# Efficient Decision-Theoretic Target Localization

**Louis Dressel, Mykel J. Kochenderfer**
Aeronautics and Astronautics
Stanford University
{dressel,mykel}@stanford.edu

**Abstract**

Partially observable Markov decision processes (POMDPs) offer a principled approach to control under uncertainty. However, POMDP solvers generally require rewards to depend only on the state and action. This limitation is unsuitable for information-gathering problems, where rewards are more naturally expressed as functions of belief. In this work, we consider target localization, an information-gathering task where an agent takes actions leading to informative observations and a concentrated belief over possible target locations. By leveraging recent theoretical and algorithmic advances, we investigate offline and online solvers that incorporate belief-dependent rewards. We extend SARSOP—a state-of-the-art offline solver—to handle belief-dependent rewards, exploring different reward strategies and showing how they can be compactly represented. We present an improved lower bound that greatly speeds convergence. POMDP-lite, an online solver, is also evaluated in the context of information-gathering tasks. These solvers are applied to control a hexcopter UAV searching for a radio frequency source—a challenging real-world problem.

## 1 Introduction

In target localization, an agent searches a region for a hidden target. The target's location is unknown, so a belief is maintained. This probability distribution over possible target locations is updated using sensor measurements, a measurement model, and Bayes' rule. The goal of target localization is to efficiently reach a concentrated belief, which implies confidence in the target location estimate.

Target localization is an important robotic task with many real-world applications, like searching for GPS jammers (Dressel and Kochenderfer 2015), radio-tagged wildlife (Soriano, Caballero, and Ollero 2009), or rescue beacons (Hoffmann, Waslander, and Tomlin 2006). In these examples, robots were used because they cost less than human solutions and can be used in hazardous environments. However, these robots require control schemes that allow them to reason over actions and their effects on localization.

Partially observable Markov decision processes (POMDPs) offer a principled, decision-theoretic approach to optimal, closed-loop control under uncertainty (Kaelbling, Littman, and Cassandra 1998). Although solving POMDPs exactly is computationally intractable (Papadimitriou and Tsitsiklis 1987), recent algorithms generate approximately optimal policies with tight bounds on suboptimality, even for large problems. This progress makes POMDPs attractive for real robotic tasks involving uncertainty. Unfortunately, tasks like target localization or active sensing are ill-served by this approach because the traditional POMDP framework requires costs to depend on state and action only. Expressions of uncertainty, such as distribution entropy, depend instead on the belief.

Early efforts to overcome this limitation used surrogate rewards, belief compression, or greedy policies. These techniques often lack bounds on suboptimality. Fortunately, recent work shows that belief-dependent rewards can be used in the POMDP framework with modifications to existing offline solvers (Araya et al. 2010). However, issues like bounds and performance merit further investigation. Online solvers are another solution as they are more flexible in the types of rewards used. The recently proposed POMDP-lite method even uses information-theoretic techniques to encourage exploration in large POMDPs (Chen et al. 2016).

In this paper, we expand on this recent work on POMDP solvers. We modify SARSOP—one of the best offline POMDP solvers—to handle belief-dependent rewards, providing compact representations for these rewards without adding many actions or $\alpha$-vectors, in contrast to prior work. We also present an improved lower bound that significantly reduces computation time. The resulting offline solver is compared to POMDP-lite in an information gathering context. Results suggest the offline solver has better performance, though POMDP-lite requires far less computation and is more scalable. Finally, we implement our policies on an unmanned aerial vehicle (UAV) searching for a simulated GPS jammer—a challenging real-world problem.

The paper proceeds as follows. Section 2 covers background information on POMDPs, including POMDP solvers like POMDP-lite and target localization with POMDPs. Sections 3 and 4 build on previous work and show how point-based offline solvers can be modified for belief-dependent rewards, with an emphasis on improving computational tractability. The resulting offline solver is compared to POMDP-lite on two example problems in Section 5. Section 6 presents the jammer hunting problem.

## 2 Background

### 2.1 POMDPs

A POMDP consists of a state space $\mathcal{S}$, action space $\mathcal{A}$, observation space $\mathcal{O}$, transition function $T$, observation function $Z$, reward function $R$, and discount factor $\gamma$. At each time step $t$, the agent takes action $a \in \mathcal{A}$ from state $s \in \mathcal{S}$, arriving in some new state $s' \in \mathcal{S}$ with probability $p(s' \mid s, a) = T(s, a, s')$. The agent also receives reward $r_t = R(s, a)$. The agent's goal is to maximize the expected discounted reward $E[\sum_{t=0}^{\infty} r_t \gamma^{-t}]$, where $\gamma \in [0, 1)$ ensures a finite sum.

If the agent always knows its state, the problem is fully observable and simply called a Markov decision process (MDP). In an MDP, a policy $\pi$ maps states to actions. The expected discounted reward starting from state $s$ and following policy $\pi$ is called the value of state $s$ and is denoted $V^{\pi}(s)$. The goal is to find an optimal policy $\pi^*$ that maximizes the value from every state. This optimal value function $V^*$ can be found by iteratively applying the Bellman update to convergence:

$$V(s) = \max_a R(s, a) + \gamma \sum_{s'} T(s, a, s')V(s'). \quad (1)$$

In a POMDP, the state is not fully observable. Instead, a noisy observation $o \in \mathcal{O}$ is sampled according to observation function $Z(a, s, o) = p(o \mid a, s)$. These noisy observations are combined with a prior to maintain a belief $b$, or probability distribution over the states. After taking action $a$ from $b$ and observing $o$, a new belief $b'$ can be generated with Bayes' rule. The solution to a POMDP is a mapping from belief to action. The Bellman update for POMDPs is

$$V(b) = \max_a \rho(b, a) + \gamma \int_{b'} \tau(b, a, o, b')V(b')db'. \quad (2)$$

