Efficient planning under uncertainty with incremental refinement

Juan Carlos Saborío[*]
Institute of Computer Science
University of Osnabrück
Wachsbleiche 27, Osnabrück, Germany

Joachim Hertzberg[†]
German Research Center for Artificial Intelligence (DFKI)
Niedersachsen Lab
Albert-Einstein-Straße 1, Osnabrück, Germany

Abstract

Online planning under uncertainty on robots and similar agents has very strict performance requirements in order to achieve reasonable behavior in complex domains with limited resources. The underlying process of decision-making and information gathering is correctly modeled by POMDP's, but their complexity makes many interesting and challenging problems virtually intractable. We address this issue by introducing a method to estimate relevance values for elements of a planning domain, that allow an agent to focus on promising features. This approach reduces the effective dimensionality of problems, allowing an agent to plan faster and collect higher rewards. Experimental validation was performed on two challenging POMDP's that resemble real-world robotic task planning, where it is crucial to interleave planning and acting in an efficient manner.

1 INTRODUCTION

Planning and acting simultaneously comes naturally to us as human beings, but is deceivingly complicated to model and replicate. As the result of a long evolutionary process, we are capable of deliberating about relatively complicated problems and often find reasonable solutions by employing a number of cognitive resources. We can shift our attention across entities, perceived or imagined, and selectively focus on one or more at a time, in order to examine their potential contributions within the context of problem solving. More often than not, however, we are not aware of the underlying selection process and do not consciously reason about the selection criteria. On the other hand, an artificial agent (such as a mobile robot) must explicitly allocate resources for deliberation – a task that grows in complexity in non-deterministic and uncertain domains. If we intend to design agents that can efficiently plan, or deliberate about the effects of actions, and subsequently execute these actions in a seamless, continuous stream to solve practical problems, we must find ways to eschew brute-force computation.

Initial attempts to include uncertainty in planning problems focused on logic models with added probabilities, such as possible-worlds planning (Thiébaux and Hertzberg, 1992; Boutilier et al., 1996). The current standard models are, however, the Markov Decision Process (MDP) and the Partially Observable MDP (POMDP). POMDP's are particularly useful to model robotic task planning: they explicitly represent the uncertainty of observations, the effect of information-gathering actions and the agent's own belief about its current state, all modeled as probability distributions. Consequently, POMDP's also become intractable very quickly.

Existing planners can solve relatively large POMDP's, and produce good results in game-like or idealized planning domains (Silver and Veness, 2010; Somani et al., 2013). Most test problems, however, tend to be simplified through prior expert analysis and therefore fail to represent challenges that an actual robot might face, making POMDP's for practical problems larger and exponentially more complex. Modern Monte-Carlo planners are particularly well suited for online planning, and manage to cope with issues such as dimensionality and history. Planning onboard robots, however, imposes additional limitations that often lead to poor performance: we suggest that dimensionality can be reduced further by selectively ignoring certain elements in the domain, effectively reducing the number of available actions and their associated observations and consequently the branching factor of the underlying search tree.

[*]Corresponding author: jcsaborio@uos.de
[†]Also with University of Osnabrück

With POMDP states as feature vectors, each element is a feature that either contributes to reaching the goal (at some point) or doesn't, in which case it becomes simply a distraction. By assigning a numerical value to a feature's usefulness, the agent can progressively shift its focus towards more useful features and their associated actions and ignore the rest. This process, which we call incremental refinement (IRE), uses the perceived relevance of features (based on current opportunities) to prune the underlying POMDP and help the agent avoid entire sections of the state space that lead to low rewards. This work is part of our effort to formalize *relevance* within the context of online planning, a concept we previously defined as "an attentional filter guiding an agent's perception and action selection".

In this paper we introduce IRE, which easily integrates into Monte-Carlo POMDP planners, as well as a new challenging POMDP. We begin by reviewing relevant related work, followed by a brief state-space analysis of *robotic* POMDP's. We then describe our method, its implications for dimensionality reduction and provide an upper limit for its approximation error. We provide experimental results in two sets of difficult POMDP's and conclude by discussing our results and derived future work.

