Online Heuristic Planning for Highly Uncertain Domains

Adam Eck and Leen-Kiat Soh
Department of Computer Science and Engineering
University of Nebraska-Lincoln
256 Avery Hall, Lincoln, NE, 68588 USA
{aeck, lksoh}@cse.unl.edu

ABSTRACT

Heuristic search algorithms for online POMDP planning have shown great promise in creating successful policies for maximizing agent rewards using heuristics typically focused on reducing the error bound in the agent's cumulative future reward estimations. However, error bound-based heuristics are less informative in highly uncertain domains requiring long sequences of information gathering, such as robotics. In these domains, all possible plan improvements look similar under error bound-based heuristics until the agent's belief uncertainty has been resolved, leaving the agent initially confused on how best to improve its plan under the real-time constraints of online planning. We propose (1) a novel heuristic guiding the agent towards policies that first reduce the agent's belief uncertainty, after which error bound-based heuristics are more effective, and (2) a novel selection mechanism for choosing which type of heuristic (error bound or uncertainty-based) to use during the current stage of planning to most quickly form a good plan. We evaluate our solution in several benchmark POMDP problems, demonstrating that our solution yields successful policies with less planning time in highly uncertain domains and comparable performance in simpler problems.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *intelligent agents, multiagent systems*

Keywords

POMDP; Online Planning; Heuristic Search; Uncertainty

1. INTRODUCTION

In real-world applications such as robotics (e.g., [5, 13]) and human-agent interactions (e.g., [14]) where intelligent agents must make decisions under uncertainty, agents frequently employ partially observable Markov decision processes (POMDPs) [3] to guide their reasoning. POMDPs are advantageous in such complex environments because they explicitly model the environment features causing uncertainty, provide a Bayesian framework for maintaining up-to-date beliefs using observations about the environment, and enable an agent to plan sequences of actions called policies that maximize its rewards and accomplish its tasks.

Online planning algorithms for POMDPs have received much recent attention, within which planning and execution are *interleaved* whilst operating in the environment [10]. Such algorithms empower the agent to plan in *real-time*, enabling it to always compute a response to its current situation. Offline planning algorithms (e.g., [4, 7]), on the other hand, perform all planning as a pre-processing step before deployment. As a result, although offline algorithms can afford more time for planning, they must produce generalized plans that need to fit as many (unknown in advance) situations as possible, potentially leaving the agent in a dangerous position if it encounters an unexpected situation, which is especially likely in complex domains such as robotics.

However, because online planning algorithms operate in real-time, they are strictly *time constrained*. As a result, the resulting policy's quality largely depends on how well the agent can explore the space of potential policies within the limited time allotted for planning. That is, the quality of a policy relies on how accurately the policy leads the agent toward large cumulative rewards, which requires *searching through an exponentially increasing number of future beliefs states* as the agent projects its actions farther into the future.

To efficiently search, one popular approach is heuristic search algorithms (e.g., AEMS2 [8] and FHHOP [15]), which incorporate heuristic information to guide the agent's planning to form the best policy as fast as possible. Previously reported algorithms (e.g., [8, 15]) have succeeded by focusing on *error bounds on the agent's cumulative rewards*. That is, by expanding plans towards regions of the policy space ultimately tightening the bounds on the expected cumulative rewards from the current belief state, the agent becomes more certain about the actual rewards it will earn and can better choose how to act to maximize its rewards.

Unfortunately, in environments with high uncertainty—where long sequences of information gathering actions are required to resolve belief uncertainty—heuristics evaluating the error bounds on agent rewards are generally less informative for improving the agent's policy. That is, when the agent is quite uncertain about the current environment state, it is also uncertain about the expected rewards for its actions, causing the error bounds on all actions to be relatively large. Thus, all possible directions during search look similar and the heuristic cannot differentiate how best to guide search to improve the agent's policy (especially so at the beginning of planning), negating the advantage of heuristic search. As a result, previously considered heuristics can

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

take a long time to find a good policy, which violates the real-time constraints placed on online planning. Therefore, existing heuristics might not be well suited for usage as-is in real-world applications with high uncertainty.

To overcome this problem, we propose a novel heuristic called Long Sequence Entropy Minimization (LSEM) focused on quickly resolving belief uncertainty. After resolving uncertainty, an agent is in a better position to use other heuristics to quickly find policies earning high rewards.

To leverage both heuristic types, we also contribute a novel Difference-based Heuristic Selection (DHS) mechanism that chooses the best heuristic to follow according to the current planning needs. DHS selects the heuristic that both (1) improves it's own value the fastest, leading planning to regions of the policy space that should *yield the most improvement*, and (2) optimistically maximizes the upper bound on rewards, leading planning to regions hopefully *closest to the optimal policy*. In this manner—unlike other heuristic search algorithms (e.g., FHHOP [15])—DHS enables us to consider and compare heuristics measuring different types of information to guide planning. Importantly, DHS also retains the theoretical guarantees of previously reported algorithms, such as convergence to an ϵ-optimal policy in finite time [9].

We empirically demonstrate the advantages of our approach in comparison to state-of-the-art online planning algorithms AEMS2 [8] and FHHOP [15] within several popular POMDP benchmark problems, finding that our approach speeds up planning by over an order of magnitude to find a successful policy in complex, highly uncertain domains and achieves comparable performance in other types of environments not requiring rapid belief uncertainty reduction.

2. BACKGROUND

2.1 POMDPs

A POMDP [3] models both the agent's uncertain, dynamic environment and the consequences of the agent's actions. Mathematically, a POMDP is defined as the tuple $P = \{S, A, T, Z, O, B, b_0, R\}$. $S = \{s\}$ represents the set of possible (hidden) states of the environment. The agent chooses actions from a set $A = \{a\}$ to accomplish its goals. In response to an action, the dynamic environment changes state according to the stochastic state transition function $T : S \times A \times S \to [0, 1]$ measuring the likelihood $P(s'|s, a)$ that action a changes the environment state from s to s'.

