with more than one step. Also, the work of He et al. is very relevant to ours as they also focus on revealing uncertainty and reward maximization simultaneously (He, Brunskill, and Roy 2010). Although our method is online, it cannot be placed among other online approaches in the literature as the focus of those approaches is on the online search tree, including ways to searching deeper and pruning the tree, but ours is a simple one step greedy search. The main component of our method is the heuristic, not the search.

This is the first time that pairs of states are the focus of a method for solving POMDP. However, pairs have been exploited in active learning (Golovin, Krause, and Ray 2010) and robot localization (Khalvati and Mackworth 2012) before. Some components of our heuristic are inspired by the active learning paper (Golovin, Krause, and Ray 2010).

Problem definition

In the planning under uncertainty problem, an agent wants to get the maximum total reward by performing a sequence of actions, while being uncertain about its state. This problem is modeled by the POMDP framework. Formally, a POMDP is represented as a tuple $(S, A, O, T, Z, b_0, R, \gamma)$. S is the finite set of states, A is the finite set of possible actions, and O is the finite set of possible observations. $T : S \times A \times S \rightarrow [0, 1]$ is the transition function, defining $p(s'|s, a)$ for all $s, s' \in S$ and $a \in A$. $Z : S \times A \times O \rightarrow [0, 1]$ specifies the probability of each observation after entering into a state by an action, $p(o|s, a)$ for all $o \in O$, $s \in S$ and $a \in A$ (note that s is the posterior state). b_0, $S \rightarrow [0, 1]$, is the initial belief state, the probability distribution over possible initial states.

$$T(s, a, s') = p(s_{k+1} = s'|s_k = s, a_k = a) \quad (1)$$

$$Z(s, a, o) = p(o_k = o|s_{k+1} = s, a_k = a) \quad (2)$$

$$b_0(s) = p(s_0 = s) \quad (3)$$

The system is Markovian: the belief state in each step depends only on the previous belief state and the most recent action and observation.

$$b_{k+1}(s) \propto Z(s, a_k, o_k) \sum_{s'} T(s', a_k, s) b_k(s') \quad (4)$$

$R : S \times A \rightarrow R$ specifies the reward for performing each action in each state. Finally, γ is the discount factor. The goal is to choose a sequence of actions to maximize the expected total reward, $E[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$.

Solving a POMDP with the Pairwise Heuristic

Our approach to solving a POMDP is a one step greedy search using the pairwise heuristic. In this section, we first explain the intuition behind the pairwise heuristic. Then, the pairwise heuristic is explained in detail. Finally, we explain the greedy strategy that uses the heuristic.

Intuition

In an MDP, the only goal is maximizing the reward. But when we deal with uncertainty, information gathering and ambiguity resolution must be considered as well. Actually, a POMDP solver implicitly does information gathering and reward maximization simultaneously. In most problems, a good policy not only obtains rewards but also tries to decrease uncertainty. This does not mean decreasing uncertainty is a separate task. In fact, in most problems this decrease helps the agent to get higher reward in the future. As a result, we can say that uncertainty decreases in general while performing a good policy. In one of the steps of the policy,

uncertainty gets low enough that we can say we approximately know the current state of the agent. This is similar to *robot localization*. In robotics, localization means finding the robot's current state (Thrun, Burgard, and Fox 2005), where the state is the robot's position. Similarly, in our case localization means finding the actual current state. But this time, the state is more general than the agent's position. At this point we can say that the agent is localized and we can treat the problem as an MDP after this point. So our whole problem can be seen as maximizing the total reward in the localization phase plus the reward of the resultant MDP after localization.

