Oracular Partially Observable Markov Decision Processes: A Very Special Case

Nicholas Armstrong-Crews and Manuela Veloso
Robotics Institute, Carnegie Mellon University
{narmstro,veloso}@cs.cmu.edu

***Abstract*— We introduce the Oracular Partially Observable Markov Decision Process (OPOMDP), a type of POMDP in which the world produces no observations; instead there is an "oracle," available in any state, that tells the agent its exact state for a fixed cost. The oracle may be a human or a highly accurate sensor. At each timestep the agent must choose whether to take a domain-level action or consult the oracle. This formulation comprises a factorization between information-gathering actions and domain-level actions, allowing us to characterize the value of information and to examine the problem of planning under uncertainty from a novel perspective. We propose an algorithm to capitalize on this factorization and the special structure of the OPOMDP, and we test the algorithm's performance on a new sample domain. On this new domain, we are able to solve a problem with hundreds of thousands of action-states and vastly outperform a previous state-of-the-art approximate technique.**

I. INTRODUCTION

The Oracular POMDP is like a traditional POMDP except there is an oracle that takes the place of observations. The oracle is an information source with full knowledge that is equally accessible regardless of belief or state; however, there is a cost to invoke the oracle. Despite having no observations, the system is still "partially observable" in that actions produce non-deterministic transitions between states and thus the agent is afflicted by uncertainty.

The presence of an oracle corresponds to many real-world situations, such as humans or robot sensors of high precision. Although we are studying environments with both observations and an oracle, in this paper we focus on OPOMDPs with only the oracle.

POMDPs provide an elegant and general framework for many realistic problems, but unfortunately are computationally intractable, in general [1] [2]. MDPs can be solved much more efficiently, but do not provide as rich or realistic a framework. Therefore, it is of practical significance to investigate frameworks "between" these extremes on the spectrum of observability, to achieve richly representative and efficient solution techniques. Some examples of such frameworks between POMDPs and MDPs are unobservable MDPs [3], which are POMDPs which produce no observations; Zubek and Dietterich's even-odd POMDPs [4], which acquire perfect state information at every other timestep; and Hansen's POMDP variant, which has no observations but acquires perfect information at finite intervals [5]. While covering much the same scenario, [5] does not analyze the special case in detail, proposes an algorithm with no better computational guarantees than standard POMDP algorithms, and indeed applies that algorithm to general POMDPs. Finally, Jaulmes et al. use the concept of an oracle in [6], though only for learning POMDP models and not at all in solving them.[1]

The OPOMDP is our contribution to this set of frameworks. It is a particularly useful one, insofar as it affords several advantages over general POMDP techniques. Most work on general POMDPs concentrates on finding vectors to represent the value function in continuous belief space [7] [8] [9] [10]; OPOMDPs open up the ability to consider the value function as a non-stationary function over discrete state space, a much more compact representation. Furthermore, the lack of observations in the OPOMDP allows for complete factorization of information-gathering actions and state-altering (or *domain-level*) actions. We present an approximate algorithm, called JIV or J^{MDP} Information Value, that takes advantage of these characteristics and runs in polynomial time. JIV can be viewed as an extreme of "belief-points" approaches, such as PBVI [11], HSVI2 [12], and PERSEUS [13]; it partially relies on Q^{MDP} [14], but does not suffer its shortcomings (i.e., JIV will choose information-gathering actions when necessary). As no domain exists in the literature that fits the OPOMDP framework, we introduce a new domain and show that JIV solves it quickly and effectively. We also scale up this new domain and show that JIV scales quite well, outperforming HSVI2 by several orders of magnitude on the largest instance.

II. THE OPOMDP, FORMALLY

We define OPOMDPs following the same formalism as MDPs and POMDPs, which we now review.