Since the environment state is hidden and dynamic, the agent is continually uncertain about the current state. To address uncertainty, an agent maintains probabilistic beliefs from the set B comprised of belief states $b : S \to [0, 1]$ measuring the probability the agent ascribes to each state being the correct current state (with initial belief state b_0 and current belief state b_c). Actions produce observations from a set $Z = \{z\}$ that reveal information about the current state. Such observations are also stochastic, occurring according to an observation function $O : S \times A \times Z \to [0, 1]$ measuring the likelihood $P(z|s', a)$ that observation z is observed after action a leads to state s'. After taking an action a and receiving observation z, the agent revises its belief state:

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

where $\frac{1}{\eta}$ normalizes $b^{a,z}$ to insure a valid distribution.

To guide the agent's decision making, $R : S \times A \to \mathbb{R}$ models the rewards received by the agent for taking an action in each state. To account for uncertainty in the agent's beliefs, agents consider the expected value of the reward function:

$$R(b, a) = E[R(s, a)|b] = \sum_{s \in S} b(s) R(s, a) \tag{2}$$

The agent builds a plan $\pi : B \to A$ called a policy that dictates what action the agent should take based on its belief state to maximize expected discounted, cumulative rewards:

$$\pi(b) = \arg\max_{a \in A} Q(b, a) \tag{3}$$

according to the recursive Bellman equations:

$$Q(b, a) = R(b, a) + \gamma \sum_{s \in S} b(s) \sum_{s' \in S} T(s, a, s') \sum_{z \in Z} O(s', a, z) V(b^{a,z}) \tag{4}$$

$$V(b) = \max_{a \in A} Q(b, a) \tag{5}$$

where $\gamma \in [0, 1)$ discounts rewards due to future uncertainty.

To plan a policy π satisfying Eq. 3, an agent must recursively solve Eqs. 4-5. This entails iteratively computing values of $Q(b, a)$ for additional belief states $b^{a,z}$ that the agent might experience in the future to accurately calculate the long-term cumulative value from its current belief state b_c. The tradeoff is that the farther into the future the agent plans, the more accurately it will account for future rewards and thus choose better actions, but deeper planning requires more time and the number of possible future belief states grows exponentially with planning depth n.

2.2 Online POMDP Planning

One approach to address the complexity of POMDP planning is to reduce the number of belief states considered while constructing π by localizing the beliefs to only those reachable from the agent's current belief b_c. This reduction enables deeper planning and thus more accurate cumulative reward estimations within a fixed amount of time, while not sacrificing coverage by only excluding beliefs the agent will not soon encounter. **Online POMDP planning algorithms** take this approach, where localized planning enables quick planning whilst also operating in the environment.

Due to real-time constraints, most online planning algorithms perform planning for a maximum allotted amount of time τ before taking each action. To do so, such algorithms compute and revise π using a tree structure where nodes represent the belief states[1] $b^{a,z}$ reachable from the current belief state b_c (and subsequent beliefs) after the agent takes each of the possible actions $a \in A$ and receiving observations $z \in Z$. Then, each path through the tree represents a sequence of actions (and received observations) projecting the agent's behavior into the future. Using each path, the agent can revise its estimates of the discounted, long-term cumulative reward of executing each action from its current belief state $Q(b_c, a)$. In this manner, the agent forms more accurate estimates about long-term reward after adding an additional belief state along any path through the tree. We illustrate this data structure in Fig. 1.

To construct the tree representation of π during online planning, agents follow the generic procedure presented in

[1] Following tradition, we override the notation of b to indicate both a belief state and its node in the tree

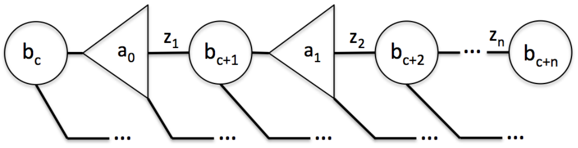

Figure 1: A π tree with path of depth n highlighted

```
PolicySearch($b_c, \tau$)
while $TimeSpent() < \tau$ and not $DoneSearching()$ do
    1. $b^*_{c+n}$ = ChooseLeafNode($\mathcal{L}$)
    2. Expand($b^*_{c+n}$)
    3. UpdateAncestors($b^*_{c+n}$)
end
return $\arg\max_{a \in A} \underline{Q}(b_c, a)$
```

Algorithm 1: Generic Online Policy Search Procedure

Alg. 1. Here, as long as the agent still has time for planning, it iteratively (1) chooses a leaf node $b^*_{c+n} \in \mathcal{L}$ from which to extend its plan (where $\mathcal{L}$ represents the set of leaf nodes in π), (2) expands the tree from b^*_{c+n} by considering the possible observations and rewards produced by each action and adds a new leaf node for each future belief immediately reachable from b^*_{c+n}, and finally (3) uses the new information to revise cumulative reward estimates for the beliefs in ancestor nodes along the path back to the root of the tree b_c using Eqs. 4-5. Each iteration of this procedure enables the agent to consider additional further future rewards in order to better estimate the long-term value of performing each action at the current belief state. This procedure can be executed as an anytime algorithm to iteratively improve the policy until the agent runs out of allotted time.

To improve long-term estimations since rewards beyond a leaf node are uncertain and not yet calculated during online planning, the agent maintains upper ($\overline{Q}$ and $\overline{V}$) and lower ($\underline{Q}$ and $\underline{V}$) bounds on the discounted, cumulative rewards from each node initialized using very simple pre-computed policies[2]. This information is then propagated back through the tree using analogues of Eqs. 4-5. After planning, the agent executes the action a maximizing the guaranteed discounted, cumulative reward represented by the lower bound $\underline{Q}(b_c, a)$ from the current belief state (although this action might not be optimal if the upper and lower bounds are far apart). For further details, please consult [10].

