Two Heuristics for Solving POMDPs Having a Delayed Need to Observe

Valentina Bayer Zubek and Thomas Dietterich
bayer@cs.orst.edu tgd@cs.orst.edu
Department of Computer Science, Oregon State University, Corvallis, OR 97331 USA

Abstract

A common heuristic for solving Partially Observable Markov Decision Problems (POMDPs) is to first solve the underlying Markov Decision Process (MDP) and then construct a POMDP policy by performing a fixed-depth lookahead search in the POMDP and evaluating the leaf nodes using the MDP value function. A problem with this approximation is that it does not account for the need to choose actions in order to gain information about the state of the world, particularly when those observation actions are needed at some point in the future. This paper proposes two heuristics that are better than the MDP approximation in POMDPs where there is a *delayed need to observe*.

The first approximation, introduced in [2], is the even-odd POMDP, in which the world is assumed to be fully observable *every other* time step. The even-odd POMDP can be converted into an equivalent MDP, the even-MDP, whose value function captures some of the sensing costs of the original POMDP. An online policy, consisting in a 2-step lookahead search combined with the value function of the even-MDP, gives an approximation to the POMDP's value function that is at least as good as the method based on the value function of the underlying MDP. The second POMDP approximation is applicable to a special kind of POMDP which we call the Cost Observable Markov Decision Problem (COMDP). In a COMDP, the actions are partitioned into those that change the state of the world and those that are pure observation actions. For such problems, we describe the chain-MDP algorithm, which in many cases is able to capture more of the sensing costs than the even-odd POMDP approximation.

We prove that both heuristics compute value functions that are upper bounded by (i.e., better than) the value function of the underlying MDP, and in the case of the even-MDP, also lower bounded by the POMDP's optimal value function. We show cases where the chain-MDP online policy is better, equal or worse than the even-MDP online policy.

1 Introduction

The Partially Observable Markov Decision Problem (POMDP) models the interaction between a single agent and a partially observable environment. Partial observability results from noisy, imperfect, or expensive sensors. Noisy and imperfect sensors are not able to uncover the complete state of the world; an agent with expensive sensors may prefer not to incur the high cost of observing the complete state of the world. For example, consider a robot moving in a building. If its sensors detect only walls, then rooms with similar configurations will look the same. Even if its sensors give complete information, they may require large amounts of battery power so that it is too expensive to use them at all times. People must deal with partial observability on a daily basis. Driving, for example, involves acting and sensing in a partially observable environment (the driver has a limited visibility of the road ahead and of surrounding cars).

Finding optimal solutions for POMDPs is undecidable [12] and nearly optimal approximations are intractable [11]. Researchers have explored three main approaches to POMDP approximation. Any POMDP can be converted into a Markov Decision Problem (MDP) in belief space (see below), and one approach is to approximate this belief space MDP via value function approximation, factored decomposition, discretization of the belief state, or a combination of these [10; 14; 8; 15; 3]. The second approach is based on solving the underlying MDP (i.e., the same POMDP but with a fully observable state). This approach computes the optimal value function V^*_{MDP} and then applies it online to construct policies for the POMDP [4; 8]. Our approximation methods are related to this approach. The third approach attempts to directly construct a finite-state controller that implements a good POMDP policy [7; 13].

POMDPs are difficult to solve because the agent may be highly uncertain about the current state of the envi-

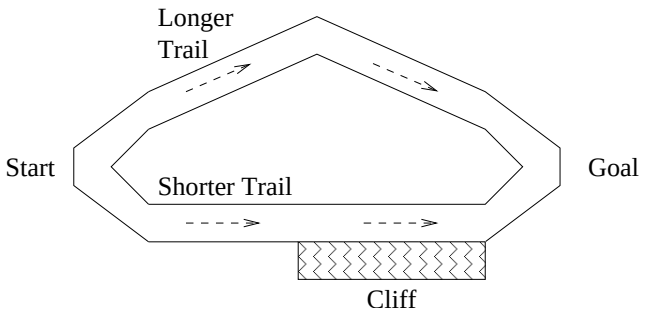

Figure 1: POMDP illustrating a delayed need to observe

ronment, i.e. it is "lost". In many problems, the agent needs to plan to gain information in order to "avoid getting lost". We call this the *"need to observe"*.

As an example of the need to avoid getting lost, consider a robot that has a choice of two different hallways to traverse. One hallway is completely dark and provides no visual landmarks. The other hallway is brightly-lit and has many visual landmarks. Even when the second hallway requires traveling a longer distance to reach the goal, it may still be the optimal choice, because the robot avoids getting lost in the dark hallway (and colliding with obstacles).

An immediate need to observe can be detected by performing a shallow lookahead search, evaluating the leaf nodes in this search tree using V^*_{MDP}, and backing-up these values to choose the best action to perform. For the robot considering the dark hallway, this kind of shallow lookahead search reveals that the robot will rapidly become uncertain of its position. The expected value of the resulting positions according to V^*_{MDP} is poor, so the robot prefers the well-lit hallway.

Unfortunately, a *delayed* need to observe is not detected by the above method. Consider the "skier problem" in Figure 1. It involves a skiing robot that starts at a known location at the top of the mountain and must choose which trail to take. The upper trail is very safe—so safe that the robot can ski this route with its eyes closed, because the trail goes through a bowl-shaped valley that naturally steers the robot down the center. The lower trail is initially just as safe as the upper one, but then it takes the skier along the side of a cliff. Here, the skiing robot must constantly observe its position and ski away from the cliff if it gets too close to the edge. Each time the robot uses its vision system, it consumes battery power, so the robot needs to minimize sensing. If this problem is solved while ignoring the costs of observation (i.e., computing V^*_{MDP}), the optimal policy will take the lower trail, because it is shorter. However, when the cost of observation is included, the upper policy is better. At the start state there is no apparent difference between the two paths (a limited lookahead will not detect any need to observe on either trail), and a shallow lookahead search combined with V^*_{MDP} will choose the cliff trail. The key difficulty is that there is a *delayed* need to observe (or equivalently, a delayed risk of getting lost), and the shallow lookahead search cannot overcome this delay.

To solve this problem, we need to propagate future observation costs backwards through the state space, such that earlier states will become more informed about the total cost of taking a certain path. This paper presents two approximate solutions that work well when there is a delayed need to observe. Because the methods involve solving MDPs with the same number of states as the original POMDP, they scale well to large POMDPs.

The first approximation, presented in detail in [2], defines a new POMDP, the even-odd POMDP, in which the full state of the environment is observable (for no cost) at all times t where t is even. When t is odd, the environment returns the same observation information as in the original POMDP (Figure 2). We showed that the even-odd POMDP can be converted into an equivalent MDP (the even-MDP or 2MDP) with different actions and rewards. The actions in this even-MDP are equivalent to 2-step lookaheads in the even-odd POMDP. Let V^*_{2MDP} be the optimal value function for the 2MDP. Then we get an improved approximation to the optimal POMDP value function by performing a shallow lookahead search and evaluating the leaf states using V^*_{2MDP}.