Equation (2) is similar to Equation (1), where the states are now beliefs. The transition function $\tau$ describes the probability of transitioning from $b$ to $b'$ given action $a$ and observation $o$. It can be rewritten in terms of $T$ and $Z$. The belief-dependent reward function $\rho(b, a)$ is rewritten using a state-based reward function $R(s, a)$: $\rho(b, a) = \sum_s R(s, a)b(s)$. Because $\rho$ is expressed as an expectation of state-based reward conditioned on belief, value functions generated with Equation (2) are piecewise linear and convex (PWLC) over belief. Therefore, the optimal value function can be approximated arbitrarily well with a set of linear functions (Smallwood and Sondik 1973). These linear functions are called $\alpha$-vectors. The value function is the upper surface of a set $\Gamma$ of $\alpha$-vectors: $V(b) = \max_{\alpha \in \Gamma} \alpha \cdot b$.

### 2.2 Offline Solvers

Offline POMDP solvers typically update $\Gamma$ until the resulting value function closely approximates the optimal. These approaches require a state-dependent reward to ensure the value function is PWLC. The updating uses point-based value iteration, where value iteration is performed over a set of points in belief space (Pineau et al. 2003). Belief space is infinite, so these methods only consider the reachable space, or set of beliefs that can be reached from an initial belief. A search tree is created from this initial belief, and the transition and observation functions are used to generate new belief nodes. A benefit of offline solvers is that all solving happens before execution; while executing the policy, an agent simply queries the offline results.

SARSOP is an offline, point-based solver that reduces computation time by estimating the reachable space under optimal policies (Kurniawati, Hsu, and Lee 2008). SARSOP maintains upper and lower bounds on the value function and uses heuristics to predict the value of new beliefs. These techniques reduce the size of the search tree.

### 2.3 POMDP-lite

Online POMDP solvers typically form a search tree whose root is the current belief during execution. This tree is smaller than the tree starting from the initial belief, making online solvers more scalable. However, the agent is forced to recalculate this tree as the belief is updated, requiring computation that might be infeasible on robotic hardware.

POMDP-lite is a recently proposed online solver that has outperformed other online methods (Chen et al. 2016), making it a useful benchmark. POMDP-lite leverages partial observability; in many problems, part of the state is observable. This is often true in target localization, where an agent might know its own position while the target remains hidden. In that case, it is computationally beneficial to decompose the monolithic state space $\mathcal{S}$ from Section 2.1 into the Cartesian product of observable state space $X$ and unobservable state space $\Theta$.

POMDP-lite reduces the POMDP to an MDP with state space $X$. Values that depend on the unobserved state are treated as expectations over the unobserved state space, with weights dictated by the current belief over unobserved states. For example, the transition function becomes

$$T_b(b, x, a, x') = \sum_{\theta \in \Theta} b(\theta)P(x' \mid x, \theta, a),$$

and the reward function becomes

$$R_b(b, x, a) = \sum_{\theta \in \Theta} b(\theta)R(x, \theta, a). \quad (3)$$

A crucial part of POMDP-lite is that it encourages exploration by augmenting the reward from Equation (3) with an exploration reward $R_e$. This reward is the expected $\ell_1$ divergence between the current belief $b$ and the next belief $b'$:

$$R_e(b, x, a) = \sum_{\theta, x', o} P(o \mid \theta, x', a)P(x' \mid x, \theta, a)b(\theta)\left\|b' - b\right\|_1. \quad (4)$$

At the current belief $b$, the MDP has state space $X$, transition function $T_{\text{MDP}} = T_b$, and reward function $R_{\text{MDP}} = R_b + \lambda R_e$, where $\lambda$ is a scale factor encoding the preference of information rewards over original rewards. This MDP is solved and the best action from the observed state is taken. An observation is received, the belief is updated, and a new MDP is solved. If the problem is small enough, the MDP can be solved exactly with value iteration. Otherwise, an online method such as UCT can be used.

POMDP-lite has been used to solve POMDPs with large state spaces but has not yet been used in a purely information-gathering context. In this work, we compare it to offline methods modified for belief-dependent rewards.

## 2.4 Prior POMDP Localization Approaches

Surrogate rewards are a common approach to circumventing the limitation of state-dependent rewards (Hsu, Lee, and Rong 2008; Dressel and Kochenderfer 2015). Surrogate rewards "trick" the agent into desired behavior with a state-based reward that requires solving the localization problem. In localization tasks, a surrogate reward might be given if the agent reaches the target's location. Although this forces the agent to find the target, the agent is also incentivized to stay near the target, even if better measurements can be made farther away.

Another POMDP localization technique is to augment the state space with a compressed version of the belief (Roy et al. 1999; Roy, Gordon, and Thrun 2005; Thrun, Burgard, and Fox 2005). The compressed belief commonly consists of the belief entropy and the index of the maximum belief. The dynamics of transitioning between these augmented states can be learned through Monte Carlo simulations (Thrun, Burgard, and Fox 2005; Dressel and Kochenderfer 2015). Although learning these dynamics allows for non-greedy planning to minimize entropy, compressing different beliefs might lead to the same compressed belief. This loss of information can lead to suboptimal control.

Another common approach is to abandon long-term planning and focus instead on the next time-step. In localization tasks, these greedy approaches guide agents to take the control action leading to lowest expected entropy after a single step (Hoffmann, Waslander, and Tomlin 2006). Entropy is a measure of spread in a distribution (a uniform distribution maximizes entropy), making it a good objective function. However, greedy behavior can be suboptimal as the agent trades long-term optimality for short-term gain.