A. MDPs and POMDPs

The Markov Decision Process, or MDP, is a framework for modeling an agent moving stochastically among a finite set of states via probabilistic transitions influenced by the agent's choice of actions. The agent's next state is only dependent upon its previous state and its current choice of action. The agent receives a reward signal at each timestep, associated with the current state and action. In this paper, we consider the case of discounted reward and infinite horizon. The tuple $(\gamma, s_0, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$ specifies the MDP: γ is the scalar discount factor; s_0 is the initial state; $\mathcal{S}$ is the set of

[1] We developed both the concept and the term "oracle" independently from Jaulmes et al. – suggesting the term is descriptive and the concept useful.

all states in the world; $\mathcal{A}$ is the set of all possible actions; $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto \Pi(\mathcal{S})$ is the state transition function, written $\mathcal{T}(s, a, s')$ for the transition s to s'; and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the (immediate) reward function.

The objective of the agent is to maximize long-term expected reward $E[\sum_0^\infty \gamma^t r_t]$, where r_t is the immediate reward at time t associated with the state s_t and action a_t at time t. The agent maximizes this expectation by determining a policy $\pi : \mathcal{S} \mapsto \mathcal{A}$, which completely determines the agent's behavior. The optimal policy can be generated by, for each state s, greedily choosing the action that maximizes $J(s)$, the long-term expected reward for s assuming the agent acts optimally (also called the value function and often denoted V). $J(s)$ is given by Bellman's Equation (1), and can be solved by a variety of techniques (e.g., value iteration).

$$J(s) \equiv \max_a \left(\mathcal{R}(s, a) + \sum_{s'} \mathcal{T}(s, a, s') J(s') \right) \quad (1)$$

The Partially Observable Markov Decision Process, or POMDP, extends the MDP framework to allow for incomplete knowledge of state. The agent now receives observations that are generated stochastically from a set of possible observations at each state. Rather than true state, the agent maintains in memory a belief state, which consists of a discrete probability distribution over states. The POMDP is then Markovian in belief state, since all the agent requires to model the transition to next belief state is its current belief state and its action. The tuple for a POMDP, $(\gamma, b_0, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O})$, contains the same elements as an MDP (except for initial state), plus: b_0, the initial belief; Ω, the set of all possible observations; and $\mathcal{O} : \mathcal{S} \times \mathcal{A} \mapsto \Pi(\Omega)$, the observation function.

The optimal solution for a POMDP can be found by considering the problem as a continuous MDP in belief state. Unfortunately, this technique has been proven to be PSPACE-hard in computation time [1], and most realistic domains are intractable to solve. A variety of optimal and heuristic techniques have been proposed (far too many to list here), but computation time remains the primary limiting factor in applying POMDPs to real systems.

B. OPOMDPs

The Oracular POMDP, or OPOMDP, replaces observations with an additional action, o, that provides the agent with full state knowledge; of course, as the agent continues to act in subsequent timesteps, its knowledge again becomes fuzzy. The action o is associated with consulting an oracle and is available in every state; however, the oracle requires a flat fee for each consultation. The policy's output is now from the expanded action space of domain-level actions and the oracle action.

The OPOMDP is a special case of a POMDP; specifically, any POMDP with an action producing an observation that is unique to the current state while all other actions produce a single (NULL) observation in all states, is an OPOMDP. In addition, an MDP is a special case of an OPOMDP; it is an OPOMDP with oracle cost 0 that performs a domain-level action followed immediately by an oracular consultation. Hence, we can see that OPOMDPs are strictly "between" POMDPs and MDPs, both in terms of generality and observability.

We will now formulate the OPOMDP, not in the typical POMDP fashion, but instead in terms of belief state b (as in the belief state MDP [8]). The OPOMDP tuple is $(\gamma, \lambda, b_0, \mathcal{S}, \mathcal{A}, \tau, \rho)$, with: γ, the scalar discount factor; λ, the oracle cost; b_0, the initial belief; $\mathcal{S}$, the set of all states in the world; $\mathcal{A}$, the set of actions, including domain-level actions and the oracle action; $\tau : \mathcal{B} \times \mathcal{A} \mapsto \mathcal{B}$, the belief transition function (a.k.a. state estimator); and $\rho : \mathcal{B} \times \mathcal{A} \mapsto \mathbb{R}$, the (immediate) expected reward function. Note that the set of possible beliefs $\mathcal{B}$, which contains elements b, is a discrete probability distribution over states. It is induced from $\mathcal{S}$, so is not necessary for specifying the OPOMDP and need not appear in the tuple.