The 2-step lookahead of the even-MDP captures *some* of the costs of observation if the first step creates enough uncertainty about the state of the environment that the agent chooses to make an observation action. In other words, the even-MDP will incorporate the costs of observation in cases where those costs become immediately apparent at some point in the future. For example, in the skier domain, as the skier approaches the cliff, it becomes immediately apparent (i.e., to a 2-step lookahead search) that there is a need to observe. Hence, the V^*_{2MDP} will include those observations—but only at times when t is odd! As the 2MDP is solved, these observation costs will be propagated backwards along the temporal sequence of states so that in the starting state at the top of the mountain, the robot skier will be able to make the optimal decision to take the upper trail.

The even-MDP captures only a part of the observation costs, because it only needs to sense at odd times t (since the following state is assumed to be fully observable). Sometimes this produces a bad approximation.

The second approximation to POMDPs attempts to estimate observation costs at *every* time step. The model we propose, Cost Observable Markov Decision Problems or COMDPs, partitions actions into purely observational (i.e., *"observation actions"*) and state changing actions (i.e., *"world actions"*). Eric Hansen showed that a COMDP with only two observation actions—one that reveals the entire state and the other that observes nothing—can be converted into an MDP, provided there is a bound on the number of world actions taken between observation actions (see [5], [6], and also Sven Koenig's extension to sensor planning in [9]).

The chain-MDP algorithm is a heuristic for approximately solving COMDPs, starting with the MDP underlying the POMDP, and constructing a sequence of MDPs $M_1, M_2, \ldots$ whose reward functions have been modified to incorporate sensing costs. The sensing costs are esti-

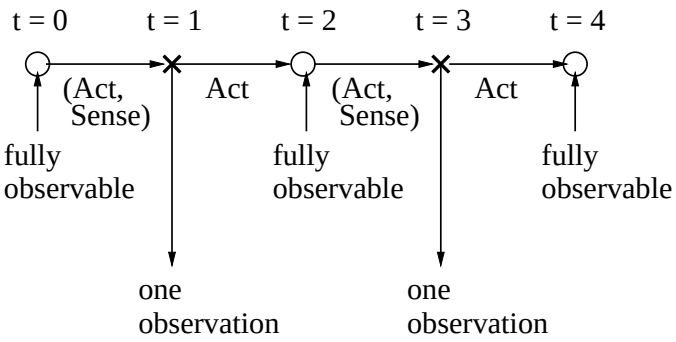

Figure 2: The even-odd POMDP fully observes the state (for free) every other time step

mated by performing a 2-step lookahead in each state, and evaluating the leaf nodes using the optimal value function of the previous MDP in the sequence. If this 2-step lookahead chooses a sensing action in a given state, then the reward function at that state is modified to include the sensing cost.

In the skier problem, the value function of the last MDP in the sequence of MDPs captures more of the observation costs than the value function of the even-MDP and yields a better approximation. While the chain-MDP method works very well in many cases, it is not always superior to the even-MDP approximation. We will show an example where it gives the same results as the even-MDP and another example where the even-MDP is superior. In general, therefore, neither method dominates the other, but both are usually much better than the MDP approximation.

2 POMDP Notations

A POMDP is a tuple $\langle S, A, O, P_{tr}(S|S,A), P_{obs}(O|S,A), R(S|S,A), \gamma\rangle$ where S is the set of states of the world, A is the set of actions, $P_{tr}(s_{t+1}|s_t,a_t)$ is the probability of moving to state s_{t+1} at time $t+1$, after performing action a_t in state s_t at time t, $R(s_{t+1}|s_t,a_t)$ is the expected immediate reward for performing action a_t in s_t causing a transition to s_{t+1}, O is the set of observations, $P_{obs}(o_{t+1}|s_{t+1},a_t)$ is the probability of observing o_{t+1} in state s_{t+1} at time $t+1$, after executing a_t, and γ is the discount factor. The states, actions and observations are discrete. We assume the model is known (the transition probabilities, the observation probabilities and the rewards). The rewards are negative or zero.

A Markov decision process (MDP) is a simplification of the POMDP where the agent is told the true state s of the environment after each action. Any POMDP can be converted into a continuous-state MDP called the belief MDP. The states in this MDP, called belief states, are probability distributions b such that $b(s_t) = P(s_t|a_0,o_1,\ldots,a_{t-1},o_t)$ is the agent's belief that the environment is in state s_t, given the entire action and observation history prior to t.

A policy π for an MDP is a mapping from states to actions. Hence, a policy for the belief MDP is a mapping from belief states to actions. The value function of a policy, $V^\pi(b) = E[\sum_{t=0}^{\infty}\gamma^t r_{t+1}]$, is the expected cumulative discounted reward of following policy π starting in belief state b. The optimal policy π^* maximizes $V^\pi(b)$ for all belief states. The value function of the optimal policy is denoted V^*. We will say that a belief state b is "pure" if $b(s) = 1$ for some state s, and instead of $V(b)$ we will write $V(s)$.

3 Even-Odd POMDP Approximation

The even-MDP method has two steps: (a) define and solve the even-MDP to obtain its optimal value function V^*_{2MDP}, and (b) choose actions for the original POMDP by performing a 2-step lookahead and evaluating the leaf states using V^*_{2MDP}.

3.1 Even-odd POMDP and Even-MDP

Given a POMDP we can define a new POMDP, the even-odd POMDP, where everything is the same except that at even times t, the set of observations is the same as the set of states ($O = S$), and the observed state is the true underlying state ($P_{obs}(o|s,a) = 1$ iff $o = s$). Note that at even times, the belief state will be pure, but at odd times, the belief state may become "spread out." The optimal value function for the even-odd POMDP can be computed by converting it into an equivalent MDP, which we call the even-MDP (abbreviated 2MDP). By "equivalent" we mean that the value function for the 2MDP is the same as the value function for the even-odd POMDP at even times t. At odd times, $V(b)$ is computed by performing a one-step lookahead search to reach an even time.