## 3 Belief-Dependent Rewards

As explained in Section 2.2, POMDP solvers like SARSOP rely on state-dependent rewards to maintain a PWLC value function that can be approximated with $\alpha$-vectors. A key insight by Araya et al. was that so long as $\rho(b,a)$ was itself PWLC, then value functions generated with Equation (2) would also be PWLC (2010). The term "$\rho$POMDP" refers to POMDPs with PWLC belief-dependent rewards.

Another framework is the POMDP with information rewards (POMDP-IR), which adds "guess" actions performed simultaneously with normal actions (Spaan, Veiga, and Lima 2014). There is one guess action per state, each yielding a state-based reward if it corresponds to the true state. Although these actions greatly increase the action space, they decompose nicely out of the Bellman update because they do not affect the system dynamics. It has actually been shown that a POMDP-IR is equivalent to a $\rho$POMDP (Satsangi, Whiteson, and Spaan 2015).

Here, we examine three PWLC belief-dependent reward functions. None rely on entropy—a common uncertainty measure—because it is not piecewise linear. We could generate a PWLC approximation with tangential hyperplanes at selected points, but generating a good, dense approximation before solving can lead to an enormous set of hyperplanes. An alternative is to only generate hyperplanes at new nodes in the belief tree, but this requires extra computation at each new node.

### 3.1 Max-Norm Reward

An alternative reward is the $\ell_\infty$-norm, or max-norm, proposed by Eck and Soh in the context of $\rho$POMDPs (2012). This PWLC function can be represented exactly with the standard basis of $\mathbb{R}^{|\mathcal{S}|}$:

$$\rho(b,a) = \max_{\alpha \in \Gamma_\rho} \alpha \cdot b, \quad \Gamma_\rho = \left\{ e_1, ..., e_{|\mathcal{S}|} \right\}. \tag{5}$$

The ability to compactly and exactly represent the max-norm reward is a great advantage over negative entropy. Surprisingly, a sparse approximation of negative entropy can perform worse than a max-norm reward, even when evaluated by the expected sum of negative entropy (Araya 2013). The max-norm is also more intuitive—a max-norm of 0.6 suggests there is a 60% chance the agent is in the most likely state, whereas a distribution entropy of 2 nats is less useful to a human evaluator.

### 3.2 Threshold Reward

A disadvantage of the max-norm is that the agent always receives some reward, even at uniform beliefs. Sometimes, we want an agent to reach a highly concentrated belief as quickly as possible, but the agent might be driven by the max-norm reward to collect rewards at less-concentrated beliefs in the near-term. Spaan, Veiga, and Lima suggested thresholded rewards in the POMDP-IR framework, but this requires an additional guess action per state (2014). Our $\rho$POMDP version does not:

$$\rho(b,a) = \max\left( \frac{\|b\|_\infty - c_\rho}{1 - c_\rho}, 0 \right), \tag{6}$$

where $c_\rho$ is the max-norm cutoff. A belief max-norm below $c_\rho$ induces no reward. Above $c_\rho$, the reward increases linearly until it reaches a maximum value of 1. An exact representation of the threshold reward only needs one hyperplane per state and an additional $\vec{0}$ hyperplane.

### 3.3 Guess Reward

We examine a final reward function introduced in the POMDP-IR literature (Spaan, Veiga, and Lima 2014). In a POMDP-IR, the agent guesses the true system state at each time step. The agent is rewarded 1 for guessing correctly and 0 otherwise. This reward function is equivalent to the max-norm, because the expected reward of the guess equals the belief max-norm (Satsangi, Whiteson, and Spaan 2015). In one variant, the agent can guess *instead of* taking a normal action. The agent's action space is augmented with a single guess action independent of the problem dynamics; it is assumed the state with highest belief density is chosen for the guess. This guess reward function can be represented as

$$\rho(b,a) = \mathbb{1}\{a = \text{guess}\} \left( \max_{\alpha \in \Gamma_\rho} \alpha \cdot b \right), \tag{7}$$

where $\Gamma_\rho$ is defined in Equation (5) and $\mathbb{1}\{x\}$ is the indicator function that returns 1 if $x$ is true. Purely belief-dependent rewards require an external termination condition, like an entropy threshold (Eck and Soh 2012). The guess action forces the agent to reason about the cost of acquiring new information, removing the need for external termination conditions.

### 3.4 Action Rewards

Often there is a cost to performing sensing actions—they might take longer than other actions or use more resources. Adding an action-dependent reward $R(a)$ maintains the PWLC property.

## 4 SARISA

Here we incorporate the PWLC rewards from the previous section into offline, point-based solvers. Specifically, we modify SARSOP and call the resulting algorithm SARSOP with information-seeking actions (SARISA).

### 4.1 Backup

The backup operation uses the Bellman update to improve the value function at belief $b$ using information at the child beliefs of $b$. We denote $\alpha_{a,o} \in \Gamma$ as the maximizing $\alpha$-vector at the belief reached when taking $a$ from $b$ and observing $o$. Then, a set of $\alpha$-vectors is created for each action $a$, where $\alpha_a$ describes the $\alpha$-vector created for action $a$. The entry in $\alpha_a$ for state $s$ is updated:

$$\begin{aligned}\alpha_a(s) &= R(s,a) + \gamma \sum_{o,s'} T(s,a,s')Z(a,s,o)\alpha_{a,o}(s') \\ &= \alpha_b(s) + R(a) + \gamma \sum_{o,s'} T(s,a,s')Z(a,s,o)\alpha_{a,o}(s')\end{aligned} \tag{8}$$

where $\alpha_b = \operatorname{argmax}_{\alpha \in \Gamma_\rho} \alpha \cdot b$. If we use the max-norm reward, the update is:

$$\begin{aligned}\alpha_a(s) = \mathbb{1}\{s = \operatorname*{argmax}_{s'} b(s')\} + R(a) + \\ \gamma \sum_{o,s'} T(s,a,s')Z(a,s,o)\alpha_{a,o}(s').\end{aligned} \tag{9}$$

A similar update can be written for the threshold reward:

$$\begin{aligned}\alpha_a(s) = \frac{\mathbb{1}\{b(s^*) > c_\rho\}}{1 - c_\rho} \Big[ \mathbb{1}\{s = s^*\} - c_\rho \Big] + R(a) + \\ \gamma \sum_{o,s'} T(s,a,s')Z(a,s,o)\alpha_{a,o}(s'),\end{aligned} \tag{10}$$

where $s^* = \operatorname{argmax}_s b(s)$. Equations (9) and (10) represent a computational benefit over the traditional $\rho$POMDP backup shown in Equation (8)—we do not need to maintain a set $\Gamma_\rho$ or compute $\alpha_b$ at each backup, a previous criticism of $\rho$POMDPs (Satsangi, Whiteson, and Spaan 2015).