2 RELATED WORK

POMDP planning has decades worth of literature. A large family of algorithms use piece-wise linear approximations of the optimal value function (Smallwood and Sondik, 1973), with methods such as Witness (Cassandra et al., 1994) and Incremental Pruning (Cassandra et al., 1997), which actively select and discard vectors that correctly approximate the optimal value function. Point-based algorithms improve scalability by maintaining fewer vectors and updating select points in belief space, often chosen through heuristics. Examples include HSVI (Smith and Simmons, 2004), PBVI (Pineau et al., 2006), Perseus (Spaan and Vlassis, 2005) and SARSOP (Kurniawati et al., 2008). PBVI and SARSOP have some limited robotic applications, but are restricted to very small POMDP's.

Alternatively, sampling-based POMDP planning uses a generative model to simulate transitions and approximate state and action values. UCT, a Monte-Carlo Tree Search (MCTS) algorithm based on UCB1 (Auer et al., 2002), is commonly used to solve fully-observable MDP's (Kocsis and Szepesvári, 2006). POMCP solves POMDP's through two significant changes to UCT: it approximates the belief-state with an unweighted particle filter, and expands a tree of histories instead of a tree of states (Silver and Veness, 2010). As an MCTS algorithm, it also translates well to online and anytime robot planning. In this paper we adopt this approach and will therefore refer to states with partially observable elements, instead of explicit belief states. Another algorithm combines heuristics and sampling to prefer reachable beliefs and obtain better worst-case performance than POMCP, but similar average performance (Somani et al., 2013). We propose instead addressing the underlying elements that lead to such a large belief space.

The dimensionality of large POMDP's can be addressed by clustering states and generalizing state or belief values (Pineau et al., 2003), random forest model learning (Hester and Stone, 2013) and value-function approximation with e.g.: neural networks. None of these methods exploit the underlying structure of problems, but focus on a general abstraction of POMDP's. Machine learning techniques have also been applied to POMDP's, but they come with challenges of their own and rely on simplistic planning methods (Karkus et al., 2017; Igl et al., 2018). Action hierarchies provide abstractions that also improve performance (Sutton et al., 1999; Dietterich, 2000; Vien and Toussaint, 2015), but they must be built in advance whereas our proposal works entirely online and does not rely on prior knowledge. The resulting simplification provided by IRE is somewhat similar to state factoring, but these techniques assume independence of state variables and require prebuilt dependencies and a *good* choice of basis functions. In addition, not all problems can be easily or conveniently factored, so we focus on planning directly over unfactored representations.

The underlying assumption of our approach is that it is possible to simplify the *dense* action set of a full POMDP model, and obtain a *sparse* representation with mostly relevant features. Feature-relevance estimation is based on action values, so in addition we use partial goal satisfaction (PGS) to improve action selection (Saborío and Hertzberg, 2019). PGS works as a Monte-Carlo rollout policy and a reward-shaping bonus, that allows an agent to exploit reward opportunities faster.

2.1 NOTATION

We rely on the standard notation for (PO)MDP's: let S and A be finite sets of states and actions, $T(s, a, s') = P(s'|s, a)$ the transition probability to state s with s, a and R a set of real-valued rewards. The tuple $\langle S, A, T, R\rangle$ is an MDP and γ its problem-defined discount factor. For partially observable domains, $O(s, a, \omega) = P(\omega|s, a)$ is the probability of receiving observation $\omega \in \Omega$, and $b \in B$ is the agent's internal belief state, where $b(s)$ is the probability of s being the current state. A POMDP is the tuple $\langle S, A, T, R, \Omega, O\rangle$, and the sequence $h_t = (a_0, \omega_1, \ldots, a_{t-1}, \omega_t)$ is the *history* at

time t. Notice that the belief space is a $|S|$-dimensional hyperplane, which underlines the importance of efficient dimensionality reduction techniques.

3 FEATURES AND COMPLEXITY

We are interested in POMDP's that model robotic task-planning under uncertainty, meaning an agent can move (in 2D or 3D), manipulate objects and obtain information through sensors. These objects or entities in the planning domain correspond to features, and we argue that the complexity of such a POMDP is largely dependent on them. This notation assumes states with mixed observability, so fully-observable features don't yield probabilistic observations. Previous studies of point-based POMDP algorithms also realized that this assumption better reflects the practical complexity of POMDP planning (Hsu et al., 2007).