The 2MDP is constructed as follows. The states are the same as the even-odd POMDP's (world) states. Each action u in the 2MDP (Figure 3) is a tuple $\langle a, a'_1, a'_2, \ldots, a'_n\rangle$, where $n = |O|$. We will write $u[0] = a$ and $u[o_i] = a'_i$. An action u is executed in state s in the even-MDP by first performing $a = u[0]$ in the even-odd POMDP. The agent will move to state s' with probability $P_{tr}(s'|s,a)$, and an observation o will be received with probability $P_{obs}(o|s',a)$. The agent then executes action $a' = u[o]$, which will cause a transition to state s'' with probability $P_{tr}(s''|s',u[o])$. This is the fully observable result state in the even-odd POMDP. The probability transition function is $P_{tr}(s''|s,u) = \sum_{s'} P_{tr}(s'|s,u[0])\cdot \sum_o P_{obs}(o|s',u[0])\cdot P_{tr}(s''|s',u[o])$. The immediate reward of executing action u in state s is $R(s,u) = \sum_{s'} P_{tr}(s'|s,u[0])\cdot [R(s'|s,u[0]) + \gamma\sum_o P_{obs}(o|s',u[0])\cdot \sum_{s''} P_{tr}(s''|s',u[o])\cdot R(s''|s',u[o])]$. The discount factor of the 2MDP is γ^2.

The "Bellman backup operator" for this 2MDP is

$h_{2MDP}V(s) =$
$\max_u R(s,u) + \sum_{s''} P_{tr}(s''|s,u)\cdot\gamma^2 V(s'')$.

By expanding the definitions, this can be simplified to

$h_{2MDP}V(s) =$
$\max_a R(s,a) +$
$\sum_o \gamma \max_{a'} \sum_{s'} P_{tr}(s'|s,a)\cdot P_{obs}(o|s',a)\cdot$
$\sum_{s''} P_{tr}(s''|s',a')\cdot(R(s''|s',a') + \gamma V(s'')),$

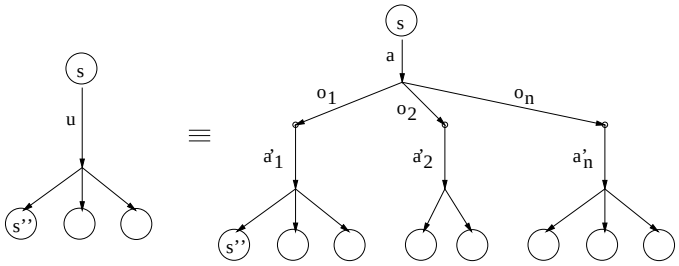

Figure 3: The even-MDP has actions of the form $\langle a, a'_1, a'_2, \ldots, a'_n \rangle$, where $n = |O|$

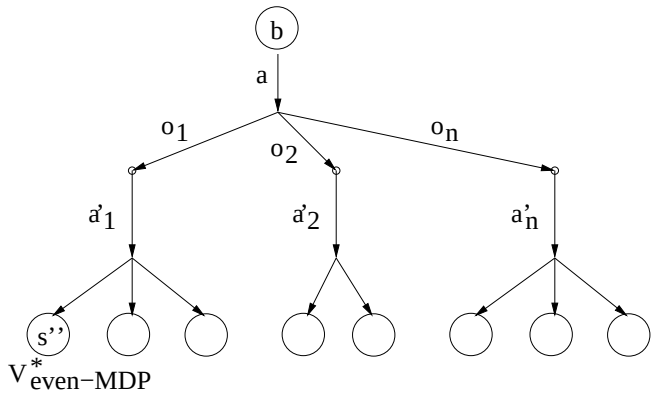

Figure 4: $LA(2)V^*_{2MDP}(b)$ computes the value for belief state b using a 2-step lookahead search and evaluating the leaf states s'' with V^*_{2MDP}

where $R(s,a) = \sum_{s'} P_{tr}(s'|s,a) \cdot R(s'|s,a)$.

3.2 Improved Approximation

Let V^*_{POMDP} be the optimal value function for the POMDP and V^*_{MDP} be the optimal value function for the underlying MDP.

Theorem 1 *(Bayer & Dietterich, [1; 2])* $V^*_{POMDP}(s) \leq V^*_{2MDP}(s) \leq V^*_{MDP}(s)$, *for all* $s \in S$.

Theorem 1 establishes that, on pure belief states, V^*_{2MDP} is a better approximation to V^*_{POMDP} than V^*_{MDP}. This result can be extended to arbitrary belief states b by considering a 2-step lookahead process. Let $LA(n)$ be an operator defined such that "$LA(n)V(b)$" estimates the value of belief state b by performing an n-step lookahead search in the original POMDP and evaluating the fully observable leaf states using V. Figure 4 shows how $LA(2)V^*_{2MDP}$ is computed.

Theorem 2 *For all belief states* b, $V^*_{POMDP}(b) \leq LA(2)V^*_{2MDP}(b) \leq LA(2)V^*_{MDP}(b) \leq LA(1)V^*_{MDP}(b)$.

Figure 5 depicts this relationship for a 2-state finite-horizon POMDP. All value functions are piecewise linear and convex. It is important to note that just because $LA(2)V^*_{2MDP}$ is a better approximation than $LA(2)V^*_{MDP}$, this does not guarantee that it will produce a better policy at run time. Nonetheless, it is usually the case that the more accurately we approximate the value

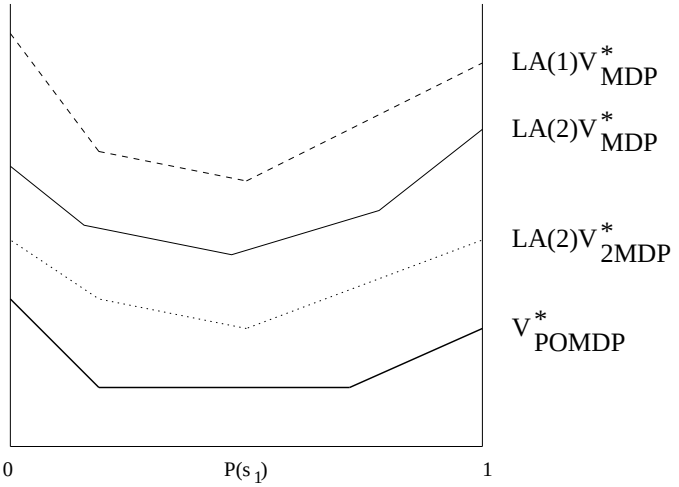

Figure 5: Schematic diagram of the optimal POMDP value function and three approximations to it for a 2-state finite-horizon POMDP

function, the better the performance of a greedy policy that corresponds to that value function.

To choose actions in the original POMDP, we first compute V^*_{2MDP} offline via value iteration. During online execution, we maintain a belief state b. At each time t, we perform the 2-step lookahead search to compute the backed-up value $LA(2)V^*_{2MDP}(b)$, as in Figure 4, and choose the action with the best backed-up value. The online policy needs to perform (at least) a 2-step lookahead in order to choose an action that performs some sensing (1-step lookahead assumes the resulting states are fully observable, so no sensing is needed).

The even-MDP value function underestimates the observation costs because it only needs to observe at odd times t. This is because at the end of the 2-step lookahead the states are assumed to be fully observable, so the second actions $u[o]$ of the even-MDP will not be chosen on the basis of the observation information they might provide. In the next section, we propose a heuristic that tries to overcome this limitation.