2.3 Heuristic Planning

Whereas the expansion (2) and update (3) steps of Alg. 1 are relatively straightforward according to Eqs. 1-5, an important piece of an online planning algorithm is deciding how to choose the leaf node $b^*_{c+n} \in \mathcal{L}$ to expand (Step 1). That is, the quality of the policy π depends on the accuracy of the agent's estimate of discounted, cumulative reward from each action in the current belief state, so the agent benefits from expanding the *most informative* belief states (with respect to its future rewards) during this step.

Two types of approaches are commonly used to choose which nodes should be expanded during planning in order to revise the agent's policy. First, *Monte-Carlo search* algorithms (e.g., POMCP [11]) exploit the stochastic probabilities T, O in the POMDP to expand planning along the most likely future belief states. Thus, this approach improves π by considering the future rewards the agent is most likely to earn, but can miss valuable rewards along less likely (but still possible) paths of subsequent actions.

Second, *heuristic search* algorithms (e.g., AEMS2 [8], FHHOP [15]) use a heuristic function $h : B \to \mathbb{R}$ to estimate the value of expanding each leaf node $b_{c+n} \in \mathcal{L}$ to improve π. Here, the agent expands the leaf maximizing h:

$$b^*_{c+n} = \text{choose}(h) = \arg\max_{b_{c+n} \in \mathcal{L}} h(b_{c+n}) \tag{6}$$

Provided with a good heuristic, this approach can guide planning to expand leaf nodes in the most informative manner, best revising the agent's policy in each iterative step of Alg. 1. Previously proposed heuristics AEMS2 and FHHOP favor leaves with the greatest error bound on future rewards, measured as the difference between upper and lower bounds:

$$e(b) = \overline{V}(b) - \underline{V}(b) \tag{7}$$

because these nodes have the greatest uncertainty in their subsequent future rewards (and thus could impart great uncertainty on the future rewards from current belief state b_c). After information from the new leaves are propagated back through the tree, this type of heuristic generally leads the agent to more accurate estimates of the cumulative rewards from its current belief, and thereby better decisions.

The first such heuristic to perform competitively with offline planning algorithms [10], in spite of using less time for planning, was the Anytime Error Minimization Search 2 (AEMS2) heuristic [8], which measures leaf node value as:

$$h_{AEMS2}(b_{c+n}) = e(b_{c+n}) \prod_{i=0}^{n-1} w(b_{c+i}, a_i) w(b_{c+i}, a_i, z_{i+1}) \tag{8}$$

where

$$w(b, a) = \begin{cases} 1 & \text{if } a \in \arg\max_{a' \in A} \overline{Q}(b, a') \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

determines whether a particular action along a path has the highest possible reward from a given belief state, and

$$w(b, a, z) = \gamma P(z|b, a) \tag{10}$$

measures the discounted likelihood of a future observation.

Altogether, AEMS2 optimistically leads search towards leaf nodes that (1) have the greatest uncertainty in their future rewards, indicated by $e(b)$, (2) along the path with the greatest possible cumulative reward (extending the IE-MAX heuristic [12]), indicated by $w(b, a)$, and (3) that have greatest likelihood, indicated by $w(b, a, z)$. By focusing on paths with the greatest possible cumulative reward, this heuristic is guaranteed to find a policy producing an ϵ-optimal action from b_c within finite (albeit possible large) time [9].

Recently, the Factored Hybrid Heuristic Online Planning (FHHOP) algorithm [15] proposed a variant of AEMS2:

$$h_{FHHOP}(b_{c+n}) = e(b_{c+n}) w_{1,2}(b_{c+n}) \prod_{i=0}^{n-1} w(b_{c+i}, a_i, z_{i+1}) \tag{11}$$

This heuristic aims to speed up how quickly the agent finds a good policy by favoring leaf nodes that are near-optimal with respect (instead) to the *lower* bound on future rewards

[2] Using algorithms such as Fast Informed Bound [2] and Blind policy [2] for upper and lower bounds, respectively.

$\underline{Q}$ (through the $w_{1,2}$ component, c.f., [15] for more details) since these values determine the action taken after planning.

Unfortunately, since h_{FHHOP} focuses on the lower bound $\underline{Q}$ (instead of upper bound $\overline{Q}$), there are no theoretical guarantees that a near optimal policy will be found using this heuristic *alone*. Instead, search can become trapped in local optima [15]. To avoid this problem, the complete FHHOP algorithm calculates both h_{AEMS2} and h_{FHHOP} *simultaneously*, then carefully selects which node to expand by comparing their weighted heuristic values to guarantee an ϵ-optimal policy in finite time [15].

Furthermore, the FHHOP algorithm also exploits a newer, special type of POMDP model called a mixed observability Markov decision process (MOMDP) [6] that factors the state space $S = \mathcal{X} \times \mathcal{Y}$ to exploit the reality that some state variables $\mathcal{X}$ are commonly fully observable by the agent, whereas only some state information $\mathcal{Y}$ actually suffers from partial observability. This exploitation speeds up two of the primary calculations in heuristic search-based online planning—belief state revision (Eq. 1) and expected observation probability (Eq. 10)—thereby further enabling the agent to perform more planning in a shorter amount of time [15]. However, this improvement is not specific to the FHHOP algorithm and can be used with any online planning algorithm.