### 4.2 Upper Bound

In SARSOP, the upper bound is represented with a set of belief-value pairs, and the sawtooth approximation (Shani, Pineau, and Kaplow 2013) is used to interpolate for values at new beliefs. The fast informed bound (FIB) approximation generates this upper bound. FIB switches max and sum operators in the Bellman update and is an upper bound on the value function (Hauskrecht 2000). Here, we derive FIB for rewards depending on belief and action. FIB is initialized with a set $\Gamma$ of $\alpha$-vectors, with one $\alpha$-vector $\alpha_a$ per action $a$, each of which is usually initialized to zeros. We start with a variant of the Bellman update:

$$\begin{aligned}V(b) &= \max_a \Big[ \rho(b,a) + \\ &\qquad \gamma \sum_o \max_{\alpha \in \Gamma} \sum_{s,s'} b(s) p(s', o \mid s, a) \alpha(s') \Big] \\ &= \max_a \Big[ \max_{\alpha \in \Gamma_\rho} \sum_s b(s)\alpha(s) + \sum_s b(s) R(a) + \\ &\qquad \gamma \sum_o \max_{\alpha \in \Gamma} \sum_{s,s'} b(s) p(s', o \mid s, a) \alpha(s') \Big] \\ &\le \max_a \sum_s b(s) \Big[ \max_{\alpha \in \Gamma_\rho} \alpha(s) + R(a) + \\ &\qquad \gamma \sum_o \max_{\alpha \in \Gamma} \sum_{s'} p(s', o \mid s, a) \alpha(s') \Big].\end{aligned}$$

We assume the PWLC belief-dependent reward is uniform at the corners of the belief simplex, as is the case with negative entropy and the max-norm. We denote this corner reward as $r_{b^*}$. Because we assume no state-dependent rewards, every element in a specific $\alpha$-vector will have the same value. For $\alpha$-vector $\alpha$, we denote this constant value $\alpha_c$. This term does not rely on $s'$ or $o$:

$$\begin{aligned}V(b) &\le \max_a \sum_s b(s) \Big[ r_{b^*} + R(a) + \\ &\qquad \gamma \max_{\alpha \in \Gamma} \alpha_c \sum_{o,s'} p(s', o \mid s, a) \Big] \\ &= \max_a \sum_s b(s) \underbrace{\Big[ r_{b^*} + R(a) + \gamma \max_{\alpha \in \Gamma} \alpha_c \Big]}_{\alpha_a^{(k+1)}(s)}.\end{aligned}$$

Each $\alpha_a$ can now be updated iteratively, independently of belief. The element corresponding to state $s$ is updated in step $k+1$ using the $\alpha$-vectors from step $k$: $\alpha_a^{(k+1)}(s) = r_{b^*} + R(a) + \gamma \max_{\alpha^{(k)} \in \Gamma} \alpha_c^{(k)}$. This iteration can be represented as a geometric sum because the $\alpha$-vector maximizing $\alpha_c^{(k)}$ always belongs to the action with highest reward—so, every element in $\alpha_a$ converges to $R(a) + (r_{b^*} + \gamma \max_a R(a))/(1-\gamma)$. Because every element in an $\alpha$-vector is the same, the $\alpha$-vector belonging to the highest reward action dominates at any belief—and each element has

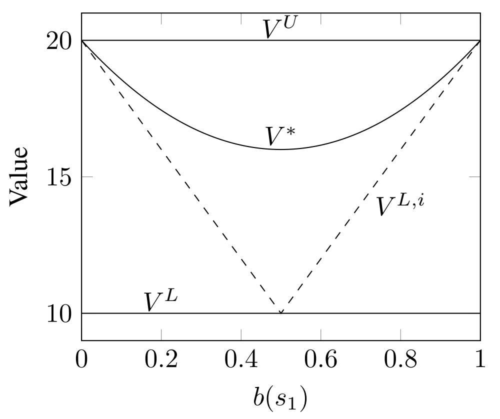

Figure 1: Example two-state problem with the max-norm reward, $\gamma = 0.95$, and no action costs. The true value $V^*$ is bounded by upper and lower bounds $V^U$ and $V^L$. The improved bound $V^{L,i}$ is much tighter than $V^L$.

the value

$$\frac{r_{b^*} + \max_a R(a)}{1-\gamma}. \tag{11}$$

This dominant $\alpha$-vector is used to generate a set of belief-value pairs, initializing the upper bound. The result in Equation (11) is easy to compute and requires no iteration. However, this FIB-generated upper bound is equivalent to a naïve upper bound that simply discounts the maximum possible reward to infinity. This result is unsurprising because the FIB iterations include no notion of belief and should not be able to capture the effect of belief-dependent rewards. However, the result is shown here for completeness. Improved upper bounds are an area of future research.