The belief transition function τ takes as input an action a and a belief b, where b is a discrete probability distribution over states: $b(s) = \Pr(s|\text{history})$. The output of τ is the updated belief state after taking action a, and thus is also a discrete probability distribution over states. Hence, we write: $b' = \tau(b, a)$. It is also sometimes convenient to write $\tau(b, a)|_{s'}$, which indicates $b'(s') = \Pr(s'|b, a)$.

The elements of the OPOMDP tuple can be induced from those of the MDP tuple:

$$\mathcal{S} \equiv \mathcal{S}^{\text{MDP}} \quad (2)$$

$$\mathcal{A} \equiv \mathcal{A}^{\text{MDP}} \bigcup \{o\} \quad (3)$$

$$\tau(b, a)|_{s'} = \sum_s b(s) \mathcal{T}^{\text{MDP}}(s, a, s') \quad \forall\, a \in \mathcal{A}^{\text{MDP}} \quad (4)$$

$$\rho(b, a) = \begin{cases} \sum_s b(s) \mathcal{R}^{\text{MDP}}(s, a) & a \in \mathcal{A}^{\text{MDP}} \\ \sum_s b(s) \mathcal{R}^{\text{MDP}}(s, \text{NO\_OP}) & a = o \end{cases} \quad (5)$$

Note that the NO_OP domain-level action is assumed to occur simultaneously with the oracle action. NO_OP often refers to staying in the same state, or simply letting the system proceed naturally. If the agent lacks a NO_OP action, then we simply set $\rho(b, o) = 0$.

III. THE JIV APPROXIMATION

Since an OPOMDP can be viewed as a POMDP, we could use the same techniques to solve it; however, we present an approximate algorithm that provides high-quality solutions at reduced computational complexity. First, however, we analyze the distinctive characteristics of the OPOMDP that the algorithm exploits, which also afford some illuminating insights into the nature of rewards under uncertainty and the value of information.

A. Insights

One important characteristic of the OPOMDP framework is the absence of observations. This fact makes the transitions between belief states totally deterministic and the agent can foresee to an arbitrary horizon the exact effect on its belief state of performing a sequence of domain-level actions. This

fact holds true until it consults an oracle, introducing non-determinism in the belief state transition. Furthermore, the lack of observations means that the only way the agent can attain information is through the oracle, *not* by any of its domain-level actions. In contrast, regular POMDPs make no distinction between domain-level and information-gathering actions [8].

Our factorization is not perfect; consider a robot (with no sensors) driving down a hallway. It might not know where it is in the hallway, but if it keeps driving straight for long enough it will certainly finish at the end of the hallway. In other words, uncertainty can be reduced somewhat just by choice of action. However, in practice these situations are rare and policies that rely on them are typically not very good.

If we assume the factorization *is* perfect, however, we know that a greedy decision to consult the oracle is a good one. Intuitively, if at some timestep the agent is confused and doesn't know what to do next, it should ask the oracle immediately rather than waiting, since, according to our assumption, any other action would only confuse it further.

Using our explicit factorization of the OPOMDP action set into strictly information-gathering actions and strictly state-altering actions, we can discuss the value of information. In an OPOMDP, the only type of "information" available is the oracle, resulting in perfect state knowledge. Hence, "information value" is essentially defined for us: the value of one of the perfectly informed beliefs. However, since we do not know beforehand what response the oracle will give, we must take the expectation over states using our prior belief. Note that in regular POMDPs, one could also define information value in this way, but there's no guarantee of ever being able to reach a perfectly informed belief and many ways to get information *without* ending up perfectly informed; so such a definition would be rather less useful.