3. SOLUTION APPROACH

3.1 Design Rationale

Although FHHOP is an improvement over AEMS2 in its simultaneous consideration of more than one heuristic to improve performance, the algorithm is still limited (just like AEMS2) to considering *only a single type of heuristic information* (i.e., bounds on future rewards). As discussed in Section 1, this type of information can lead to difficulties in highly uncertain domains requiring long sequences of information gathering to resolve the hidden environment state. In such domains, many uncertain belief states might have similar, relatively large gaps between upper and lower bounds on future rewards. Thus, *it is difficult for heuristics such as AEMS2 and FHHOP to successfully differentiate leaf nodes—especially so during initial planning—and guide search to expand the most appropriate nodes.* Instead, considering heuristics motivated to reduce uncertainty could better inform search and lead to better policies faster.

Moreover, *adding additional, differently motivated heuristics to FHHOP is not straightforward.* Specifically, the selection mechanism choosing which heuristic to favor for node expansion relies on the fact that the two heuristics (h_{AEMS2} and h_{FHHOP}) measure the same type of information so they are inherently normalized for comparison. Unfortunately, it is not clear how to normalize differently motivated heuristics (e.g., error bound-based vs. uncertainty-based), so there is a need for a new selection mechanism that can compare and properly select between different types of heuristics.

The big picture for our solution is the idea that for highly uncertain domains requiring long sequences of information gathering, we can split up agent planning and execution in into two separate stages, illustrated in Fig. 2: first belief uncertainty reduction, then reward maximization. In Stage 1, the agent begins with high uncertainty in its beliefs and must perform actions that reduce its uncertainty as fast as possible. Then, in Stage 2, once the agent is more certain about the true state of the environment, it can exploit its beliefs in order to maximize its rewards and accomplish its tasks. Planning in each of these two stages requires forming a policy that leads to different types of agent behavior. As described previously, existing heuristics such as AEMS2 and FHHOP are good at handling the second stage, but can struggle in the first since they fail to differentiate "good" leaf nodes to expand in order to resolve belief uncertainty.

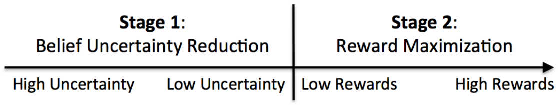

Figure 2: Stages in Highly Uncertain Domains

Therefore, we require both (1) a different type of heuristic for guiding planning to choose the best actions during the first stage, and (2) a mechanism for favoring one heuristic over the other dependent on the current stage of the agent's planning. We have developed the novel Long Sequence Entropy Minimization (LSEM) heuristic to accomplish the former, as well as the novel Difference-based Heuristic Selection (DHS) mechanism to accomplish the latter.

3.2 Long Sequence Entropy Minimization

In order to guide search towards good policies under high belief uncertainty, several pieces of information are valuable to the agent to differentiate different leaf nodes b_{c+n} with respect to quickly resolving uncertainty and leading to a good policy. First, because each belief state b is a probability distribution over environment states, its uncertainty can be directly measured by the *entropy* in the belief state:

$$H(b) = -\sum_{s \in S} b(s) \log b(s) \qquad (12)$$

To reduce uncertainty, heuristics should guide planning to beliefs with the least uncertainty, commonly[3] measured by:

$$C(b) = \log S - H(b) \qquad (13)$$

Although using $C(b)$ as a heuristic would lead the agent to expand belief states with the least uncertainty, it would not necessarily do so along paths from b_c toward large rewards. Thus, this information alone could easily become trapped in local optima. Instead, we combine $C(b)$ with three other pieces of information to better guide planning:

1. The upper bound on agent rewards from the leaf node $\overline{V}(b_{c+n})$ to optimistically guide the agent towards the optimal policy, similar to the $w(b, a)$ component of the AEMS2 heuristic,

2. The depth n of the leaf node from the current belief state b_c, in order to resolve uncertainty as fast as possible by guiding the agent towards the necessary large sequences of information gathering actions that reduce uncertainty,

3. The likelihood of reaching the belief state in the leaf node b_{c+n}, guiding search first towards beliefs most likely to be encountered during execution so that anytime Alg. 1 considers the expected most informative leaves first.

[3] This information has previously been added to the agent's *rewards* during planning in order to guide agent behavior towards uncertainty reduction (e.g., [1, 5]), but has never before been considered in a *heuristic* for guiding search.

Altogether, we combine such information in LSEM as:

$$h_{LSEM}(b_{c+n}) = C(b_{c+n})\overline{V}(b_{c+n})d(b_{c+n})\prod_{i=0}^{n-1} w(b_{c+i}, a_i, z_{i+1}) \tag{14}$$

where

$$d(b_{c+n}) = 1 + \log(n+1) \tag{15}$$

is a positive, increasing function that weights deeper belief states higher under the heuristic to encourage expansion during policy search along longer sequences of actions. We multiply in Eq. 14 to avoid domain-specific normalization across components with different ranges (upper bounds vs. entropy).

This heuristic function has several additional desirable properties. First, each component is non-negative, so the product is non-negative and increasing along paths that best reduce uncertainty, satisfying the goal of LSEM. Second, $d(b_{c+n})$ exhibits diminishing returns as n increases, allowing the heuristic to *avoid maximizing depth at the expense of the other components*. Otherwise, the heuristic could be biased towards continually extending the first long sequence chosen during expansion of π. Instead, LSEM can switch from recommending expanding one long path to a shorter path if the shorter path better reduces uncertainty.

Overall, LSEM guides search first towards the *least uncertain beliefs the agent is likely to encounter* using $C(b_{c+n})$ and $w(b_{c+i}, a_i, z_{i+1})$. If two leaf nodes have similar likelihoods and amounts of low uncertainty, preference is given to leaf nodes both (1) *possibly earning higher rewards*, measured by $\overline{V}(b_{c+n})$, and (2) *lying along longer sequences of information gathering actions* necessary for resolving uncertainty, measured by $d(b_{c+n})$. Thus, this heuristic correctly satisfies the requirements of planning in the first stage of Fig. 2, enabling search in this stage to overcome the challenges of previously reported heuristics (c.f., Section 3.1) for online POMDP planning in highly uncertain domains.