Let us explain this intuition more with a simple example. Consider the map shown in Fig. 1. A robot wants to go to the goal state, cell G, knowing that it is in cell A or cell B with equal probability at first. The actions are deterministic. The observation in all of the states except cells C, D, E and F is 0. In cells C and E the observation is 10 and in cells D and F the robot observes 20. How can the robot reach the goal state in the minimum number of steps? If there were no uncertainty and the robot knew it was in A or knew it was in B, it could reach the goal in only 4 steps. But with uncertainty, going left or right doesn't help the robot. It should determine its exact position while traveling to the goal state. If it goes down for 3 steps it would then know its exact position and reach the goal in 3 steps after that. So it would reach the goal in 6 steps. Alternatively, it could go up for two steps and find itself. After those 2 steps, the robot could reach the goal in 6 steps. So this policy takes 8 steps and is worse than the previous one, even though it removes the uncertainty sooner. In this robot navigation problem the solution contains two phases, a path before localization and a path after that. In the optimal strategy, the aggregate of those two paths should be minimal. If we use rewards instead of path costs this cost minimization is exactly the same as reward maximization. In fact, we used path cost in this example because it is more intuitive for the reader.

We changed the POMDP problem to something that seems easier to solve, but even this problem is very difficult if the agent is uncertain about being in many states. But if the uncertainty is limited to only two states, the problem is not that hard. We can solve the problem for every pair of states and then use these solutions to solve the main problem.

The Pairwise Heuristic

As explained above, the heuristic needs an optimal sequence of actions for each pair. We call the reward of this optimal sequence the value function of the pair. To explain how to find these sequences, we assume that we only want to do the localization task for each pair at first. But how can we find an optimal sequence for localization for each pair? Actually, some pairs do not need any action for localization. These pairs are distinguishable. For example, for the pair (s, s'), if all possible observations in s have zero probability in s' and vice versa, there is no need for localization. For other states, we can carry out the localization by going to distinguishable pairs. After the localization, the current state is determined and to fulfill the reward maximization goal we just need to solve the problem in the MDP framework. One could argue that as the states are partially observable and the actions are not deterministic, the uncertainty about the state may arise again. In this situation we do the localization task again. This task is further explained in the next sections of the paper.

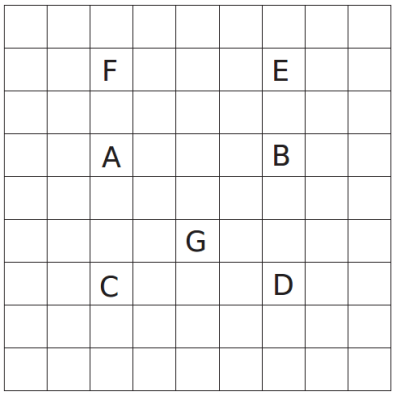

Figure 1: The robot wants to go to cell G while being uncertain about its initial position: cell A or cell B. The localization information is in cells C, D, E and F.

As the observation depends on both the state and the action that leads to that state, instead of finding distinguishable states, we find states distinguishable by an action. Two states are distinguishable by an action if, after performing that action, there is a high probability that different observations are recorded in the two states. Formally, s and s' are distinguishable by action a if and only if:

$$o^* = argmax_o p(o|s'', a)$$
$$o'^* = argmax_o p(o|s''', a)$$

$$\sum_{s'',s'''} p(s''|s)p(s'''|s')[p(o^*|s'', a)(1 - p(o^*|s''', a))+ p(o'^*|s''', a)(1 - p(o'^*|s'', a)] \geq 2\lambda \quad (5)$$

λ is a constant that is specified by a domain expert. If it is 1, the observations should be completely different. But, as in the localization problem where a robot is usually considered localized if the probability of a state is more than a threshold like 0.95, we can set this threshold to a value less than 1 in noisy environments. As shown in the formula the observations that are considered are the most probable observations of the posterior states. If there is more than one observation with maximum probability, one would be chosen arbitrarily.

The value function of this distinguishable pair is set to:

$$V(s, s') = 0.5[R(s, a)+R(s', a)+\gamma\times(V(s)+V(s'))] \quad (6)$$

$V(s)$ and $V(s')$ are the value functions of s and s' in the underlying MDP. Also, $u(s, s')$, the optimal action of s and s' is set to a.