## 4.3 Lower Bound

The lower bound maintained by SARSOP is the set $\Gamma$ of $\alpha$-vectors representing the value function. This bound is initialized with one $\alpha$-vector per action using a blind policy (Hauskrecht 1997). In a POMDP with belief and action-dependent rewards, the worst greedy reward is equal to $r_{b_w} + \max_a R(a)$, where $r_{b_w}$ is the worst belief-dependent reward, typically achieved when the belief is uniform. As with the upper bound, the resulting $\alpha$-vectors will be dominated by the $\alpha$-vector corresponding to the action with the highest reward, where every element is

$$\frac{r_{b_w} + \max_a R(a)}{1-\gamma}. \tag{12}$$

The lower bound can be initialized to a single $\alpha$-vector belonging to the highest reward action, with each element equal to the value shown in Equation (12), but this bound is very loose.

We can derive a tighter lower bound for the max-norm reward if the agent has an action that is guaranteed not to change the belief. The belief max-norm remains unchanged after applying this action, and the infinitely discounted max-norm is a lower bound on the value at the belief. Figure 1 shows the improved bound, which directs exploration and helps convergence.

Localization of a stationary target always satisfies this assumption. The agent only needs a non-observing action that returns a null observation—common in target localization, where agents often have the option to move or make a measurement. If the agent is always sensing, we can simply add an action that discards the observation. This action only exists to guide exploration during solving, and it is unlikely to be the optimal action selected during execution.

If non-zero, the non-observing action's reward can be included in the infinite discounting. The improved bound can be expressed compactly with one $\alpha$-vector per state, each corresponding to the non-observing action. The same bound holds for the guess reward function—the guess action takes the role of the non-observing action. A similar bound can be derived for the threshold reward function. An additional $\vec{0}$ $\alpha$-vector represents the no reward belief region.

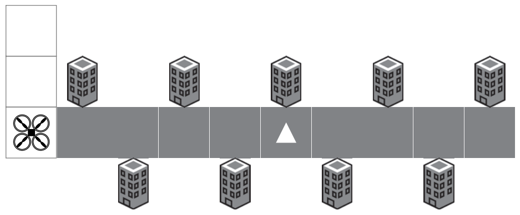

Figure 2: The LazyScout problem. The UAV must find a radio beacon (white triangle) located between some buildings. Grey cells indicate possible locations of the hidden beacon. The UAV can climb above the buildings to receive a perfect observation.

## 5 Example Problems

Before using SARISA on a larger problem, we validate it against POMDP-lite on toy examples. We also compare performance against surrogate and greedy methods.

## 5.1 LazyScout

We present LazyScout, a toy localization task showing the possible suboptimality of greedy entropy minimization and surrogate rewards. A UAV equipped with a range sensor seeks a radio beacon located between buildings. The UAV knows its own location, so each range measurement implies a beacon location. When the UAV travels between buildings, its observations are degraded by clutter and multipath. It might observe the grid cell containing the beacon, the cell before, or the cell after, each with equal probability. Alternatively, the UAV can climb above the buildings. Climbing takes two time steps and no measurements can be made while climbing. However, once above the buildings, the UAV observes the true beacon location. Figure 2 is a graphical representation of LazyScout.

The optimal action is to climb above the buildings, which ensures localization in two steps. However, both greedy entropy minimization and a surrogate reward strategy act suboptimally. Greedy entropy minimization fails because the

Table 1: Reward comparison for LazyScout

| reward structure | first action | reward | steps to localize | solve time (s) |
|---|---|---|---|---|
| surrogate | buildings | 17.521 | 4.3 | 0.86 |
| greedy | buildings | 17.521 | 4.3 | - |
| SARISA ($\ell_\infty$) | climb | 18.266 | 2.0 | 0.06 |
| POMDP-lite | climb | 18.266 | 2.0 | - |

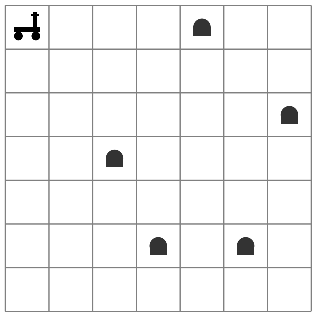
Figure 3: Grid used for rock problems: five rocks, $\gamma = 0.95$, rover starts in upper left.

noisy measurement received through the buildings is "better" than receiving no measurement while climbing. If we define a surrogate, state-dependent reward function that rewards the UAV for reaching the beacon location, the UAV will try to stay near the estimated location of the beacon. The extra time required to climb and descend is not worth it—the UAV can piece together enough noisy measurements as it moves through the buildings and closer to the beacon. Localization might take longer, but the time to physically reach the beacon is reduced.

Simulation results comparing surrogate rewards, greedy rewards, SARISA with the max-norm reward function, and POMDP-lite are shown in Table 1. SARISA and POMDP-lite lead to the correct first action, cutting localization time in half (here localization means concentrating belief to a single cell). SARISA's bounds converge to 18.266, the theoretically correct initial value when evaluating with the max-norm reward and $\gamma = 0.95$.

The SARISA solver used the improved lower bound, leading to a solve time of 0.06 s. When this improved bound was not used, convergence took 0.99 s, nearly a factor of 17 longer. The improved bound drastically reduces the number of backups necessary: the improved version only used 237 backups while the unimproved version needed 1,961.

## 5.2 RockSample and RockDiagnosis

RockSample is commonly used to test the effectiveness of POMDP solvers (Smith and Simmons 2004). A rover moves in a square grid and samples rocks that exist at known locations and might have scientific value. From a given grid cell, the rover can move to a non-diagonal neighbor cell, use a laser to scan any rock, or sample a rock occupying the same cell. Scanning a rock provides a noisy measurement of its value. Sensor noise increases with the rover's distance from the rock. The rover is rewarded for sampling a valuable rock and penalized for sampling a worthless one.