4 Cost Observable Markov Decision Processes (COMDPs)

To define the chain-MDP approximation, we need to be able to separate out the cost of sensing from the cost of acting in the world. In standard POMDPs, there is no fundamental distinction between these two: every action can simultaneously change the world and also provide information about the world that reduces the uncertainty in the current belief state. However, for the chain-MDP method, we will require that the POMDP have a special structure in which these two components are separated. To do this, we define a Cost Observable Markov Decision Problem (COMDP) as a tuple $\langle S, A, C, O, P_{tr}(S|S,A), P_{obs}(O|S,C), R_A(S|S,A), R_C(C), \gamma \rangle$, where A is a set of *world actions* that change the state

of the world according to $P_{tr}(S|S,A)$, and C is a set of *observation actions* that return observation information according to $P_{obs}(O|S,C)$. The world actions give a reward of $R_A(S|S,A)$, and the observation actions give a reward of $R_C(C)$, which does not depend on the current state. The set of states S and the discount factor γ are the same as in the POMDP definition. The set C of observation actions must include an observation action No-Obs that does not provide any information and has zero cost. All other observation actions have positive costs (equivalently, they have negative rewards), i.e., they are "cost observable".

The COMDP works as follows. At each time t, a COMDP policy π maps the current belief state b_t to a (world action, observation action) pair: $(a_t, c_{t+1}) = \pi(b_t)$. The world action a_t is executed, which causes the environment to move from state s_t to state s_{t+1}. Then the observation action c_{t+1} is executed to return observation o_{t+1} (see Figure 6). If c_{t+1} = No-Obs, then no observation information is received.

COMDPs are intended to model situations that arise in diagnosis and active vision where there are many observation actions that do not change the world.

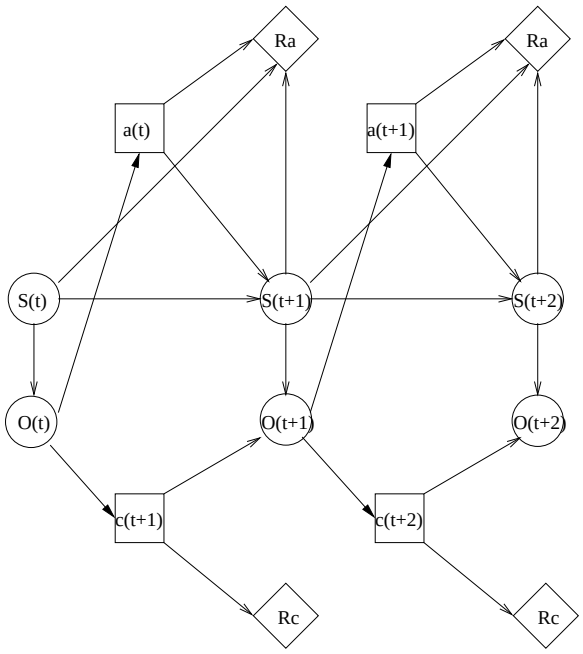

Figure 6: Decision diagram for a COMDP

It is easy to see that any COMDP can be converted into an equivalent POMDP by defining each POMDP's action $a'_t = (a_t, c_{t+1})$ as a pair of a world action a_t and an observation action c_{t+1} in the COMDP. Similarly, the equivalent reward function $R(s_{t+1}|s_t, a'_t)$ is the sum of the world action's reward $R_A(s_{t+1}|s_t, a_t)$ and the observation action's reward $R_C(c_{t+1})$. However, the restriction that there exist a No-Obs observation action in the COMDP means that not all POMDPs can be converted to COMDPs, because the No-Obs action means that every world changing action can be executed without receiving *any* observation information. We refer to the set of POMDPs that are equivalent to COMDPs as *EC-POMDP* (POMDPs that are Equivalent to COMDPs).

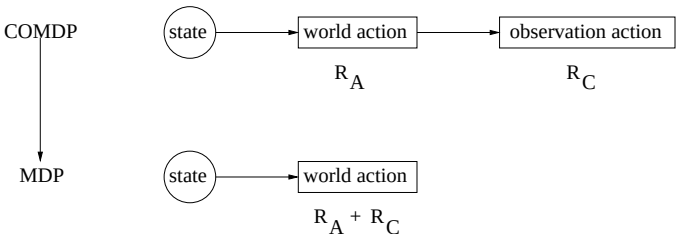

Figure 7: Idea for approximately solving a COMDP

4.1 Approximation Algorithm for COMDPs

The goal of our COMDP approximation algorithm is to create an MDP that incorporates the cost of sensing. That is, if the COMDP policy in a state s is to perform an action a followed by an observation action c, then we want an MDP where the cost for action a is changed to reflect the cost for observation action c (see Figure 7). We can then compute the optimal value function for this MDP by value iteration and employ it to choose actions online, just as we used the even-MDP value function.

Table 1 shows the chain-MDP algorithm for constructing this equivalent MDP. Let M_1 be the underlying MDP for the COMDP, that is, an MDP with the same $S, A, P_{tr}(S|S,A), R_A(S|S,A)$ and γ as the COMDP, but where the state is fully observable after each world action. Starting with M_1, the chain-MDP algorithm iteratively constructs a series of MDPs $M_2, \ldots, M_K$. The MDPs are all identical to M_1, except for the immediate reward function, which is changed to include the cost of observation actions.

To construct MDP M_{i+1}, the chain-MDP algorithm performs a lookahead search for each state s_t and action a_t assuming the world is cost observable for one step and fully observable in the second step. This lookahead search considers all possible sequences $(c_{t+1}, o_{t+1}, a_{t+1})$. During the lookahead, the cost of a_t is computed according to the original COMDP's reward function R_A for world actions, the cost of the observation action c_{t+1} is computed according to the reward function R_C for observation actions, but the cost of a_{t+1} is computed according to reward function R_i, from MDP M_i, since this may include some observation costs, which otherwise would not be captured by the 2-step lookahead. The resulting states s_{t+2} are evaluated using the optimal value function $V^*_{M_i}$ of MDP M_i from the previous iteration.