Let η be the number of valid positions the agent can assume, p the number of locations or positions of features, k the number of features with variable positions (eg.: those that move or that the agent might move), and $\Omega_f \subseteq \Omega$ the subset of observations of feature $f \in \mathcal{F}$ (eg. obtained through information-gathering actions). The state-space of such a POMDP is:

$$\underbrace{\eta \binom{p}{k}}_{\text{World configurations}} \cdot \overbrace{\binom{p}{|\mathcal{F}|} \prod_{f \in \mathcal{F}} |\Omega_f|}^{\text{Observation space}} \tag{1}$$

where the left-most side corresponds to the number of observable world configurations and the right-most side, to the *possible worlds* generated by the different observations. The binary coefficient in both sides is the number of positions of the k movable features. The observation space however assumes an agent doesn't know which features can actually move, so it maintains a belief about the position of every feature. The product of feature-related observations gives the total distribution of qualitative observations, such as "valuable", or "useful".

For example, in Rocksample any given rock can be either valuable or not valuable, so for 8 rocks there are 2^8 possible arrangements, and for Rocksample[7,8] eq. 1 returns 12544 states. If there were 15 additional objects, with two types of observations each, and 8 of them were movable, the state space would increase to more than 10^{15}.

For a goal-driven agent, however, many features and their actions are irrelevant and will in fact be ignored (e.g.: after many simulations). We propose estimating feature relevance directly as a function of the combined values of actions for a given feature. Features are only relevant for goal-oriented planning as long as they provide or lead to useful (goal-related) interactions, so the discounted rewards of these useful action opportunities should be reflected in their relevance value. This is formally presented in the next section.

4 FEATURES AND RELEVANCE

We will first define a value function that associates action-values to features and second, a feature-value function that combines these quantities to estimate an overall relevance value. In practice it is necessary to identify which features are affected by which actions, and we propose doing so in a simple lookup table with an update rule. Unlike tabular methods for planning or reinforcement learning, this particular table contains entries only for each feature and action pair. This is perfectly manageable for most if not all POMDP's, unlike state or belief tables.

For simplicity these equations are given in terms of states (regardless of partial observability). Transferring these methods to POMDP's is trivial under the Monte-Carlo POMDP approach.

4.1 FEATURE VALUES

MCTS and similar planning methods estimate an action-value function $\hat{q}(s, a)$, that approximates the true $q(s, a)$ function. This represents the discounted return of executing action a in state s and then following some policy π. The action-value function q can be written in terms of a state-value function:

$$q(s, a) = \mathbb{E}[R(s, a, s') + \gamma v^*(s')]$$

where $\gamma \in [0, 1]$ is a discount factor and v^* is the optimal state-value function, that is:

$$v^*(s) = \max_{a \in A} \sum_{s'} p(s'|s, a)[r(s, a, s') + \gamma v^*(s')]$$

Bellman's optimality equation for q is therefore:

$$q(s, a) = \sum_{s'} p(s'|s, a)[r(s, a, s') + \gamma \max_{a'} q(s', a')]$$

which relates the action-value function to its transition probability, its immediate reward ($r(s, a, s')$) and its expected reward following the optimal policy. Intuitively

this means that for any given state s, all of its associated $q(s,a)$ represent the immediate and the long-term benefit of executing a, which is linked to a particular feature. For example, if a = 'scan object 1', then $q(s,a)$ offers information about object 1 (the feature) in the form of a high or a low reward. However, for some particular s and some particular policy, the opportunity to execute a and discover its potential benefit may be too far into the future and its contribution to $q(s,a)$ too heavily discounted. But if such action and its associated feature are relevant in the future, they might also be relevant now. For the purposes of relevance estimation, let's assume the following:

Hypothesis 1 *For some given $a \in A \;.\; \exists\; s, s' \in S$ s.t. $r(s,a,s')$ is significant $\rightarrow$ a is relevant.*