Some previous attempts have been made to examine, equivalently, the cost of uncertainty. The Expected Entropy and Entropy Value techniques of Cassandra et al. [14] compress the uncertainty parameters into a single number, entropy, but do not address the cost of that uncertainty by examining it in connection with the value function over state space. Information (lack of uncertainty) has no inherent value; one can imagine a value function that is constant everywhere, so knowing more precisely the state does not help an agent accumulate greater reward. On the contrary, the value of information is only defined with respect to some value function over a specific state space. Similarly, the value of information is tied to the available actions; in a situation where the agent only observes and has no actions, it will accumulate the same reward regardless of whether it knows its state or not. Hence, in defining the value of information for an OPOMDP, we must reference the states, actions, and value function.

Similar to [14], we'd expect a close connection between an OPOMDP's value function over belief space and its underlying MDP's value function over state space (much closer than that of a general POMDP, because we can intermittently achieve perfect information via the oracle). Rather than solving for the true value of information, which is essentially as hard as solving the whole POMDP, we will use the MDP's value function as an approximation. This approximation, along with the perfect factorization assumption, is the basis of our solution algorithm.

B. The JIV Algorithm

We now present our algorithm for efficiently solving an OPOMDP using a greedy approximation to the value function. This approximation defines the value of information using the J^{MDP} value function, so we title the approximation and algorithm J^{MDP} Information Value, or JIV.

The basic idea is to first solve the underlying MDP during a pre-planning phase, then use the MDP solution at execution time. During execution, at each timestep, the agent decides whether to take the domain-level action that maximizes long-term expected reward, or to consult the oracle to reduce uncertainty and pay the immediate oracle cost. The former option uses the Q^{MDP} approximation, just as in [9]; however, the latter option addresses the major flaw of Q^{MDP} (namely, that it will never take information-gathering actions) by also considering the single available information-gathering action: consulting the oracle. Hence, we achieve essentially the same alacrity as Q^{MDP}, while executing a sane policy that keeps the agent well-informed.

The Q^{MDP} approximation is:

$$\widehat{J}^{\,\text{QMDP}}(b) \equiv \max_a \sum_s b(s) Q^{\text{MDP}}(s,a) \tag{6}$$

This is a weighted voting scheme, in which each state's vote for best action is weighted by the probability of being in that state. This approximation would be exact if its assumption were satisfied that the agent would acquire full knowledge at the next timestep and retain full knowledge thereafter.

We define the *information value* (J^{IV}) of a belief to be:

$$J^{\text{IV}}(b) \equiv E[J^{\text{cert}}(s)] = \sum_s b(s) J^{\text{cert}}(s) \tag{7}$$

Here, $J^{\text{cert}}(s)$ represents the long-term expected reward of being in state s *with full instantaneous certainty* and thereafter executing the optimal policy. The information value is the expected value over the states the oracle might say the agent occupies. Since the oracle tells the truth, this is an expectation over the agent's belief b. Upon consulting the oracle, the agent knows the true state with no uncertainty.

How do we determine the values for J^{cert}? We could start with the approximation $\widehat{J}^{\text{cert}}(s) \equiv J^{\text{MDP}}(s)$, then perform value or policy iteration to convergence; but we'd need to keep track of the value of every possible belief, an uncountably infinite set. We could discretize belief space, but the computation time would be prohibitive and would degrade with finer discretizations.

In favor of performance and simplicity over optimality (again, standard techniques could solve the problem as a POMDP), we choose to simply approximate $J^{\text{cert}}(s)$ with $J^{\text{MDP}}(s)$, the value function of the underlying MDP, since

it requires negligible computation, can be performed at execution time, and, as we shall soon see, produces an effective policy.

We define the *J^{MDP} information value* ($\widehat{J}^{\,\text{JIV}}$) of a belief to be:

$$\widehat{J}^{\,\text{JIV}}(b) \equiv \sum_s b(s) J^{\text{MDP}}(s) \tag{8}$$

Hence, we use $\widehat{J}^{\,\text{JIV}}$ to calculate the value of invoking the oracle and weigh this against the value of the best domain-level action. The long-term expected reward of executing an action a and thereafter executing the optimal policy is given by:

$$\widehat{Q}^{\,\text{JIV}}(b,a) \equiv \begin{cases} \rho(b,a) + \gamma \widehat{J}^{\,\text{QMDP}}(b') & \text{if } a \neq o \\ \rho(b,o) - \lambda + \gamma \widehat{J}^{\,\text{JIV}}(b) & \text{if } a = o \end{cases} \tag{9}$$