To find the value function and optimal action for indistinguishable pairs, we use a value iteration algorithm in an

Algorithm 1: Finding the value functions and optimal actions for the pairs

Data: $(S, A, O, T, Z, R, \gamma)$
Result: $V(s, s')$ and $u(s, s')$ for all pairs
Calculate value functions, $V(S)$ of $MDP(S, A, T, R, \gamma)$
foreach *pair* (s, s') **do** $V(s, s') = R_{min}$
foreach *pair* (s, s') *and action* a **do**
  $R((s, s'), a) = 0.5[R(s, a) + R(s', a)]$
foreach *pair* (s, s') *and* (s'', s''') *and action* a **do**
  $p((s'', s''')|(s, s'), a) = 0$
  $s^* = argmax_{\hat{s}} p(\hat{s}|s, a)$
  $s'^* = argmax_{\hat{s}} p(\hat{s}|s', a)$
  $p((s^*, s'^*)|(s, s'), a) = 1$
foreach *pair* (s, s') *distinguishable by action* a **do**
  $V(s, s') = 0.5[R(s, a)+R(s', a)+\gamma(V(s)+V(s'))]$
  $u(s, s') = a$
repeat
  foreach *indistinguishable pair* (s, s') **do**
    $V_k(s, s') = max_a[R((s, s'), a) + \gamma \sum_{s'', s'''} V(s'', s''')p((s'', s''')|(s, s'), a)$
    $u(s, s') = argmax_a[R((s, s'), a) + \gamma \sum_{s'', s'''} V(s'', s''')p((s'', s''')|(s, s'), a)$
until *convergence*

MDP where the states are pairs of states of our original problem. The transition function is determined as follows:

$$s^* = argmax_{s''} p(s''|s, a)$$
$$s'^* = argmax_{s''} p(s''|s', a)$$

$$p((s^*, s'^*)|(s, s'), a) = 1 \tag{7}$$

$$\forall s'', s''' : s'' \neq s^* \vee s''' \neq s'^* : p((s'', s''')|(s, s'), a) = 0 \tag{8}$$

The equations above show that we ignore the noise of actions in the new MDP and consider only the most probable posterior states. Again, if there is more than one most probable state, one would be chosen arbitrarily. Also the reward, $R((s, s'), a)$ is equal to $0.5[R(s, a) + R(s', a)]$ where $R(s, a)$ and $R(s', a)$ belong to the original problem. We run the value iteration only for indistinguishable pairs. The initial value function for these pairs is set to the minimum reward in the main problem. Actions are the same as actions in the original problem. In addition, the discount factor of the new MDP is the same as the discount factor of the original problem. This algorithm is shown as Algorithm 1.

As shown above, we use the value of 0.5 in all of the equations for value functions which gives the unweighted average. This means we assume equal probability for the two states in computing their optimal action and value function. The states may not have equal probability in the original problem, but the pairwise value function is used as a heuristic and does not need to be exact. By using this simplification, obtaining value functions and optimal actions does not depend on the initial belief state and would be an offline calculation. So for each domain, we only need to run this algorithm once.

Algorithm 2: Choosing the optimal action in step k

Data: $(S, A, O, T, Z, b_k, R, \gamma)$, compare ratio
Result: Optimal action, a_k^*
maxBel = $max_s b_k(s)$
$S' = \{s|b_k(s) \geq \text{maxBel/compare ratio}\}$
$A' = \{a|a = u(s, s')\ s, s' \in S'\}$
if $|S'| = 1$ **then**
  a_k^* = optimal action of S' in the MDP
else
  foreach $a \in A'$ **do**
    foreach $s \in S'$ **do** $s^* = argmax_{s'} p(s'|s, a)$
    $H(a) = \sum_{s,s'}[(0.5(R(s, a) + R(s', a)) + \gamma V(s^*, s'^*))b_k(s)b_k(s')]$
  $a_k^* = argmax_{a \in A'} H(a)$

The greedy strategy

To solve the POMDP, we only need a one step greedy strategy that uses the value functions of the pairs. In each step, the selected action should maximize the expected total value function of the pairs. However, the expected instant reward of the actions should be considered as well. We ignore the noise of actions in the greedy strategy. As a result the selected action is:

$$s^* = argmax_{s''} p(s''|s, a)$$
$$s'^* = argmax_{s''} p(s''|s', a)$$

$$a_k^* = argmax_a \sum_{s,s'}[(0.5(R(s, a) + R(s', a)) + \gamma V(s^*, s'^*))b_k(s)b_k(s')] \tag{9}$$