The purpose of this lookahead search is not to compute the backed-up value of the starting state s_t (as it was in the even-MDP approximation), but instead to determine which observation action c_{t+1} maximizes the expected return of this 2-step lookahead. Let c^*_{t+1} be the best observation action. The cost of this observation is

Table 1: Chain-MDP algorithm for approximately solving a COMDP

Let M_1 be the underlying MDP for the COMDP (the states are fully observable, at no cost), with the reward function $R_1(s_{t+1}|s_t, a_t) := R_A(s_{t+1}|s_t, a_t) = R_{MDP}(s_{t+1}|s_t, a_t)$.
Let $i := 0$ (iteration number)
Let i_{max} be a limit on the number of iterations
repeat

- $i := i + 1$
- solve the MDP M_i with reward R_i to find its value function $V^*_{M_i}$
- for each state s_t and action a_t
  - perform a 2-step lookahead search to choose the best observation action c^*_{t+1}, and for each possible observation o_{t+1}, choose the best next action a_{t+1}. At time $t+2$, we assume the states s_{t+2} are fully observable and we evaluate them using the value $V^*_{M_i}(s_{t+2})$ of MDP M_i.
  - for every valid transition to a state s_{t+1}, modify the reward for action a_t to include the cost for c^*_{t+1}

$$R_{i+1}(s_{t+1}|s_t, a_t) := R_A(s_{t+1}|s_t, a_t) + R_C(c^*_{t+1})$$

until $R_{i+1} = R_i$ or $i \geq i_{max}$
Return the last MDP M_K, with reward R_K and value function V^*_K.

---

incorporated into the reward function for MDP M_{i+1} by adding it to R_A:

$$R_{i+1}(s_{t+1}|s_t, a_t) := R_A(s_{t+1}|s_t, a_t) + R_C(c^*_{t+1}).$$

This rule is applied in all states s_t unless they are absorbing states. We require that absorbing states keep their MDP rewards unmodified so that the value of every absorbing state remains zero.

Once the chain-MDP has computed its final MDP, M_K, the value function V^*_K is applied to compute the online policy as follows (see Figure 8). At run time, a 2-step lookahead search is performed starting from the current belief state b_t to find the (action, observation action) $= (a_t, c_{t+1})$ with the best backed-up value. As in the chain-MDP algorithm, the reward for a_t is R_A, the reward for c_{t+1} is R_C, the reward for a_{t+1} is R_K, and the resulting states s_{t+2} are evaluated using V^*_K. It is important to use R_K to evaluate a_{t+1}—our experiments show that this is responsible for a significant portion of the improvement obtained from the chain-MDP approach.

We now prove that all the value functions $V^*_{M_i}$ are less-than or equal to V^*_{MDP}, the value function used in the MDP approximation. This also holds for the value function of the last MDP M_K, V^*_K, so it provides some evidence that the chain-MDP approximation will be better than the MDP approximation.

Theorem 3 *$R_i(s'|s, a) \leq R_{MDP}(s'|s, a)$ and $V^*_{M_i}(s) \leq V^*_{MDP}(s)$ for all s, s', a and i*

Here, R_i and $V^*_{M_i}$ are the reward and the value function of MDP M_i constructed by the chain-MDP algorithm in its i^{th} iteration.

Proof: For any of the MDPs M_i, because $R_i(s'|s, a) := R_{MDP}(s'|s, a) + R_C(c^*)$ and $R_C(c) \leq 0$ for all observation actions c, then immediately $R_i(s'|s, a) \leq R_{MDP}(s'|s, a) = R_1(s'|s, a)$ for all s, s', a. Let $h_{MDP_{R_i}}$ be the Bellman backup operator for the MDP with reward R_i. We will prove that its value function $V^*_{M_i}(s) \leq V^*_{MDP}(s), \forall s$, using an argument similar to the value iteration algorithm ([16]) to compute the value functions. Let $V_{init}(s) := 0, \forall s$, be the initial value function. We will prove by induction that $h^k_{MDP_{R_i}} V_{init}(s) \leq h^k_{MDP_{R_1}} V_{init}(s), \forall k, s$, where the notation $h^k_{MDP} V$ means the operator was applied k times (equivalent to k iterations of value iteration). Base step: $(k = 1)$ $h_{MDP_{R_i}} V_{init}(s) \leq h_{MDP_{R_1}} V_{init}(s)$ because $R_i(s'|s, a) \leq R_1(s'|s, a)$. Induction step: $h^{k+1}_{MDP_{R_i}} V_{init}(s) \leq h^{k+1}_{MDP_{R_1}} V_{init}(s)$ because $R_i(s'|s, a) \leq R_1(s'|s, a)$ and $h^k_{MDP_{R_i}} V_{init}(s) \leq h^k_{MDP_{R_1}} V_{init}(s)$ (induction hypothesis). Therefore $\lim_{k\to\infty} h^k_{MDP_{R_i}} V_{init}(s) \leq \lim_{k\to\infty} h^k_{MDP_{R_1}} V_{init}(s)$, so $V^*_{M_i}(s) \leq V^*_{MDP}(s), \forall s$. **Q.E.D.**

This theorem only relates the value function constructed by the chain-MDP algorithm to V^*_{MDP}. But the result can be extended to apply to the value functions computed during online execution. Let $LA^R(2)V(b)$ be the backed-up value for belief state b computed by an online 2-step lookahead search in which the leaf nodes are evaluated using value function V and the second step actions a_{t+1} have reward function R. Using this notation, $LA^{R_K}(2)V^*_K(b)$ is the backed-up value for belief state b using the last MDP M_K computed by the chain-MDP algorithm, whereas $LA^{R_{MDP}}(2)V^*_{MDP}(b)$ is the value computed using the underlying MDP's value function and reward function. The following theorem says that the backed-up values computed online using M_K are upper-bounded by the values computed online using the MDP approximation.

Theorem 4 *For all belief states b, $LA^{R_K}(2)V^*_K(b) \leq LA^{R_{MDP}}(2)V^*_{MDP}(b)$.*

Proof: Because $R_K(s'|s, a) \leq R_{MDP}(s'|s, a)$ for all s, s', a, and $V^*_K(s) \leq V^*_{MDP}(s), \forall s$ (from theorem 3 applied to the reward and value function of the last MDP M_K computed by the chain-MDP algorithm), it can be shown that $LA^{R_K}(2)V^*_K(b) \leq LA^{R_{MDP}}(2)V^*_K(b) \leq LA^{R_{MDP}}(2)V^*_{MDP}(b)$. **Q.E.D.**

In the chain-MDP algorithm (Table 1), we have included a maximum number of iterations i_{max}, because otherwise the chain-MDP may loop forever through a repeating series of MDPs. Here's an example of how this may happen. Assume the reward function R_2 of MDP M_2 captures some observation costs. In the next iteration of the chain-MDP algorithm, the 2-step lookahead evaluates the leaf states s_{t+2} using $V^*_{M_2}$. It is possible that this time the lookahead will find that sensing is not necessary, because the purpose is to maxi-