The two cases highlight the tradeoff between taking a domain-level action and an information-gathering action. With this definition of $\widehat{Q}$, Bellman's Equation is as usual:

$$\widehat{J}(b) = \max_a \widehat{Q}^{\,\text{JIV}}(b,a) \tag{10}$$

The policy is then given by:

$$\pi(b) = \arg\max_a \widehat{Q}^{\,\text{JIV}}(b,a) \tag{11}$$

As previously mentioned, JIV should typically perform better than Q^{MDP}, since it will sometimes choose the information-gathering action (oracle). Since both $\widehat{J}^{\,\text{JIV}}$ and $\widehat{J}^{\,\text{QMDP}}$ are upper bounds, JIV won't necessarily choose to consult the oracle at the optimal time nor with the optimal frequency, nor is it guaranteed to achieve higher reward than Q^{MDP} on all problems. But qualitatively, it does solve Q^{MDP}'s major shortcoming; and we will soon see quantitatively that JIV indeed performs quite well in comparison to Q^{MDP} and to HSVI2.

Finally, note that JIV can be applied in POMDPs that meet the requirement of an oracle but still have regular observations as well. The only necessary change would be to append a Bayesian update for observations to (4). JIV should still outperform Q^{MDP}, since it doesn't reason about observations, either.

The algorithm for execution using the $\widehat{J}^{\,\text{JIV}}$ approximation, the JIV algorithm, is listed as Algorithm 1. In this listing, SOLVE_MDP is any algorithm that solves the MDP.

C. Computational efficiency

The benefits of performing some calculations at execution time are not to be underestimated. "Reachable beliefs" approaches [13] [12] reduce the number of beliefs whose value must be computed, with great success; the "execution time" approach (as presented here) uses only the beliefs actually reached during execution, the *minimum* possible number of beliefs necessary for acting optimally.

In fact, the theoretical complexity of the algorithm is reduced from that of solving a general POMDP; now all we must do is: 1) solve the MDP beforehand, a polynomial-time problem [15]; then 2) at execution time, examine $|\mathcal{A}|$ actions, each with a single expectation to compute over $|\mathcal{S}|$ states,

Algorithm 1 EXEC_JIV$(\gamma, \lambda, b_0, \mathcal{S}^{\text{MDP}}, \mathcal{A}^{\text{MDP}}, \mathcal{T}^{\text{MDP}}, \mathcal{R}^{\text{MDP}})$

$(Q^{\text{MDP}}, J^{\text{MDP}}) \leftarrow$ SOLVE_MDP$(\gamma, \mathcal{S}^{\text{MDP}}, \mathcal{A}^{\text{MDP}}, \mathcal{T}^{\text{MDP}}, \mathcal{R}^{\text{MDP}})$
$\mathcal{S} \leftarrow \mathcal{S}^{\text{MDP}}$
$\mathcal{A} \leftarrow \mathcal{A}^{\text{MDP}} \bigcup \{o\}$
$b \leftarrow b_0$
loop
  $(j_s, a_s) \leftarrow$ CHOOSE_BEST_ACTION$(b, \gamma, \mathcal{A}^{\text{MDP}}, J^{\text{MDP}})$
  $j_o \leftarrow \rho(b,o) - \lambda + \gamma \widehat{J}^{\,\text{JIV}}(b)$
  if $j_o \geq j_s$ **then**
    $b \leftarrow$ ground truth {consult oracle}
  else
    $b \leftarrow \tau(b, a_s)$ {perform action a_s}

Algorithm 2 CHOOSE_BEST_ACTION$(b, \gamma, \mathcal{A}^{\text{MDP}}, J^{\text{MDP}})$

Ensure: Returns best domain-level action and its value (using the Q^{MDP} approximation)

$(j_s, a_s) \leftarrow (-\infty, \text{undefined})$
for all $a \in \mathcal{A}^{\text{MDP}}$ **do**
  $b' \leftarrow \tau(b,a)$
  $j \leftarrow \rho(b,a) + \gamma \widehat{J}^{\,\text{QMDP}}(b')$
  if $j > j_s$ **then**
    $(j_s, a_s) \leftarrow (j, a)$
return (j_s, a_s)

which clearly takes time linear in both actions and states. Hence, this algorithm is polynomial. Additionally, note that the computation time per timestep is minimal, so there is little concern of lagging decisions during execution.