We should note that maximization is not done over all possible actions and the selected action should be the optimal action for at least one pair of states. Also, one may argue that using value functions of the pairs is a kind of localization strategy (to be precise both localization and reward maximization) and the algorithm may get stuck in localization and never collect rewards. This is, in fact, true so to resolve this issue only the states with probability of more than a specified threshold are considered in the heuristic. This threshold is relative and is equal to the probability of the most likely state divided by a constant greater or equal to 1 which is specified by the domain expert. This constant is called the *compare ratio*. If in one of the steps of the planning the probabilities of all states, except the most likely one, become less than the threshold, the selected action would be the optimal action of the underlying MDP for that most likely state in that step. Obviously, as compare ratio is greater than or equal to 1, the probability of the most likely state is always above threshold. The whole strategy is shown as Algorithm 2.

Experiments

We tested our algorithm on several classical test benchmarks in the POMDP literature in three different domains: robot navigation, target tracking and scientific sampling. Three of these bench-marks are large problems, hence considered to be our main tests. The computational efficiency of our method is best reflected in large problems. These tests are *Fourth* ($|S| = 1052, |A| = 4, |O| = 28$, *max reward*$= 1$) (Cassandra 1998), *RockSample[7,8]* ($|S| = 12544, |A| = 9, |O| = 2$, *max reward*$= 10$) (Smith and Simmons 2004), and *Homecare* ($|S| = 5408, |A| = 9, |O| = 30$, *max reward*$= 10$) (Kurniawati, Hsu, and Lee 2008). While the size of *Fourth* does not seem great, the noise in the actions and the observations make this problem quite challenging.

Other tests are problems that used to be test bench marks in the literature but now considered to be too small to be challenging for the state of the art solvers. The only reason for performing our method on these problems is to test the near-optimality of the solution found in a larger set of problems. In fact, SARSOP (Kurniawati, Hsu, and Lee 2008) is a better approach for small problems. These tests are *Hallway* ($|S| = 61, |A| = 5, |O| = 21$, *max reward*$= 1$) (Littman, Cassandra, and Kaelbling 1995), *RockSample[4,4]* ($|S| = 257, |A| = 9, |O| = 2$, *max reward*$= 10$) (Smith and Simmons 2004) and *Tag* ($|S| = 870, |A| = 5, |O| = 30$, *max reward*$= 1$) (Pineau, Gordon, and Thrun 2006).

We also tested some other state of the art POMDP solvers and compared the total reward and the time required for these approaches with our method. The solvers are SARSOP (Kurniawati, Hsu, and Lee 2008) and HSVI2 (Smith and Simmons 2004) from point-based methods, POMCP (Silver and Veness 2010) from online methods, and QMDP (Littman, Cassandra, and Kaelbling 1995) and entropy-weighting(EW) (Cassandra, Kaelbling, and Kurien 1996) from heuristics. For SARSOP, HSVI2, and POMCP, we used the implementations available from the developers of those methods, ZMDP 1.1.7 for HSVI2[1], APPL 0.94 for SARSOP[2] and POMCP 1.0[3]. QMDP was implemented by us and the resultant reward of entropy-weighting for navigation problems is reported from Cassandra's PhD thesis (Cassandra 1998). As a result, the required time is not available for this method. This heuristic was introduced specifically for robot navigation. In addition, we could not test the method as there are some parameters that should be set by the domain expert.

All approaches were tested on a personal computer with Intel core i7-2600K 3.40 GHz CPU and 16GB DDR3 RAM. The operating system was Ubuntu 10.4 and the programming language for all methods was C++. All methods were compiled with g++ 4.4.3. We tested the methods on the problems many times to find the optimum reward that they could gain and the minimum time needed for that reward. In the cases where a method could not reach its highest possible reward the reported reward is the reward of running the method for a limited time. This limit is one hour for SARSOP and HSVI2 and one minute for each trial for POMCP. One minute may seem unfair in comparison to the time given to the offline methods. However, the goal of these experiments is testing the pairwise heuristic, not comparing SARSOP with POMCP. In fact, one minute is more than a hundred times the time required for the pairwise heuristic to find the solution.