Table 2: Reward comparison for RockSample, when evaluated by max-norm reward.

| policy | solve time (s) | reward |
|---|---|---|
| surrogate | 34 | 7.6 |
| SARISA ($\ell_\infty$) | 7200 | 12.7 |
| POMDP-lite | - | 10.7 |
| reach | - | 8.1 |
| random | - | 8.4 |

The goal in a modified version of RockSample called RockDiagnosis is only to determine whether each rock is valuable (Araya 2013). The rover has no sample action—instead, it maneuvers and scans to learn the worth of each rock. The original RockSample can be seen as RockDiagnosis with a surrogate reward, where the sample costs exist only to encourage this learning. A RockDiagnosis agent should determine the rock states more quickly.

We first solved the RockDiagnosis problem shown in Figure 3 with SARISA and the max-norm reward function, comparing it to a "surrogate" policy solved on the RockSample model with SARSOP, a random action policy, and a "reach" policy that moved the agent in the shortest path to each rock, making a perfect observation at each. We also used POMDP-lite, solving the MDP exactly at each step with value iteration. We also use the original information reward shown in Equation (4), instead of experimenting with other reward functions, such as the expectation of $\left\|b' - b\right\|_\infty$ or simply the expectation of $\left\|b'\right\|_\infty$. Preliminary results suggest performance does not vary much with these reward variants. Therefore we just use the original, leaving variants as a subject for future work.

Table 2 shows the mean sum of discounted max-norm rewards during 2000 simulations of 100 steps for each policy. SARISA yields the highest reward, which is unsurprising because its reward function matches the evaluation reward function. However, the result is not insignificant. An early attempt at solving RockDiagnosis of the same size used a modified version of Perseus (Spaan and Vlassis 2005) and could not outperform the random policy (Araya 2013), suggesting our work is an improvement over early POMDP solvers incorporating belief-dependent rewards.

A notable result is the slow convergence of SARISA—after 7200 s, the lower and upper bounds were 12.3 and 14.4. In contrast, the RockSample policy bounds converged to a width of 0.001 in just 34 s. One way to improve convergence is to find tighter starting bounds, a subject of future research. Although POMDP-lite underperforms SARISA, it is extremely efficient; it requires no offline solving and actions are selected at each belief in under 0.02 s.

We also explore the effect of other reward functions. Suppose we want the rover to be 95%-confident—according to its model—in a rock configuration, as fast as possible. We might use the threshold reward from Equation (6) with a cutoff $c_\rho = 0.9$. Because beliefs with max-norm below 0.9

Table 3: Reward comparison for RockSample, when evaluated by threshold reward.

| policy | solve time (s) | reward |
|---|---|---|
| SARISA (thresh 0.9) | 7200 | 6.1 |
| SARISA (max-norm) | 7200 | 4.2 |
| POMDP-lite | - | 7.1 |

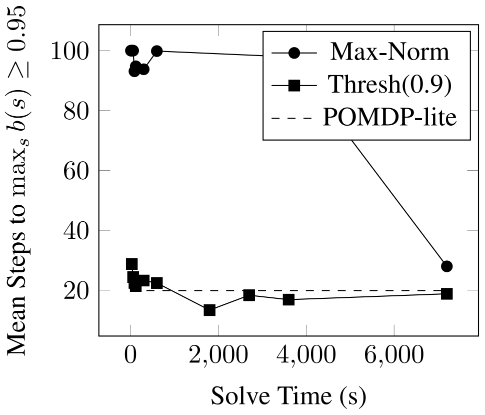

Figure 4: Average steps to reach a highly concentrated belief. If a trajectory did not reach the desired max-norm, the worst-case value of 100 was assigned.

yield no reward, the agent is encouraged to reach highly concentrated beliefs more quickly. Figure 4 shows how quickly policies solved with max-norm and threshold rewards reach a belief with a max-norm of 0.95. The max-norm policies almost always failed to reach the desired confidence if they had been solved for less than an hour. After solving for two hours, the performance was much better, probably because SARISA had time to reach further down the belief tree to more highly-concentrated beliefs. In contrast, threshold policies solved for even a short amount of time reach the desired confidence quickly. POMDP-lite (with the reward from Equation (4)) also reached highly concentrated beliefs quickly.

Policies were evaluated using the threshold reward. Mean discounted rewards are shown in Table 3. As expected, the threshold policy outperforms the max-norm policy because it was trained on the evaluation reward function. However, POMDP-lite outperforms the threshold SARISA policy. This is probably due to SARISA's unconverged bounds, which were 4.6 and 13.7 after 7200 s. These bounds are much wider than in the max-norm case, most likely because rewards only occur deep in the search tree at concentrated beliefs. The improved lower bound also assigns no value to beliefs below the threshold max-norm, so the lower bound is probably loose, leading to poor convergence. As a result, POMDP-lite might be a better choice when we desire threshold-like rewards. Still, the improved lower bound significantly helps SARISA's performance. As Figure 5 shows, the improved lower bound is higher after 30 seconds of solving than the unimproved bound after two hours.

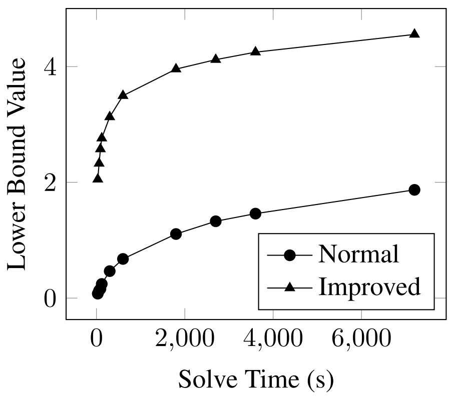

Figure 5: Lower bound on RockDiagnosis when using threshold reward with cutoff of 0.9. The improved lower bound improves convergence.