Note that the worst-case complexity of optimally solving an OPOMDP is still intractable. Consider the case when the oracle is too costly to ever make consultation worthwhile; now we have an unobservable POMDP, whose difficulty is still NP-complete [3]. It remains an open question whether general OPOMDPs are PSPACE-hard.

IV. EXAMPLE DOMAIN

To illustrate the theoretic analysis above, we introduce the following example domain, The Wizard's Curse. It is necessary to construct a new example domain, because no POMDP instances in the literature satisfy the conditions of an OPOMDP. However, in future work, we plan to present a method that approximates a general POMDP with an OPOMDP, at which point examinations of classical POMDP instances with JIV will be feasible.

In a faraway kingdom, a princess is kidnapped by an evil wizard and imprisoned in a remote tower. Clearly, the hero must go rescue the princess. The wizard, knowing this, places a curse on the hero that blinds him. The king's hunting grounds lie near the tower, and if the hero travels through them he incurs a tax; however, he can't tell when this happens due to his blindness. The hero's only tools are his precise knowledge of the region (map) and a magic ring with which

he can consult the king's oracle for exact knowledge of his location; however, the king provides this service only at a nominal fee per invocation.

A. Specification

The state space is a 6×6 grid, as pictured in Figure 1. In each state, the hero (agent) can attempt to move in any cardinal direction or stay in place, but the outcome is noisy – see Table IV-A (if the hero ends up moving off the map, he'll be replaced on it at the closest cell). The oracle action does not change the true state, so its effect on state is equivalent to the STAY action. The oracle cost is $\lambda = .25$ and the discount factor is $\gamma = .75$. The immediate reward is the same across all actions and is only a function of state. The reward associated with the princess' location is +2; for the hunting grounds it is -1; and for all other states it is 0.

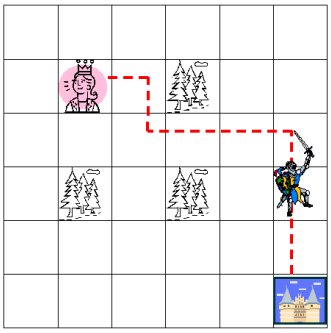

Fig. 1. Wizard's Curse problem

B. Experimental results

Figure 2 shows the contours of the J^{MDP} values and the hero's path for an example execution. The red line represents the true path the hero takes, while the blue line represents the sequence of most-likely states (MLS) at each timestep. An open circle indicates an oracle consultation, at which point the blue line breaks and begins again on the true path. Notably, the hero travels through the hunting grounds but believes he avoids it; the oracle is not consulted at this point because the hunting grounds penalty is not severe (-1) and the hero is reasonably certain he won't walk through it (although, in this rare case, he does anyway). However, the oracle *is* consulted twice near the princess, since in this area the value function changes steeply and the value of information is quite high. This example substantiates our previous theoretical analysis.

We compare the JIV algorithm with two naïve approaches in order to support our claim of JIV "behaving reasonably." One naïve approach, "Always Ask" (AA), acts timidly and consults the oracle every other timestep, thereby always having complete state information when choosing domain-level actions (essentially enforcing the assumption of the even-odd POMDP [4]). The other naïve approach is "Never Ask" (NA), which never consults the oracle and behaves equivalently to Q^{MDP} would if the POMDP were unobservable [3]. All of these algorithms, including JIV, utilize the MDP solution determined offline. Additionally, we compare