[1]http://www.cs.cmu.edu/~trey/zmdp/

[2]http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/

[3]http://www.cs.ucl.ac.uk/staff/D.Silver/web/Applications.html

Table 1: The Average Reward and the Time Required for the Methods on the Main Problems

Method	Average reward	Offline time(s)	Online time(s)
Fourth			
Pairwise Heuristic	$.56 \pm .02$	0	0.06
POMCP	$-2.3 \pm .36$	0	63.47
QMDP	$.33 \pm .01$	0	0.23
EW	.35	0	NA
SARSOP	$.53 \pm .01$	3651.56	0
RockSample[7,8]			
Pairwise Heuristic	$18.76 \pm .23$	0	0.03
POMCP	$15.09 \pm .21$	0	61.06
QMDP	0	0	0.20
HSVI2	$21.20 \pm .15$	80.00	0
SARSOP	$21.22 \pm .12$	31.33	0
Homecare			
Pairwise Heuristic	15.79 ± 0.81	0	0.45
POMCP	$-15.34 \pm .63$	0	40.84
QMDP	$9.86 \pm .43$	0	2.39
HSVI2	$16.44 \pm .85$	1108.00	0
SARSOP	$16.85 \pm .63$	302.19	0

For all methods we found the average reward for performing the test on a run of 1000 trials. We then determined the range of average rewards achieved over a set of 10 runs of the methods, and report the range and its midpoint. The time needed is not the same in all trials for our approach and in one trial the agent may reach the goal in fewer steps. Because of this, we reported the maximum total time (including all steps) among all trials. We should add that in performing a test, the code would continue until the agent reaches the goal states or the possible instant reward gets lower than 0.005. That is the loop terminates when for the loop variable, t, $\gamma^t max_{s,a}|R(s,a)|$ is less than 0.005.

The reward and the time needed for the main problems are in Table 1. We could not find the reward of HSVI2 for the Fourth problem as we got a "bad alloc" error after a few seconds of running the method. In addition, although the reported reward of POMCP for Homecare is for running it in less than a minute ($40s$), it is not its optimum reward. In fact, it needs at least around 90 seconds to get a better reward. As a result, we put its reward after 40 seconds of runtime.

As shown in Table 1, the pairwise heuristic gains the highest reward in the Fourth problem and its reward in the other two main problems is close to the reward of SARSOP and HSVI2 with only one step search and in much less time. The offline time column in Table 1 is a little confusing especially

Table 2: The Rewards Gained by SARSOP and HSVI2 for Different Times on RockSample[7,8] and Homecare Compared to Pairwise Heuristic

Method	Average reward	Total time(s)
RockSample[7,8]		
Pairwise Heuristic	$18.76 \pm .23$	$0 + 0.03$
SARSOP	$7.35 \pm .00$	$0.22 + 0$
SARSOP	$17.57 \pm .30$	$0.37 + 0$
HSVI2	$10.43 \pm .00$	$2.00 + 0$
HSVI2	13.87 ± 0.12	$4.00 + 0$
Homecare		
Pairwise Heuristic	$15.79 \pm .81$	$0 + 0.45$
SARSOP	$14.44 \pm .67$	$230.12 + 0$
HSVI2	$14.36 \pm .52$	$250.00 + 0$

because we put zero for our method. Actually, the offline computation in HSVI2 and SARSOP is different from our offline computation. Unlike our method, the offline computation in HSVI2 and SARSOP is dependent on the initial belief state. As a result, they should be performed again each time the initial belief state changes. But, our offline part should be performed only once for each problem, no matter what the initial belief state is. Also, note that the total online time is the time needed to find and execute the entire plan; it is not the time required for just one step.