3.3 Difference-based Heuristic Selection

Even after carefully considering multiple pieces of information in the structure of h_{LSEM} to avoid suboptimal search, this heuristic is still not guaranteed to find optimal policies once the agent's belief uncertainty has been resolved (ending Stage 1). That is, in Stage 2, the heuristic alone could greedily continue to focus on reducing belief uncertainty along action sequences with possible (but not certain) large rewards and could fall into local optima. Instead, once the agent's uncertainty is adequately resolved, the agent should shift its planning focus to maximizing rewards. Here, the error bounds on future rewards from each leaf should be more highly differentiated because belief uncertainty is low, and thus error bounds should successfully guide planning.

Therefore, we propose employing LSEM *simultaneously* with other heuristics such as AEMS2 (or FHHOP) to improve agent planning. However, we face two key challenges:

Challenge 1: Determining how and when to switch between favoring (1) uncertainty reduction and (2) error bounds to optimize agent performance (i.e., identifying which stage the agent is in and choosing the appropriate heuristic).

Challenge 2: Producing an ϵ-optimal policy, as already guaranteed within finite planning using AEMS2 alone [9].

To address Challenge 1, our solution enables the agent to determine which stage it is currently in so that it can know which heuristic should be used to choose the leaf to expand in the next planning iteration. In the following, we consider exploiting h_{LSEM} in Stage 1 and h_{AEMS2} in Stage 2.

Our key insight is that in the different stages of the problem, the values of the different types of heuristics are changing at different rates, and we can leverage this to identify the agent's current stage. Specifically, in Stage 1, the h_{LSEM} heuristic initially increases quickly as uncertainty is reduced (increasing the logarithmic $C(b_{c+n})$ component of the heuristic), then as the agent approaches low belief uncertainty, the heuristic values at the leaves change slower after each expansion. Moreover, in Stage 2, the agent has already reached low uncertainty, so the values of h_{LSEM} continue to change slowly. In contrast, in Stage 1, the heuristic values for h_{AEMS} and h_{FHHOP} change slowly since most leaf nodes have similar upper $\overline{V}(b_{c+n})$ and lower bounds $\underline{V}(b_{c+n})$ (and thus similar $e(b_{c+n})$ values) due to the agent's uncertainty in its beliefs. Then, once the agent has less belief uncertainty in Stage 2, the bounds (and their difference) are more distinguished between the nodes and change at a faster rate as the agent finds good policies for maximizing its rewards.

Therefore, the key to determining which stage the agent is presently in is based on the change in the heuristic values at the leaf nodes: when the h_{LSEM} values are changing the most, then the agent is still in Stage 1 and should favor uncertainty reduction, whereas when the h_{AEMS2} and/or h_{FFHOP} values are changing the most, the agent is in Stage 2 and should favor reward maximization. Exploiting this rationale, we have developed the novel DHS selection mechanism for choosing which heuristic to guide expansion during online POMDP planning. This mechanism compares the relative changes in heuristic values to determine which heuristic to maximize during expansion.

Let k be the number of heuristics simultaneously considered during planning. For each heuristic $h_j \in \{h_1, \ldots, h_k\}$ at each leaf node $b_{c+n} \in \mathcal{L}$, the agent calculates the relative (undiscounted[4]) change in the heuristic along the path back to the root of the tree b_c before and after b_{c+n} was added:

$$\Delta_{h_j} = [h_j(b_{c+n})/\gamma - h_j(b_{c+n-1})]/h_j(b_{c+n-1}) \tag{16}$$

for monotonically increasing heuristics such as h_{LSEM}, and

$$\Delta_{h_j} = |h_j(b_{c+n})/\gamma - h_j(b_{c+n-1})|/h_j(b_{c+n-1}) \tag{17}$$

for error bound-based heuristics that decrease as the agent plans closer to its terminal states (causing $e(b)$ to decrease).

The higher the value of Δ_{h_j}, the more the heuristic value is changing, indicating that the agent is likely in Stage 1 or 2, depending on h_j: when $\Delta_{h_{LSEM}} > \Delta_{h_{AEMS2}}$, the agent is likely in Stage 1, and vice-versa for Stage 2. Additionally, the heuristic with the greatest rate of change is the most appropriate one for its current stage. Moreover, many commonly used heuristics for POMDP planing (e.g., AEMS2 [10], FHHOP [15]) and our LSEM assume that best policies lie near continually high heuristic-value branches and recommend leaves along such branches for choice in Alg. 1, so recommended leaves improving the most are a likely indicator of the best such candidate branch. Therefore, we can

[4] We divide the $h_j(b_{c+n})$ term by γ in Eqs. 16-17 to negate the difference caused solely by discounting, rather than the actual change in the heuristic's measured information.

use the heuristic maximizing the Δ function to know how to choose the leaf node to expand to best improve π.

However, to avoid becoming stuck with locally improving heuristics, we also consider the upper bound on the reward from the best leaf node for each heuristic (using Eq. 6) to optimistically bias heuristic selection towards the optimal policy (again similar to the IE-MAX heuristic for offline planning and AEMS2 for online planning). Thus, in DHS, the agent simply chooses the heuristic h^* to use for selection:

$$h^* = \underset{j \in \{1,\ldots,k\}}{\arg\max}\ \Delta_{h_j} \overline{V}(\text{choose}(h_j)) \tag{18}$$

then expands the leaf node maximizing that heuristic:

$$b^*_{c+n} = \text{choose}(h^*) \tag{19}$$

Of note, another potential benefit of our approach is that first reducing belief uncertainty through LSEM provides the agent with a planning scenario very similar to that encountered when assuming full observability (i.e., where there is full belief certainty). Given that approximations for upper bound $\overline{V}$ are often based on policies for a (completely or nearly) fully observable approximation of the POMDP (e.g., QMDP, FIB [2]), the agent can then exploit highly certain beliefs to quickly find good policies by following the gradient of information contained in $\overline{V}$. Thus, our solution might also be highly beneficial in other types of domains, as well.