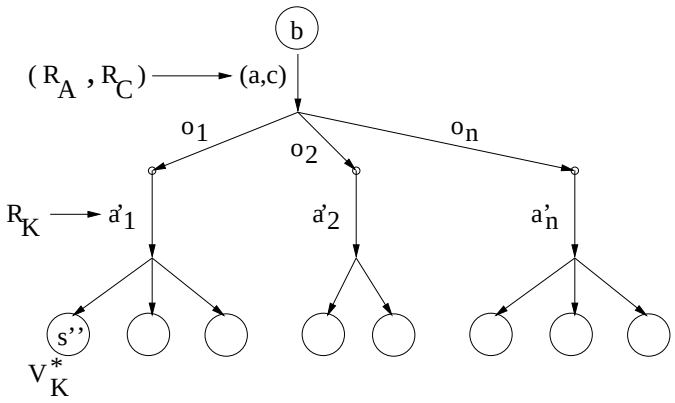

Figure 8: The chain-MDP online policy uses a 2-step lookahead exhaustive search to compute the best (action a, observation action c) in a belief state b

mize the expected return of the lookahead, and the resulting states s'' have lower values than before (during the lookahead in iteration 1) since $V^*_{M_2}(s'') \leq V^*_{MDP}(s'')$, plus $R_2 \leq R_{MDP}$ is used for a_{t+1}. Therefore the reward R_3 will not be modified, $R_3 = R_{MDP} \neq R_2$ and the algorithm will oscillate. This looping may not be a problem, however, because our experiments to date have not shown any special benefit to going beyond MDP M_2.

Instead of looping, another possible behavior of the algorithm is that it terminates in one iteration, with $M_K = M_1$. This occurs when none of the 2-step lookahead searches finds that an observation action is necessary. In some problems, such as those where no observation actions are available, this can't be avoided (see Section 5.3). But in other problems, if no observations are performed, uncertainty in the belief state gradually accumulates until major errors are made. Neither the even-MDP nor the chain-MDP approximations can handle the graduate accumulation of uncertainty (see such an example in Section 5.2).

An open theoretical question is whether the value function returned by the chain-MDP algorithm is lower-bounded by the optimal POMDP value function on pure belief states, that is, whether $V^*_{POMDP}(s) \leq V^*_K(s)$. If this is true, it would imply that whenever the chain-MDP algorithm produces a new MDP $M_K \neq M_1$, the online backed-up value function using M_K is a better approximation than the one using M_1, and the resulting online policy could be better than the policy computed from the MDP approximation. Based on our experiments, we believe V^*_K overestimates V^*_{POMDP}.

5 Comparison Between the Two Approximations

Now that we have introduced the even-MDP and chain-MDP approximations, we will compare them experimentally through a series of COMDP problems. The examples included here have a small number of states and are intended to demonstrate the strengths and weaknesses of the two methods. However, these small problems should not be taken to be representative of the size and complexity of POMDPs that can be solved. Both approximations scale well with the number of states, because they only involve solving MDPs. We demonstrated this in a previous study [2] where we presented a large EC-POMDP example with 10000 states and observations that is far beyond the capabilities of existing exact POMDP planning algorithms. However, the even-MDP approximation was easily computed, and the resulting policy gave excellent performance. In an unpublished experiment, we have applied the chain-MDP algorithm to this problem as well. It converges in 4 iterations and results in an identical online policy. This policy is much better than the policy computed using the MDP approximation.

5.1 Example with Delayed Need to Observe

Our first example demonstrates a case in which the chain-MDP produces the optimal policy, while the even-MDP produces a sub-optimal policy (and both are better than the MDP approximation).

Figure 9(a) shows a skier at the known start state on the left, and there are three trails leading down the mountain to circled absorbing states. The goal of the skier is to reach the bottom of the mountain by the least expensive path. Trail 1 is the longest path, but no sensing is required to traverse it, so it is optimal. Trails 2 and 3 are shorter, but each of them runs along a cliff where there is a danger of falling. To avoid this danger, the skier must observe its current position and take corrective action if he gets too close to the edge. These extra observation costs make Trails 2 and 3 suboptimal.

Here are the details concerning the world actions, observation actions, and reward functions for this example. The skier has three world actions: SE, N, and E. SE is only available in the start state, and it deterministically takes the skier to the start of Trail 3 (reward -1). N deterministically moves the skier north one square (reward -4). Bumping against a "wall" leaves the state unchanged. E normally moves east with probability 0.5 and southeast with probability 0.5. The reward of E is normally -1, but an additional penalty of -100 is received if the skier goes over the cliff. E moves east with probability 1 in the start state and in all states where there is no choice. There are two observation actions: No-Obs (reward 0) and Obs (reward –1). There are two observations: No-Cliff and Cliff. The first observation action No-Obs does not provide any sensing, so it always returns the observation No-Cliff. The second observation Obs deterministically returns the observation Cliff in states immediately adjacent to the cliff, and No-Cliff in all other states. The circled states and the cliff states are absorbing and any world action performed in an absorbing state has zero cost (and, of course, the world action transitions back to the same absorbing state). We still charge the cost of observation actions even if they are performed in absorbing states.

Because the number of reachable belief states in this

Table 2: Offline and online value functions $V^*(Start)$ (Fig. 9 (a))

	MDP	even-MDP	chain-MDP	POMDP
Trail 1	-18	-18	-18	-18
Trail 2	-17	-17.5	-19	-19
Trail 3	-16	-18.5	-20	-20

Table 3: Online policies' preferences (Fig. 9 (a))

MDP	even-MDP	chain-MDP	POMDP
Trail 3	Trail 2	Trail 1	Trail 1

problem is small and finite, we were able to compute the POMDP optimal policy and compare it with the even-MDP, chain-MDP, and MDP approximations. The chain-MDP algorithm converges in two iterations, and the resulting online policy is optimal; the even-MDP produces a sub-optimal policy; and the MDP approximation is worse than either the even-MDP or chain-MDP policies (see Table 3).

The reason the chain-MDP beats the even-MDP approximation is that it captures more of the observation costs. The value function computed by the chain-MDP ($V^*_{M_2}$) captures the costs of 4 observation actions on Trail 3, and of 2 observation actions on Trail 2. In fact, for this problem, $V^*_{M_2}$ captures all the necessary sensing costs. In contrast, the even-MDP value function (V^*_{2MDP}) detects only 2.5 observation actions on Trail 3 and 0.5 on Trail 2. The MDP value function, V^*_{MDP}, does not detect any need to observe, since it assumes all states are fully observable.

Table 2 shows the offline value functions V^* computed in the start state by the MDP, the even-MDP, the chain-MDP and the POMDP; these values are also equal to the online backed-up values in the initial "pure" belief state.

Interestingly, because of the online lookahead search, none of the online policies goes over the cliff. The policy computed by the MDP approximation, $LA(2)V^*_{MDP}$, will observe just as much on Trail 3 as the POMDP optimal policy would if the POMDP were to take Trail 3. Similarly, the even-MDP policy, $LA(2)V^*_{2MDP}$, will observe just as much on Trail 2 as the POMDP optimal policy would on this trail. This shows that 2-step lookahead computations are enough to permit the MDP and even-MDP approximations to give sensible, if not optimal, results on this problem.