Table 1 shows that the pairwise heuristic is definitely a better approach for solving the Fourth problem and its time needed for solving RockSample[7,8] and Homecare is much less than the time needed for SARSOP and HSVI2. Furthermore, our method always gains much more reward than QMDP, entropy-weighting and also POMCP with the time limit of one minute. In regards to RockSample[7,8] and Homecare, one may argue that even though SARSOP and HSVI2 need more time to gain the optimum reward, they might gain the same or even higher reward than our method in its required time ($0.03s$ for Rocksample and 0.6 for Homecare). For Rocksample problem, we tested these methods in $0.22s$, 7 times our time needed and $0.37s$, 10 times our time needed for SARSOP and $2.00s$ and $4.00s$ for HSVI2 and show the results in Table 2. As shown, they achieve less reward in 10 times the time required by the pairwise heuristic.

For Homecare, SARSOP needs $208s$ only for initialization, showing that it cannot generate any policy in less than 460 times the time required by the pairwise heuristic. Initialization time is $244s$ for HSVI2. We report the reward of SARSOP in $230.12s$, and HSVI in $250.00s$ in Table 2 showing that these methods gain less reward in more than 500 times the time required by our method.

The reward and the time needed for the other problems are illustrated in Table 3. This table shows that the pairwise heuristic gains a near-optimal reward in all tested problems.

Table 4 shows the parameters and required offline time of the pairwise heuristic in all problems. The reported offline time shows that in some large problems even the sum of offline and online time required for the pairwise heuristic is less than the time SARSOP and HSVI2 require for solving one trial.

Table 3: The Average Reward and the Time Required for the Methods on Classical Small Problems

Method	Average reward	Offline time(s)	Online time(s)
Hallway			
Pairwise Heuristic	$.81 \pm .02$	0	0.01
QMDP	$.33 \pm .03$	0	0.01
EW	.60	0	NA
HSVI2	$1.01 \pm .03$	1.00	0
SARSOP	$.99 \pm .04$	0.30	0
RockSample[4,4]			
Pairwise Heuristic	$16.21 \pm .32$	0	0.01
POMCP	$14.15 \pm .31$	0	10.12
QMDP	$3.29 \pm .36$	0	0.02
HSVI2	$17.91 \pm .12$	1.00	0
SARSOP	$18.03 \pm .06$	0.56	0
Tag			
Pairwise Heuristic	-7.18 ± 0.25	0	0.03
POMCP	$-6.44 \pm .45$	0	12.74
QMDP	-16.55 ± 0.32	0	0.05
HSVI2	$-6.30 \pm .10$	7.00	0
SARSOP	$-6.12 \pm .15$	1.07	0

Table 4: The Parameters and required offline time of the Pairwise Heuristic in Different Problems

Problem	λ	Comp Ratio	Max Iterations	Offline Time(s)
Fourth	.56	6	551	4.68
RockSample[7,8]	.85	3	151	231.69
Homecare	1	6	201	77.96
Hallway	.7	8	151	0.07
RockSample[4,4]	.85	3	151	3.09
Tag	1	4	151	1.19

Analysis and Discussion

We tested the pairwise heuristics on classical test benchmarks in the POMDP literature and got near-optimal solutions in all of them. However, our approach does not always work well especially if reducing uncertainty is not essential for getting the maximum reward. In fact, the biggest drawback of the pairwise heuristic is that there is no lower bound for the reward of its solution. However, there is a bound for the removing uncertainty phase in some cases. Golovin et al. used a similar pairwise heuristic named EC^2 (Equivalence Class Edge Cutting) in the active learning field and got a near-optimal policy with the following bound:

$$C(\pi_{EC^2}) \leq (2\ln(1/p_{min}) + 1)C(\pi^*) \qquad (10)$$

π^* is the optimal policy and π_{EC^2} is the policy generated by EC^2 method. $C(\pi)$ is the total cost of a policy and

p_{min} is the minimum probability in the initial belief state (Golovin, Krause, and Ray 2010). They tackled the problem of finding the correct hypothesis among many candidates by performing some available tests. Each test has a cost and the observation from performing a test is noisy. The goal is to minimize the total cost of needed tests for finding the correct hypothesis. The information gathering phase of our algorithm is convertible to the Golovin et al. approach in the case that the actions are deterministic.