3.4 Theoretical Performance

To address Challenge 2 and theoretically guarantee that our solution will produce an ϵ-optimal policy in finite time (as in AEMS2 alone [9]), we advocate using a slightly modified version of DHS when h_{AEMS2} is used with h_{LSEM}.

Specifically, consider a similar mechanism DHS' using the following selection rule, replacing Eq. 18:

$$h^* = \begin{cases} h_{AEMS2} & \text{if } N \mod m = 0 \\ h^* \text{ selected by Eq. 18} & \text{otherwise} \end{cases} \tag{20}$$

where N is the number of previous expansions to π, and $m \in \mathbb{N}$ is any natural number. Then, we find that:

Theorem 1. *DHS' is guaranteed to find an ϵ-optimal policy in finite time, as long as h_{AEMS2} is one of the heuristics considered by the selection mechanism.*

Proof. Alg. 1 using h_{AEMS2} alone is guaranteed to find an ϵ-optimal policy in finite time in at most $M < \infty$ expansions of the highest heuristic leaf [9] . Within $mM < \infty$ iterations, DHS' will also have caused Alg. 1 to select the highest heuristic leaf according to h_{AEMS2} for expansion at least M times, simulating at worst the search of h_{AEMS2} by itself for those M iterations. Therefore, DHS' also finds an ϵ-optimal policy in finite time. □

Selecting m for Eq. 20 can change the overall behavior of DHS' : a small m biases expansion in favor of AEMS2 and has a lower upper-bound on the time guaranteed to reach an optimal policy, but a small m also negates the advantages of considering other heuristics, and thus in practice might lead to longer planning time to actually find a good policy.

Intuitively, we suggest setting $m = k$, the number of heuristics simultaneously considered during planning. We use this particular setting in our experimental results to follow. Alternatively, m could be automatically adapted based on the agent's current situation, which we intend to explore as future work. For example increasing m proportional to current uncertainty would encourage more uncertainty reduction as necessary, whereas agents facing high cost actions could decrease m to best minimize costs.

4. EXPERIMENTAL SETUP

To empirically evaluate LSEM and DHS, we performed experiments using three popular POMDP benchmarks varying in complexity and uncertainty: (1) Tag [7], (2) RockSample(7, 8) [12], and (3) AUVNavigation [6].

In the **Tag** benchmark problem [7], a robotic agent and opponent are randomly placed in a 2D grid (with 29 cells) while playing laser tag. The agent's own position is fully observable, whereas the opponent's location is unknown to the agent and only discovered when the two are in the same cell. On the other hand, the opponent always knows the agent's location and tries to move away from the agent to avoid being tagged. The agent earns a cost for moving in each cardinal direction (N, S, E, W) to find its prey, and a reward for tagging the opponent, which ends the game. The agent's discounted rewards are maximized by finding and tagging the opponent as fast as possible.

In the **RockSample(7, 8)** benchmark problem [12], a robotic agent is tasked with collecting samples from 8 random rocks distributed throughout an 7 × 7 2D grid. Rocks are either of good or bad quality, and the agent's task is to only sample good rocks. The agent's location is fully observable, but the quality of rocks is unknown is advance and the agent can collect noisy information about the quality of a rock using a separate Check action for each rock. The agent can also move in each cardinal direction to be closer to rocks (which increases observation accuracy), or perform a Sample action to sample from the rock located in the same cell. Once the agent exits the far right of the grid, the problem ends. The agent earns a reward for sampling a good rock or exiting the grid and a penalty for sampling bad rocks. The agent's discounted rewards are maximized by sampling all (and only) good rocks and exiting as fast as possible.

In the **AUVNavigation** benchmark problem [6], a robotic agent is randomly placed in one of several starting locations in a $20 \times 6 \times 4$ 3D grid and tasked with navigating underwater through rock obstacles to reach one of two exit locations. The agent can Stay in its current position, turn Left, Right, Up, and Down to orient itself in the direction it wishes to move, or it can incur a small cost to move Forward to try to eventually reach the goal location. The agent's depth and orientation are fully observable, but its (x, y) position is unknown. The agent can surface to learn its (x, y) coordinates, which incurs a moderate cost, otherwise it observes nothing (unless it hits a rock or the goal). The problem ends when the agent either runs into a rock (incurring a large penalty) or reaches the goal (earning an even larger reward). The agent's discounted rewards are maximized by reaching the goal location as fast as possible while minimizing costs.

By considering multiple benchmarks, our goal is to determine whether our approach indeed *improves planning in highly uncertain domains*, as well as *how it performs in other types of environments*. Comparing these three benchmarks, they increase in order of *complexity* (in the sizes of the POMDPs listed in Tables 1-3) from relatively small Tag to moderate RockSample to difficult AUVNavigation. Considering *uncertainty*, they range from less uncertain RockSample (the agent always knows its own location and that of the terminal state) to more uncertain Tag (the agent doesn't know where its opponent is in the small 2D grid or how to reach it) to most uncertain AUVNavigation (the agent doesn't know its own starting location, nor how to build a path to the goal through obstacles across the larger 3D grid).