In other words, there might be a transition with a substantial reward that makes this action, and its feature, relevant. This suggests all potential future rewards should be (somewhat) equally considered. The contribution of an action/feature pair can then be estimated by *expanding* the underlying MDP and considering all possible outcomes on the subset $\mathcal{S}_a \subseteq S$ of states where a is available:

$$\text{relevance of action } a \begin{cases} q(s_0,a) \\ q(s_1,a) \\ \cdots \\ q(s_{n-1},a) \\ q(s_n,a) \end{cases}$$

where $\forall i.0 \leqslant i \leqslant n \;.\; s_i \in \mathcal{S}_a$, which leads to the value function:

$$\begin{aligned} v(f,a) &= \mathbb{E}[q(\cdot,a)] \\ &= q(s_0,a)p_0 + q(s_1,a)p_1 + \\ &\quad \ldots + q(s_{n-1},a)p_{n-1} + q(s_n,a)p_n \\ &= \frac{1}{|\mathcal{S}_a|} \sum_{s \in \mathcal{S}_a} q(s,a) \end{aligned} \tag{2}$$

We assume the distribution of action values is uniform in this case because we are dealing with prospects, estimating the value of all afforded actions, and actual q-values correctly combine reward and probability. An effect of this average is that if there are many states where a has a high (or low) return, this will be reflected in the value of this feature-action pair.

4.2 FEATURE RELEVANCE

The relevance of a feature is defined as the expected value of its afforded, weighted actions, as in eq. 3.

$$\begin{aligned} V(f) &= \mathbb{E}\left(w(v(f,a))\right) \\ &= \frac{1}{N} \sum_{a} w(v(f,a)) \end{aligned} \tag{3}$$

where w is some chosen importance scaling function. This is necessary to more closely model preferences and intuitive relevance estimation in extreme cases such as:

1. Underestimation: a feature has mostly actions with very low rewards, except for one, extremely good, goal-related action. Its relevance would otherwise be very low.
2. Overestimation: a feature has very few or no useful actions but they are not particularly punishing. Its arithmetic mean is above the relevance threshold, but the feature is not particularly relevant.
3. Partial sampling: a feature offers several actions but few of them can actually be sampled (eg.: unreachable states). If only case 2 actions are available, its value will be inaccurate.

We propose a value scaling function (eq. 4) based on the idea that the contribution of values to feature relevance is linear when they are negative but grows rapidly when they are positive, with a fixed punishment for actions that cannot be executed ($N = 0$).

$$w(v(f,a)) = \begin{cases} v^n & \text{if } v > 0 \\ \kappa & \text{if } N = 0 \\ v & \text{if } v < 0 \end{cases} \tag{4}$$

with $\kappa \in \mathbb{R}^-$ and $n \in \mathbb{R}^+$. This assumes, as is the case in most POMDP's, that goal-related rewards are positive. Eqs. 4 and 3 can also be chosen to represent (and test) some particular relevance criteria (e.g.: human preferences in known domains).

4.3 VALUE APPROXIMATION

In order to implement these methods and easily identify action-feature pairs and values we propose using a simple table.

Definition 1 *Let $\mathbb{T}$ be a catalog or table with entries corresponding to the mapping $\mathcal{F} \times A \rightarrow \mathbb{R}$, for features $f \in \mathcal{F}$, their associated actions $a \in A$ and their corresponding feature value v.*

Table entries are therefore tuples of the form $\langle f, a, v \rangle$. Values can be obtained alongside state-action values, since they also use a combination of immediate and discounted, delayed rewards. This value, $v(f, a)$, can be computed as per equation 5:

$$v(f,a) = \frac{1}{N} \sum_{s} r(s,a,s') + \gamma_f \max_{a'} q(s',a') \quad (5)$$

where N is the number of times action a was executed and γ_f is a discount factor. Eq. 5 can be easily implemented as an online average (eq. 6):

$$v(f,a) \leftarrow v(f,a) + \frac{r - v(f,a)}{N} \quad (6)$$

where $r = r(s,a,s') + \gamma_f \max_{a'} q(s',a')$ is the immediate reward plus the discounted future reward and N counts the number of updates. We approximate $v(f,a)$ using a different discount factor, $\gamma_f < \gamma$, to reflect the short-term effect of actions instead of the long horizon normally used in planning. Finally the feature-value, $V(f)$, is computed from the elements in $\mathbb{T}$.