5.2 Example of Gradually Getting Lost

Our second example shows a case where neither the chain-MDP nor the even-MDP approximations are able to do better than the MDP approximation.

Figure 9 (b) shows a slightly different skiing problem.

Table 4: Offline value functions $V^*(Start)$ (Fig. 9 (b))

	chain-MDP $\equiv$ MDP	even-MDP	POMDP
Trail 1	-5	-5	-5
Trail 2	-3.8	-4.71	-5.4

Table 5: Backed-up value functions in the initial belief state of Fig. 9 (b). For the approximations, the operator $LA(2)V^*(b)$ is used, and for the POMDP, V^*_{POMDP}

	chain-MDP $\equiv$ MDP	even-MDP	POMDP
Trail 1	-5	-5	-5
Trail 2	-4.71	-4.71	-5.4

Table 6: Online policies' preferences (Fig. 9 (b))

chain-MDP $\equiv$ MDP	even-MDP	POMDP
Trail 2	Trail 2	Trail 1

Here we have changed the dynamics so that the E action moves east with probability 0.9 and southeast with probability 0.1. The optimal POMDP policy is to take Trail 1, but all the approximations take Trail 2 (see Table 6). The MDP approximation ($LA(2)V^*_{MDP}$ policy) takes Trail 2 for the same reasons as before: it cannot detect the need to observe. Unfortunately, the even-MDP approximation and the chain-MDP approximation have the same problem: the probability 0.1 of moving southeast is not large enough to cause the 2-step lookahead to choose an observation action. So the even-MDP and chain-MDP do not detect the need to observe. This illustrates a weakness of both approximations: the gradual accumulation of uncertainty. If uncertainty accumulates gradually, these approximations will not detect the need to observe.

Interestingly, the value function computed offline by the even-MDP approximation, V^*_{2MDP}, is a more accurate approximation to the optimal value function than the value functions computed by the chain-MDP and the underlying MDP approximations. The reason is that although the even-MDP does not detect a need to observe, its value function in state s_t reflects the uncertainty about the next state s_{t+1}, by choosing the same action a_{t+1} in belief state b_{t+1}. This uncertainty is eliminated at time $t+2$ when the even-MDP is able to fully observe the state s_{t+2}. In contrast, both the chain-MDP and the underlying MDP, when computing their value functions, always assume the world states are fully observable at all times t. Hence, at time $t+1$, instead of taking the action that is the best for *all* the (undifferentiated) world states in belief state b_{t+1}, as the even-MDP does, they will take the best action in *every* fully observable state s_{t+1}. On this problem, the chain-MDP makes no changes to the reward function, and hence, it converges to the underlying MDP. Table 4 shows the offline values computed by the various methods.

On this problem, the increased accuracy of V^*_{2MDP} is not enough to make it choose the optimal action when applied online, but it does explain why sometimes the even-MDP approximation can be better than the chain-MDP approximation.

At execution time, the agent maintains a belief state, so it realizes when the uncertainty has accumulated, and it will choose to observe. As a result, the skier will usually avoid going over the cliff. Table 5 shows the value of the start state computed by the online lookahead search.

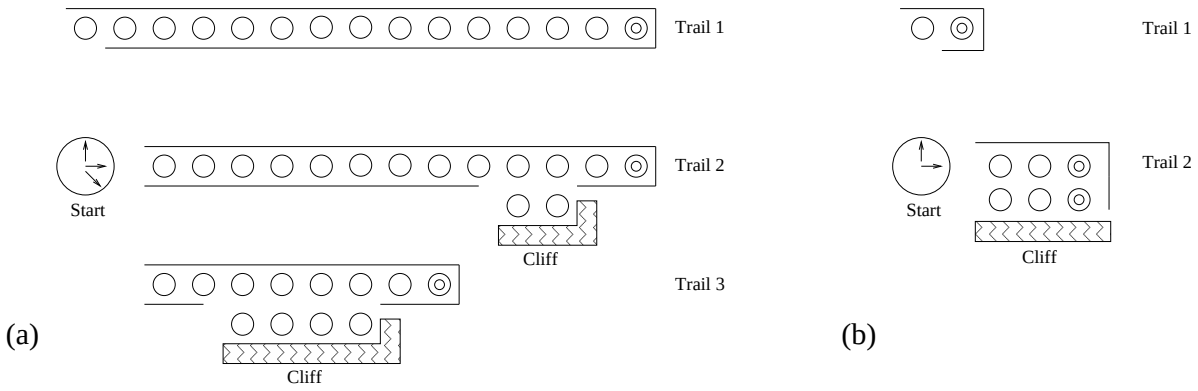

Figure 9: (a) Three paths for skiing down a mountain. There is a delayed need to observe on Trails 2 and 3. (b) A second skier example

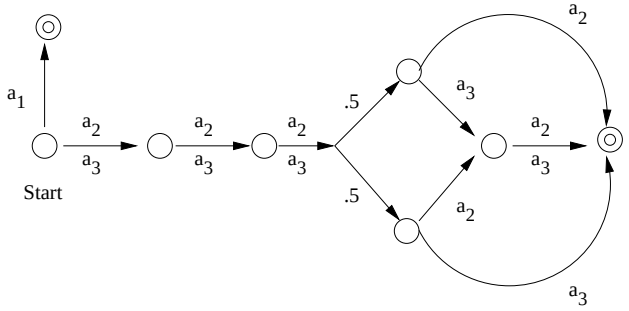

Figure 10: Example where no sensing is available

5.3 Example Where No Sensing is Available

Our third example shows a case where the even-MDP approximation gives a better policy than either the chain-MDP or the MDP approximations. The idea underlying this example is to create a situation in which the increased accuracy of V^*_{2MDP} results in a better policy.

Figure 10 shows a problem in which there is no sensing available (that is, $C = \{\text{No-Obs}\}$). There are three world actions a_1, a_2 and a_3. Actions a_2 and a_3 have a fixed reward of $r < 0$ in all states (except when executed in absorbing states). Action a_1 is only available in the start state, and it has a reward of $4.25 \cdot r$. Transition probabilities are 1.0 unless indicated otherwise.

Because there is no sensing available, the chain-MDP algorithm converges in one iteration and its value function is equal to V^*_{MDP}. As in the previous example, V^*_{2MDP} is a better approximation to V^*_{POMDP} than V^*_{MDP}, because the even-MDP experiences uncertainty for one time step, while the MDP assumes every state is fully observable.