For our analysis, we compare the performance of our solution (with DHS simultaneously considering h_{LSEM} and h_{AEMS2}) against the two state-of-the-art heuristic search algorithms for online POMDP planning: AEMS2 [8] and FHHOP [15]. All three approaches use the FIB and Blind algorithms [2] to calculate upper and lower bounds, respectively, from a leaf node. We measure performance as the discounted ($\gamma = 0.95$), cumulative rewards earned by the agent. To understand the impact of differing real-time constraints common to real-world applications of online planning, we also vary the amount of time τ allotted for planning. Finally, we also compare our results against those previously reported for the state-of-the-art offline planning algorithms: SARSOP [4] and a special algorithm for MOMDPs [6].

For fair comparison, all experiments were conducted on the same computer with an Intel i5 (Haswell) 3.4GHz Quad Core processor (using only one thread) with 8GB of RAM (3GB were allotted for planning). All solutions and problems were implemented in Java. To more accurately measure performance, we ran each combination 1,000 times using different random seeds (only 100 times for the more complicated and time consuming AUVNavigation) and report 95% confidence intervals. To speed up planning, each problem was implemented[5] as a MOMDP model [6], as in [15].

5. RESULTS

Observations First, we consider the results from the Tag benchmark, presented in Table 1. We observe that LSEM & DHS performed quite well: very comparable to the state-of-the-art offline algorithms and better at low time allocations for planning ($\tau = 0.01$ seconds) than the other online heuristic algorithm relying on multiple heuristics (FHHOP). In fact, all algorithms were able to find policies with only 0.01 seconds of planning time per action, and increasing planning time did not significantly improve performance (except for lower FHHOP). Thus, it appears that in environments with low complexity (recall Tag had the lowest complexity of our three benchmarks), there is less need to divide planning into stages and treat belief uncertainty reduction separately. On the other hand, doing so did not adversely affect performance as our solution achieved comparable performance.

Next, we consider the results from RockSample, presented in Table 2. In this benchmark, we observe similar results as Tag: our approach generally outperformed FHHOP with the least amount of time allocated to planning ($\tau = 0.01$ seconds), was comparable to the state-of-the-art offline approaches as time increased, and was very similar to AEMS2 for all time allocations. Again, this similarity between our approach and AEMS2 is noteworthy, but in this instance because RockSample does not require long sequences of information gathering to know how to operate in the environment (instead it can start maximizing rewards with very little information gathering). Thus, RockSample does not require two stages of planning, yet our approach compares very similar to error bound-based heuristics alone that are ideal for this environment type. This demonstrates the efficacy of DHS to rely on the best type of heuristic for its current planning, since our approach could simply favor the h_{AEMS2} heuristic. Therefore, our approach appears safe to use in different types of domains, and not just highly uncertain ones requiring long sequences of information gathering.

Table 1: Tag Benchmark Performance

Tag $\lvert S\rvert = 870$ $\lvert\mathcal{X}\rvert = 30$ $\lvert\mathcal{Y}\rvert = 29$ $\lvert A\rvert = 5$ $\lvert Z\rvert = 2$			
τ	**AEMS2**	**FHHOP**	**LSEM & DHS**
0.01 s	-5.75 ± 0.38	-8.10 ± 0.41	-6.35 ± 0.42
0.05 s	-5.52 ± 0.39	-6.62 ± 0.38	-6.04 ± 0.36
0.10 s	-5.78 ± 0.38	-6.45 ± 0.38	-5.80 ± 0.38
0.50 s	-5.74 ± 0.39	-5.86 ± 0.37	-5.80 ± 0.38
Offline Algorithms [6] **MOMDP**: -6.03 ± 0.04 **SARSOP**: -6.03 ± 0.12			

Table 2: RockSample Benchmark Performance

RockSample(7,8) $\lvert S\rvert = 12,545$ $\lvert\mathcal{X}\rvert = 50$ $\lvert\mathcal{Y}\rvert = 256$ $\lvert A\rvert = 13$ $\lvert Z\rvert = 2$			
τ	**AEMS2**	**FHHOP**	**LSEM & DHS**
0.01 s	11.84 ± 0.34	7.36 ± 0.02	8.93 ± 0.24
0.05 s	18.22 ± 0.39	18.10 ± 0.38	18.09 ± 0.39
0.10 s	19.01 ± 0.39	18.93 ± 0.41	18.96 ± 0.41
0.50 s	19.59 ± 0.39	19.29 ± 0.38	19.20 ± 0.38
1.00 s	20.22 ± 0.42	20.38 ± 0.40	20.24 ± 0.40
Offline Algorithms [6] **MOMDP**: 21.47 ± 0.04 **SARSOP**: 21.47 ± 0.04			

Finally, we consider the AUVNavigation results, presented in Table 3. Recall that this is the ideal benchmark for evaluating our approach because it has the highest uncertainty and is much more complex than Tag, making it very difficult to find a good policy. Due to this complexity, we limited the agents to only performing at most 200 actions before we ended execution (an optimal policy requires roughly 25 actions), else the agent might have operated forever until it accidentally ran into a rock or randomly reached the goal. If the agent exceeded this limit for any of the 100 runs, we considered the algorithm to have failed to solve the problem.

Here, we observe that our solution (LSEM & DHS) indeed greatly improved the agent's ability to find a good policy. Specifically, it was able to find successful policies for navigating through the grid with much less time for planning: it successfully solved the problem with only 0.5 seconds of planning per action, whereas the other heuristics could not find an acceptable policy using an order of magnitude more time (5 seconds). These results are the fastest known online solution to this challenging problem ([15] reported a similar 10 seconds for error bound-based heuristics to form a good plan). Instead, for small τ, the error bound-based heuristics aimlessly chose actions until stopping after 200 actions.