Also, in calculating value functions of the pair, we assume equal probability for both states of the pair. We can define other value functions for the pairs to cover more situations: for example, one value function for the times that the probability of the states are close to each other and two more for (high, low) and (low, high) situations.

In fact, we have tested both modifications of considering noise in the actions and having more value functions for the pairs. However, they did not change the total gained reward much in our problems. One reason may be the nature of the test bench-marks. In the problems with noisier actions these modifications may help more. Another reason however is that these simplifications are only in the heuristic. Everything is considered in the belief updates in our online strategy and actions are selected and performed in single steps. As a result, these simplifications do not affect the solution that much. Further work is needed.

Conclusions

In this paper, we have proposed the pairwise heuristic to solve POMDP problems quickly. The resulting reward for our method is close to or sometimes better than the results of the other state of the art POMDP solvers while the time required is much less. The method uses only one step search to find the optimal strategy. This shallow search is the foundation of its computational efficiency, making it useful for large problems.

References

Brafman, R. I. 1997. A heuristic variable grid solution method for POMDPs. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*, 727–733.

Cassandra, A. R.; Kaelbling, L. P.; and Kurien, J. A. 1996. Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 963–972.

Cassandra, A. R. 1998. *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes*. Ph.D. Dissertation, Brown University, Department of Computer Science, Providence, RI.

Du, Y.; Hsu, D.; Kurniawati, H.; Lee, W. S.; Ong, S. C.; and Png, S. W. 2010. A POMDP approach to robot motion planning under uncertainty. In *International Conference on Automated Planning & Scheduling, Workshop on Solving Real-World POMDP Problems*.

Golovin, D.; Krause, A.; and Ray, D. 2010. Near-optimal Bayesian active learning with noisy observations. *CoRR* abs/1010.3091.

He, R.; Brunskill, E.; and Roy, N. 2010. PUMA: Planning under uncertainty with macro-actions. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI)*.

Kaplow, R.; Atrash, A.; and Pineau, J. 2010. Variable resolution decomposition for robotic navigation under a POMDP framework. In *Proceedings of the International Conference on Robotics and Automation*.

Khalvati, K., and Mackworth, A. K. 2012. Active robot localization with macro actions. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 187 –193.

Kurniawati, H.; Du, Y.; Hsu, D.; and Lee, W. S. 2011. Motion planning under uncertainty for robotic tasks with long time horizons. *International Journal of Robotics Research* 30:308–323.

Kurniawati, H.; Hsu, D.; and Lee, W. S. 2008. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proceedings of Robotics: Science and Systems*.

Littman, M. L.; Cassandra, A. R.; and Kaelbling, L. P. 1995. Learning policies for partially observable environments: Scaling up. In *Proceedings of the Twelfth International Conference on Machine Learning*.

Pineau, J.; Gordon, G.; and Thrun, S. 2006. Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research* 27.

Poon, K. M. 2001. A fast heuristic algorithm for decision-theoretic planning. Master's thesis, The Hong Kong University of Science and Technology.

Ross, S.; Pineau, J.; Paquet, S.; and Chaib-draa, B. 2008. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*.

Roy, N., and Gordon, G. 2003. Exponential family PCA for belief compression in POMDPs. In *Neural Information Processing Systems (NIPS)*, 1043–1049. MIT Press.

Silver, D., and Veness, J. 2010. Monte-carlo planning in large pomdps. In *In Advances in Neural Information Processing Systems 23*, 2164–2172.

Smith, T., and Simmons, R. G. 2004. Heuristic search value iteration for POMDPs. In *Proceedings of International Conference on Uncertainty in Artificial Intelligence (UAI)*.

Thrun, S.; Burgard, W.; and Fox, D. 2005. *Probabilistic Robotics*. Cambridge, MA,: MIT Press.

Viswanathan, P.; Little, J. J.; Mackworth, A. K.; and Mihailidis, A. 2011. Navigation and obstacle avoidance help (NOAH) for older adults with cognitive impairment: a pilot study. In *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility*, ASSETS '11, 43–50.