Online, in the start state, a 2-step lookahead using V^*_{2MDP} prefers to take action a_1, because its backed-up value of $4.25 \cdot r$ is better than the backed-up value of $4.5 \cdot r$ of either action a_2 or a_3. Consequently, the even-MDP policy is optimal. In contrast, the chain-MDP overestimates the backed-up value of actions a_2 and a_3 as $4 \cdot r$ (see Table 7), so it prefers either action a_2 or a_3 and produces a suboptimal policy (Table 8).

Table 7: Action-value functions (Start state, Fig. 10)

	chain-MDP ≡ MDP	even-MDP	POMDP
a_1	$4.25 \cdot r$	$4.25 \cdot r$	$4.25 \cdot r$
a	$4 \cdot r$	$4.5 \cdot r$	$4.5 \cdot r$

Table 8: Online actions in the Start state (Fig. 10)

chain-MDP ≡ MDP	even-MDP	POMDP
a_2 or a_3	a_1	a_1

6 Conclusions

This paper has introduced two methods—the even-MDP and the chain-MDP—for obtaining approximate solutions to POMDPs. The methods seek to overcome a serious weakness of the commonly-used MDP approximation—namely, that the MDP approximation ignores the cost of observing the world. Both methods work by constructing an MDP whose value function captures some of the cost of observation. The first method, the even-MDP, is based on the even-odd POMDP in which the state of the world is partially observable at odd-numbered time steps and fully observable at even-numbered time steps. It captures observation costs during the odd-numbered time steps. The second method, the chain-MDP, is based on the Cost Observable MDP (COMDP), that separately models sensing and acting. The chain-MDP algorithm constructs an MDP in which the reward function has been modified to incorporate observation costs. By capturing these observation costs, both approximations usually give better policies than the MDP approximation. This is particularly true when there is a *delayed need to observe*—that is, when observation costs must be detected long before they are incurred. The delay means that *online* strategies based on a shallow lookahead search using V^*_{MDP} cannot detect the need to observe soon enough, but *offline* strategies that propagate detected observation costs backwards through time can succeed.

Of the two approximations, the chain-MDP method captures more of the observation costs. This is because it can incorporate costs at all times t rather than just at the odd-numbered times. However, it requires that the POMDP be an EC-POMDP, i.e., that it be equivalent

to a COMDP. In contrast, the even-MDP approximation works for any POMDP, and it is better understood theoretically. This paper has shown examples where the chain-MDP gives better results than the even-MDP and vice versa, so neither method is always superior to the other. However, the cases where the even-MDP gives better results are somewhat contrived situations in which neither method detects a need to observe but the even-MDP models the resulting uncertainty slightly better. The cases where the chain-MDP method is better are more typical cases in which sensing must be performed. Consequently, when the chain-MDP method is applicable—that is, when the POMDP is a COMDP—we believe it will give better results than the even-MDP, when both methods detect a great need for sensing.

Both methods share the same fundamental weaknesses. Neither method addresses the problem of acting when the agent is lost (i.e., where the agent is highly uncertain about the current state of the environment). Both methods also suffer from gradual accumulation of uncertainty. If uncertainty accumulates gradually, the offline 2-step lookaheads performed by these heuristics will not detect the need to observe, because the belief state after the first step will not be sufficiently diffused to make sensing worthwhile. One solution to this problem is to conduct a k-step lookahead, which will capture the costs of sensing that are apparent within k steps. But the computational cost of this solution grows exponentially with k. Another disadvantage of both even-MDP and chain-MDP algorithms is that their offline computations are based only on belief states that result after the agent starts in a known state and performs a single action. These belief states are not very spread out, so they are less likely to encompass situations where the agent is highly confused about which state it is in. Therefore both methods will, in general, underestimate the observation costs.

This paper has included the following theoretical results. First, we have shown that the even-MDP approximation is at least as good as the MDP approximation, in terms of both the offline and online value functions that it computes. Finally, we have shown that the offline and online value functions computed by the chain-MDP are upper-bounded by the value functions computed by the MDP approximation. It remains an open problem to prove that the chain-MDP never underestimates the optimal value of a belief state. Such a result would prove that the chain-MDP is always at least as good as the MDP approximation.

References

[1] Bayer, V., Dietterich, T.: A POMDP Approximation Algorithm that Anticipates the Need to Observe. Technical Report 00-30-01, Oregon State University, Dept. of Computer Science (2000)

[2] Bayer Zubek, V., Dietterich, T.: A POMDP Approximation Algorithm that Anticipates the Need to Observe. PRICAI 2000, LNAI 1886 (2000) 521–532
http://www.cs.orst.edu/∼bayer/

[3] Bonet, B., Geffner, H.: Planning with Incomplete Information as Heuristic Search in Belief Space. AIPS 2000. AAAI Press/MIT Press (2000) 52–61

[4] Cassandra, A. R., Kaelbling, L.P., Kurien, J. A.: Acting under Uncertainty: Discrete Bayesian Models for Mobil-Robot Navigation. IROS-96. IEEE (1996)

[5] Hansen, E. A.: Cost-Effective Sensing During Plan Execution. AAAI-94. AAAI Press/MIT Press (1994) 1029–1035

[6] Hansen, E. A., Barto, A. G., Zilberstein, S.: Reinforcement Learning for Mixed Open-loop and Closed-loop Control. NIPS-9, MIT Press (1996)

[7] Hansen, E. A.: Solving POMDPs by Searching in Policy Space. UAI-14. Morgan Kaufmann (1998) 211–219

[8] Hauskrecht, M.: Value-function Approximations for Partially Observable Markov Decision Processes. JAIR, vol. 13 (2000) 33–94

[9] Koenig, S., Liu, Y.: Sensor Planning with Non-Linear Utility Functions. ECP-99. Springer Verlag (1999)

[10] Littman, M. L., Cassandra, A. R., Kaelbling, L.P.: Learning Policies for Partially Observable Environments: Scaling Up. ICML-95. Morgan Kaufmann (1995) 362–370

[11] Lusena, C, Goldsmith,J., Mundhenk, M.: Nonapproximability Results for Markov Decision Processes. Technical Report 275-98, University of Kentucky. (1998)

[12] Madani, O., Hanks, S., Condon, A.: On the Undecidability of Probabilistic Planning and Infinite-Horizon POMDPs. AAAI-99. AAAI Press/MIT Press (1999) 541–548

[13] McCallum, R. A.: Instance-based Utile Distinctions for Reinforcement Learning with Hidden State. ICML-95. Morgan Kaufmann (1995) 387–396

[14] Parr, R., Russell, S.: Approximating Optimal Policies for Partially Observable Stochastic Domains. IJCAI-95. Morgan Kaufmann (1995) 1088–1094

[15] Rodríguez, A., Parr, R., Koller, D.: Reinforcement Learning Using Approximate Belief States. NIPS-12, MIT Press (2000)

[16] Sutton, R. S., Barto, A. G.: Reinforcement Learning. MIT Press (1999)