Discussions Looking closer at planning behavior in AUVNavigation, our solution generally relied more on h_{LSEM} in the first few planning periods than for its last actions per run. This enabled the agent to first build a plan to determine its location, after which it relied almost exclusively on h_{AEMS2} as it navigated through the obstacles towards the goal. Therefore, our approach operated exactly as intended:

[5] Based on the POMDPX model configuration files at http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/index.php?n=Main.Repository

Table 3: AUVNavigation Benchmark Performance. DNF = did not finish within 200 executed actions

AUVNavigation			
$\lvert S\rvert = 13,536$ $\lvert\mathcal{X}\rvert = 96$ $\lvert\mathcal{Y}\rvert = 141$ $\lvert A\rvert = 6$ $\lvert Z\rvert = 144$			
τ	**AEMS2**	**FHHOP**	**LSEM & DHS**
0.5 s	DNF	DNF	317.8 ± 126.5
1 s	DNF	DNF	507.2 ± 107.3
5 s	DNF	DNF	568.6 ± 97.7
10 s	928.4 ± 107.6	DNF	599.2 ± 104.6
15 s	929.5 ± 107.8	843.8 ± 111.6	882.2 ± 109.1
20 s	928.4 ± 107.6	928.4 ± 107.6	928.7 ± 107.8
Offline Algorithms [6]			
MOMDP: 808.0 ± 3.4 **SARSOP**: 799.3 ± 2.9			

first the DHS mechanism used h_{LSEM} to reduce belief uncertainty, then it used h_{AEMS2} to maximize rewards.

Interestingly, AEMS2 alone was able to find a great policy faster than our approach (earning greater rewards at $\tau = 10$ seconds). It appears that using h_{LSEM} too often early can trap the agent near suboptimal policies and requires greater uses of h_{AEMS2} later to find a global optima. This issue could be addressed by tweaking the DHS selection mechanism, which we intend to explore as future work. However, even as-is, our approach is still very beneficial. Extrapolating to other environments in highly uncertain real-world domains such as robotics, it is better for the agent to have some acceptable policy than none at all under commonly strict real-time constraints for planning and reasoning. It is also reassuring that given more time, our approach eventually yields a solution closer to optimality.

Finally, note that in our results, AEMS2 performs much more favorably against FHHOP when compared to the initially reported results for FHHOP [15]. We believe this is due to our use of a MOMDP representation for all algorithms, whereas Zhang and Chen demonstrated the benefits of using a (new at the time) MOMDP representation only as part of their algorithm. Nothing about AEMS2 precludes the use of this more advanced POMDP model to speed up performance, so we used the same model for all algorithms in our comparison. If we had not, AEMS2 would have required much more time for planning, as reported in [15].

6. CONCLUSIONS

In conclusion, we have presented (1) a novel LSEM heuristic and (2) DHS selection mechanism for choosing between heuristics. Together, these advancements improve the quality of heuristic search for online POMDP planning in highly uncertain domains requiring long sequences of information gathering actions common to many real-world applications of intelligent agents, including robotics. Prior heuristics generally rely on information such as error bounds in the agent's cumulative reward estimations, which are less informative in such domains. Our solution relies on splitting planning into two separate stages: belief uncertainty reduction, followed by reward maximization. Our LSEM heuristic improves planning in the first stage, whereas DHS automatically detects when to switch from the first stage to the second and chooses the best heuristic to guide planning. Using three popular POMDP benchmark problems differing in complexity and uncertainty, we demonstrated that our approach (1) is very beneficial in highly uncertain domains, enabling the agent to find acceptable policies more than an order of magnitude faster than the state-of-the-art heuristics, and (2) can be applied to other types of domains without deteriorating performance and thus is generally safe to use.

We plan to extend this research by considering additional environments with even greater uncertainty to see how much uncertainty our solution can handle, including actual deployments to real-world systems, as well as improving DHS to yield near optimal solutions faster.

7. ACKNOWLEDGMENTS

This material is based upon work partially supported by the National Science Foundation under Grant No. SES-1132015.

8. REFERENCES

[1] M. Araya-Lopez, O. Buffet, V. Thomas, and F. Charpillet. A POMDP extension with belief-dependent rewards. In *Proc. of NIPS'10*, 2010.

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

[3] L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[4] H. Kurniawati, D. Hsu, and W. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proc. of Robotics: Science & Systems*, 2008.

[5] L. Mihaylova et al. Active sensing for robotics – a survey. In *Proc. of NM&A'02*, 2002.

[6] S. C. W. Ong, S. W. Png, D. Hsu, and W. S. Lee. Planning under uncertainty for robotic tasks with mixed observability. *Int. J. Rob. Res.*, 29(8):1053–1068, 2010.

[7] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proc. of IJCAI'03*, pages 1025–1032, 2003.

[8] S. Ross and B. Chaib-draa. AEMS: An anytime online search algorithm for approximate policy refinement in large POMDPs. In *Proc. of IJCAI'07*, pages 2592–2598, 2007.

[9] S. Ross, J. Pineau, and B. Chaib-draa. Theoretical analysis of heuristic search methods for online POMDPs. In *Proc. of NIPS'08*, pgs. 1233–1240, 2008.

[10] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online planning algorithms for POMDPs. *J. of Artificial Intelligence Research*, 32:663–704, 2008.

[11] D. Silver and J. Veness. Monte-carlo planning in large POMDPs. In *Proc. of NIPS'10*, pgs. 2164–2172, 2010.

[12] T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. In *Proc. of UAI'04*, pages 520–527, 2004.

[13] M. Spaan, T. Veiga, and P. Lima. Active cooperative perception in networked robotic systems using POMDPs. In *Proc. of IROS'10*, pgs. 4800–4805, 2010.

[14] J. Williams and S. Young. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21:393–422, 2007.

[15] Z. Zhang. and X. Chen. FHHOP: A factored heuristic online planning algorithm for POMDPs. In *Proc. of UAI'12*, 2012.