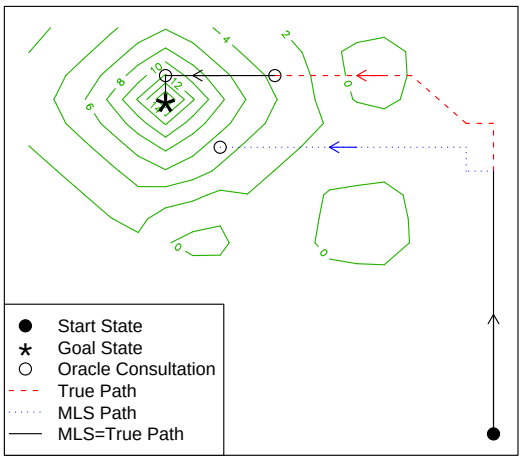

Fig. 2. Contours of J^{MDP} and agent path

JIV against HSVI2 [12], one of the most scalable general POMDP solvers available (unfortunately, not all competitive algorithms are available for comparison, and previous results for these algorithms on typical non-oracular problems are not particularly edifying).

Figure 3 compares the performance of these approaches by accumulated reward as the agent progresses in time, averaged over 500 runs. Overall, JIV far outstrips the naïve algorithms, while achieving statistically indistinguishable results from HSVI2 (which is guaranteed to be within .001 of the optimal value[2]). The dips in the plot correspond to when the agent moves between the two eastern hunting grounds and tends to lose reward. JIV dips lower than NA, since it pays the immediate cost to the oracle for the benefit of state information; but it is apparent that this behavior later pays off. AA doesn't produce a dip, since it successfully avoids the hunting grounds as a result of its perpetual state certainty; but its accumulated reward decreases drastically as it repeatedly pays the oracle cost (until it reaches the princess, at which point its accumulated reward begins to increase).

Figure 4a shows how JIV results in fewer oracle consultations as the cost of the oracle is increased, up until some point at which the oracle is never consulted. The number of consultations is averaged over 100 runs. This decline also satisfies intuitive expectations of reasonable agent behavior.

To examine scalability, we increased the size of the original domain by subdividing the grid an integral number of times in both the x and y dimensions. Figure 4b shows the necessary computation time for solving the MDP as we grew the domain.

A domain of 4,500 action-states was solved in 4.2 seconds with JIV, taking over 2 hours to solve with HSVI2. The largest domain solved consisted of over 300,000 action-states and took 3h49m. The simulation was run using the R software package on a Pentium IV 3.4 GHz machine

[2]Cassandra's optimal solver software was unable to find a solution.

TABLE I

NOISY ACTION OUTCOMES

Action	N			S			W			E			STAY		
	.1	.7	.1	0	0	0	.1	0	0	0	0	.1	0	0	0
Outcome	0	.1	0	0	.1	0	.7	.1	0	0	.1	.7	0	1	0
	0	0	0	.1	.7	.1	.1	0	0	0	0	.1	0	0	0

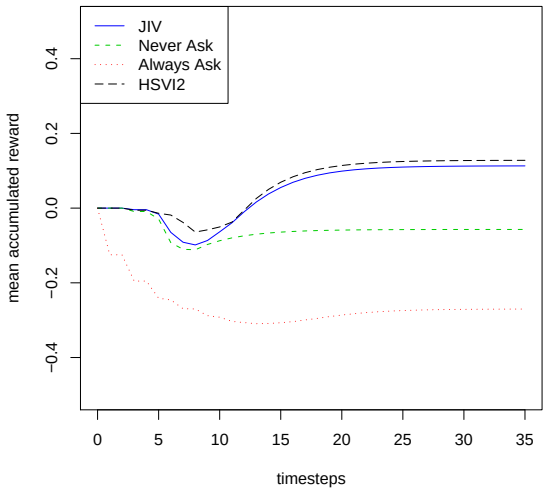

Fig. 3. Comparison of algorithms by accumulated reward

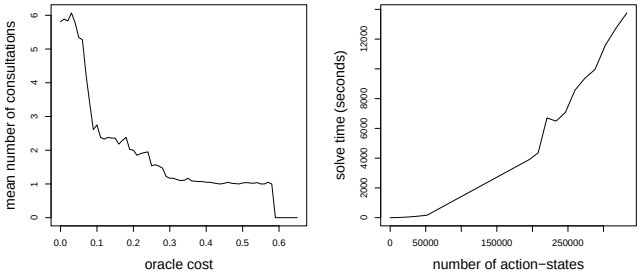

Fig. 4. **(a)** Frequency of oracle consultation as a function of oracle cost **(b)** Solve time (s) as a function of number of action-states