## 6 Jammer Localization

GPS is a critical part of air travel, but it is susceptible to RF interference and jamming (Geyer and Frazier 1999). We consider the problem of using a multirotor UAV to hunt a GPS jammer, modeled as a WiFi router. The search area is modeled as an $11 \times 11$ grid with $10\,\mathrm{m} \times 10\,\mathrm{m}$ cells. A state $s$ consists of the known UAV position $(x_v, y_v)$ and unknown jammer position $(x_j, y_j)$—we assume the UAV has an alternate (non-GPS) navigation system. At each step, the UAV can deterministically move to a neighboring grid cell, rotate in place, or hover (terminate the search). The UAV is constrained to a constant altitude.

A multirotor UAV armed with a directional antenna can estimate bearing to an RF source by rotating in place (Graefenstein et al. 2009). We model the noise as zero-mean Gaussian, consistent with results from the literature (Perkins et al. 2015). Noise standard deviation is 13° at most ranges, but it increases to roughly 40° if the jammer and UAV are in adjacent cells. To reduce computation, the angular space is split into 10° bins. An additional null measurement is received when the UAV does not rotate, yielding 37 possible observations.

We want the agent to reason about when to stop making measurements, so we use a guess reward: $\rho(b,a) = \mathbb{1}\{a = \text{hover}\}\|b\|_\infty + \lambda R(a)$, where $R(a)$ is the action reward and $\lambda$ is a scale factor relating action and information rewards. The sensing reward $R(a)$ depends roughly on the time to complete an action; $R(a) = -1$ for moving in a cardinal direction, $R(a) = -\sqrt{2}$ for moving diagonally, and $R(a) = -3$ for rotating to measure bearing. A similar surrogate reward replaces the max-norm reward with 1 if the UAV hovers over the jammer.

We varied $\lambda$ and solved each model for 12 hours with SARISA. We ran 1210 simulations to completion, with the jammer at random locations and the UAV starting at the center. We measured the time to make a decision (hover) and whether the agent's guess—the state with highest probability—matched the true state. We compared our policies to a greedy policy that moves the UAV to the

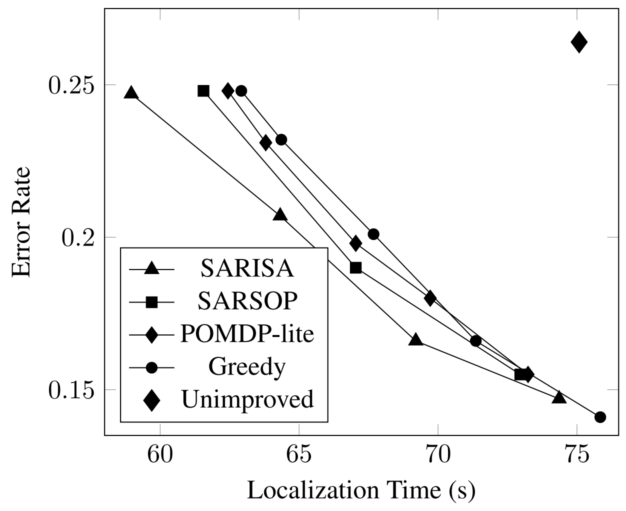

Figure 6: Simulation-produced Pareto curve showing the effectiveness of belief-dependent rewards in the jammer-hunting problem.

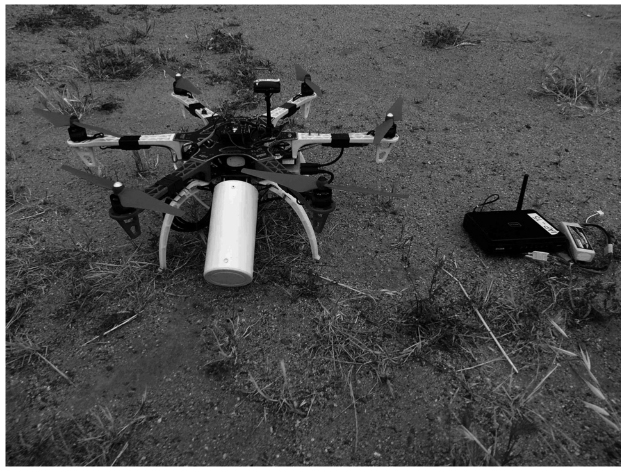

Figure 7: The hexcopter UAV used for flight tests. The white cylinder in front of the UAV is a directional Yagi antenna. The WiFi router to the right of the vehicle serves as the simulated GPS jammer.

cell that, after rotation, yields the lowest expected entropy. These greedy policies were stopped at different cutoff max-norm values. As seen in Figure 6, SARISA policies achieve slightly less error in less time. However, at lower error rates than shown, POMDP methods underperform the greedy method, probably because this requires reaching further down the search tree.

We also implemented POMDP-lite using the information reward $R_e$ from Equation (4). Just as we did with SARISA and SARSOP, we scaled the expected POMDP reward by a factor $\lambda$, leading to the MDP reward $R_{\text{MDP}} = \lambda R_b + R_e$. POMDP-lite outperforms the greedy method but underperforms SARISA and SARSOP. However, POMDP-lite's relative performance seems to improve at lower error rates, where the offline solvers have trouble reaching deep nodes.

Although SARISA seems to perform best, the methods all have comparable performance. It is possible the greedy method is close to the true optimal policy for this particular problem. The more striking result is the effect of the improved lower bound. Solving for $\lambda = 2$ yielded bounds of (45.8, 91.6) for the improved bound and (7.6, 92.6) for the unimproved bound. This inferior bound limited the depth of search tree exploration, and highly concentrated beliefs were not reached. As Figure 6 shows, only a single value of $\lambda$ yielded a comparable error rate, and this point is Pareto dominated by all other solvers. In this problem, the improved lower bound enables our use of belief-dependent rewards and a point-based POMDP solver. Another important insight arises from the surrogate's bounds: (44.1, 86.4). These are similar to SARISA's, suggesting information-gathering problems are inherently difficult, even if we wrap the belief-dependent reward into a similar state-dependent reward.