with 2 GB RAM. We used simple value iteration to solve the MDP; faster techniques suggested by [15] would surely improve JIV's performance, as well as would a C/C++ implementation. While we've only compared JIV against a single POMDP solver on a single domain, these initial empirical results are encouraging and demonstrate the soundness of our theoretical complexity results and of the JIV algorithm.

V. CONCLUSIONS AND FUTURE WORKS

The OPOMDP is a special case of the POMDP that deserves special attention, as its unique characteristics afford novel analytical techniques, tractable solution algorithms, and insight into the value of information in an uncertain environment. Furthermore, OPOMDPs correspond to a number of real-world cases and are therefore practically useful. This paper is merely an introduction to OPOMDPs; much more work remains to be done.

The JIV algorithm is perhaps the simplest of all algorithms that approximate J^{cert}, and in the future we will construct efficient, exact algorithms that draw on the special structure of the OPOMDP (in particular, heuristic search lookahead looks promising). Furthermore, in many cases, the oracle does not in fact know the truth entirely and exactly; hence, we plan to examine partial-knowledge oracles. These should allow us to approximate general POMDPs by treating observations as partial-knowledge oracles and thus make OPOMDPs and their efficient solution techniques much more universally applicable. This extension should also enable a larger comparison of JIV against standard POMDP algorithms on standard POMDP benchmark domains. Finally, we plan to address distributed multi-agent systems (DEC-POMDPs) with OPOMDP techniques by casting communicating agents as partial-knowledge oracles.

VI. ACKNOWLEDGMENTS

Special thanks to Tony Cassandra and Trey Smith for their open-source POMDP solvers; also to Reid Simmons (and again Trey Smith) for many helpful discussions. This research is partially supported by a DoD NDSEG fellowship and under Grant No. NBCH-1040007. The views and conclusions contained herein are those of the authors only.

REFERENCES

[1] C. Papadimitriou and J. Tsisiklis. The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[2] O. Madani. *Complexity Results for Infinite-Horizon Markov Decision Processes*. PhD thesis, University of Washington, 2000.

[3] O. Madani. Models for decision making in dynamic and uncertain domains. Technical Report UW-CSE-98-12-01, 1998.

[4] V. Zubek and T. Dietterich. A POMDP approximation algorithm that anticipates the need to observe. In *Proceedings of PRICAI-00*, 2000.

[5] E. Hansen. Markov decision processes with observation costs. Technical Report UM-CS-1997-001, 1997.

[6] R. Jaulmes, J. Pineau, and D. Precup. Active learning in POMDPs. In *Proceedings of ECML-05*, 2005.

[7] E. Sondik. *The Optimal Control of Partially Observable Markov Processes*. PhD thesis, Stanford University, 1971.

[8] L. P. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence Journal*, 101(1-2):99–134, 1998.

[9] M. Littman, A. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. In *Proceedings of ICML-95*, 1995.

[10] M. Hauskrecht. Value-function approximations for partially observable markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94, 2000.

[11] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of IJCAI-03*, 2003.

[12] T. Smith and R. Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *Proceedings of UAI-05*, 2005.

[13] M. Spaan and N. Vlassis. A point-based POMDP algorithm for robot planning. In *Proceedings of ICRA-04*, 2004.

[14] A. Cassandra, L. P. Kaelbling, and J. Kurien. Acting under uncertainty: Discrete bayesian models for mobile robot navigation. In *Proceedings of IEEE/RSJ-96*, 1996.

[15] M. Littman, T. Dean, and L. P. Kaelbling. On the complexity of solving markov decision problems. In *Proceedings of UAI-95*, 1995.