We implemented our policies on a DJI Flamewheel F550 hexcopter searching for a WiFi router (simulated GPS jammer). The UAV carries a directional Yagi antenna and can be seen in Figure 7. The UAV also carries a small ODROID computer to execute SARISA and POMDP-lite policies. The ODROID passes the selected action to the Pixhawk flight controller over serial. The flight controller executes the action. If the action is to rotate, the ODROID estimates bearing from antenna measurements, updates its belief, and picks a new action. The ability to run these policies on a real UAV in a plausible target localization task is a promising step for decision-theoretic target localization.

## 7 Conclusion

Previous work showed how to incorporate belief-dependent rewards into offline POMDP solvers. We build on this prior work in the context of target localization with the goal of improving computational efficiency. We examine different belief-dependent rewards and how they induce different information-gathering behavior. We show that the backup operations of these rewards do not need a set $\Gamma_\rho$ of linear functions, leading to reduced computation during backup—the core, inner loop of POMDP solvers. We provide an improved lower bound that greatly improves performance, allowing us to tackle larger problems like jammer-hunting.

We compare our resulting offline solver, named SARISA, to POMDP-lite, a powerful online solver. SARISA outperforms POMDP-lite on problems where our improved lower bound greatly improves convergence—like with the max-norm and guess reward functions. When the starting bounds are loose, as with the threshold reward, POMDP-lite begins to outperform SARISA. POMDP-lite is also more scalable. Improving convergence and guaranteeing near-optimality requires further research. A promising avenue might be myopic policy bounds (Lauri et al. 2016).

The SARISA solver is open-source and available at https://github.com/sisl/SARISA.jl.

**Acknowledgments** This work was supported by NSF grant DGE-114747 and the Hellman Fellows Fund.

## References

Araya, M.; Buffet, O.; Thomas, V.; and Charpillet, F. 2010. A POMDP extension with belief-dependent rewards. In *Advances in Neural Information Processing Systems (NIPS)*, 64–72.

Araya, M. 2013. *Des algorithmes presque optimaux pour les problèmes de décision séquentielle à des fins de collecte d'information*. Ph.D. Dissertation, Université de Lorraine.

Chen, M.; Frazzoli, E.; Hsu, D.; and Lee, W. S. 2016. POMDP-lite for robust robot planning under uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, 5427–5433.

Dressel, L., and Kochenderfer, M. J. 2015. Signal source localization using partially observable Markov decision processes. In *AIAA Infotech@Aerospace Conference*.

Eck, A., and Soh, L.-K. 2012. Evaluating POMDP rewards for active perception. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 1221–1222. International Foundation for Autonomous Agents and Multiagent Systems.

Geyer, M., and Frazier, R. 1999. FAA GPS RFI mitigation program. In *International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS)*, 107–114.

Graefenstein, J.; Albert, A.; Biber, P.; and Schilling, A. 2009. Wireless node localization based on RSSI using a rotating antenna on a mobile robot. In *Workshop on Positioning, Navigation and Communication (WPNC)*, 253–259. IEEE.

Hauskrecht, M. 1997. Incremental methods for computing bounds in partially observable Markov decision processes. In *AAAI Conference on Artificial Intelligence (AAAI)*, 734–739.

Hauskrecht, M. 2000. Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research* 33–94.

Hoffmann, G. M.; Waslander, S. L.; and Tomlin, C. J. 2006. Distributed cooperative search using information-theoretic costs for particle filters, with quadrotor applications. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, 21–24.

Hsu, D.; Lee, W. S.; and Rong, N. 2008. A point-based POMDP planner for target tracking. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2644–2650. IEEE.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1):99–134.

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

Lauri, M.; Atanasov, N.; Pappas, G. J.; and Ritala, R. 2016. Myopic policy bounds for information acquisition POMDPs. *arXiv preprint arXiv:1601.07279*.

Papadimitriou, C. H., and Tsitsiklis, J. N. 1987. The complexity of Markov decision processes. *Mathematics of Operations Research* 12(3):441–450.

Perkins, A.; Dressel, L.; Lo, S.; and Enge, P. 2015. Antenna characterization for UAV based GPS jammer localization. In *International Technical Meeting of The Satellite Division of the Institute of Navigation*, volume 2015. Tampa, Florida.

Pineau, J.; Gordon, G.; Thrun, S.; et al. 2003. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 3, 1025–1032.

Roy, N.; Burgard, W.; Fox, D.; and Thrun, S. 1999. Coastal navigation-mobile robot navigation with uncertainty in dynamic environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, 35–40.

Roy, N.; Gordon, G. J.; and Thrun, S. 2005. Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research* 23:1–40.

Satsangi, Y.; Whiteson, S.; and Spaan, M. T. J. 2015. An analysis of piecewise-linear and convex value functions for active perception POMDPs. Technical Report IAS-UVA-15-01, Informatics Institute, University of Amsterdam.

Shani, G.; Pineau, J.; and Kaplow, R. 2013. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems* 27(1):1–51.

Smallwood, R. D., and Sondik, E. J. 1973. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research* 21(5):1071–1088.

Smith, T., and Simmons, R. 2004. Heuristic search value iteration for POMDPs. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 520–527. AUAI Press.

Soriano, P.; Caballero, F.; and Ollero, A. 2009. RF-based particle filter localization for wildlife tracking by using an UAV. In *International Symposium of Robotics*.

Spaan, M. T., and Vlassis, N. 2005. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research* 195–220.

Spaan, M. T.; Veiga, T. S.; and Lima, P. U. 2014. Decision-theoretic planning under uncertainty with information rewards for active cooperative perception. *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* 1–29.

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