In between planning phases, once an action has been chosen and executed in the "real world", all table entries must be updated to reflect the new current state. Feature relevance may change overtime, behaving in practice as a nonstationary problem. We suggest including a type of learning factor to combine previous and current estimates, weighing samples from the current state more heavily. A simple solution is resetting the count in each tuple to 1 and using the current value as a prior, which leads to a variable learning factor of $1/N$ for N samples, a solution known to minimize regret (Auer et al., 2002). With this simple update rule, if the approximation was correct then new updates won't affect the current estimate. If the approximation is incorrect but the action is promising, the action selection policy will choose it often in simulation and the entry will be corrected. If the value was correct but in the current state is no longer valid (eg.: the reward source was exhausted), the old information will quickly be overwritten.

Intuitively, the relevance of a feature at a given time is determined by the possible actions that can be simulated from the starting state and how they affect this feature. Without useful actions, the feature might be considered irrelevant. Features can be selectively enabled and disabled during (online) planning according to their current relevance status, allowing simulations and rollouts to focus only on those actions that affect active features. This reduces the number of reachable states and consequently the effective problem size.

4.4 DIMENSIONALITY REDUCTION

The core principle of our proposal is that irrelevant features may be ignored, since their presence does not significantly affect the resulting policy. By avoiding features, the action branching factor is reduced and many (belief) states avoided, pruning the search tree. Depending on each problem, ignoring a feature might involve additional considerations (eg.: a mobile robot may ignore a chair but not drive through it). In its current form our proposal uses a simple threshold check:

$$\text{active}(f) = \begin{cases} \textbf{true} & \text{if } V(f) \geqslant \tau \\ \textbf{false} & \text{otherwise} \end{cases} \quad (7)$$

where $\tau \in \mathbb{R}$. The set of available actions at some given moment t reduces to:

$$A_t = A \setminus \mathcal{A}^- \quad (8)$$

with $\mathcal{A}^-$ the set of inactive actions, defined by the actions of inactive features:

$$\mathcal{A}^- = \bigcup_{f \in \mathcal{F}^-} \mathcal{A}(f) \quad (9)$$

where $\mathcal{F}^- = \{f \mid V(f) < \tau\}$ is the set of inactive features, and $\mathcal{A}(f) \subset A$ the subset of actions linked to feature f. This set may be used with any action-selection rule to ignore actions of features with low relevance values, resulting in relevance-aware versions of UCB, ϵ-greedy, etc.

4.4.1 Bounds For Feature Activation

Inactive features are, in principle, those that do not help an agent achieve its goal. Here we will briefly explain their properties and their connection to the activation threshold. For the following let

$$w^+ = \sum_{a} \mathbb{1}_{\mathbb{R}^+}(w(v(f,a))) \quad (10)$$

denote the sum of all positive (including 0) values,

$$w^- = \sum_{a} \mathbb{1}_{\mathbb{R}^-}(w(v(f,a))) \quad (11)$$

represent the sum of all negative values (where $\mathbb{1}_{\mathbb{R}}(x) = x \in \mathbb{R}$ is the indicator function), and N the number of actions linked to feature f (those that participate in its value function). Inactive features satisfy $V(f) < \tau$ and correspondingly:

$$\begin{aligned} \frac{w^+ + w^-}{N} &< \tau \\ w^+ &< \tau N - w^- \\ \text{(and consequently) } w^+ &< |w^-| \ \wedge \ \tau N - w^- > 0 \end{aligned} \tag{12}$$

This states the mean of all positive, importance-weighted values $w(v(f,a))$ is lower than the mean of negative values. From eq. 12 it follows that:

$$\nexists a \in \mathcal{S}_a \quad w(v(f,a)) \geqslant \tau N - w^- \tag{13}$$

in other words, there is no single action with an importance-weighted value of at least $\tau N - w^-$. Since the importance weight for positive values is a power of n, and $\tau N - w^- > 0$ it is also true that:

$$\forall a \in \mathcal{S}_a \quad v(f,a) < 0 \ \vee \ v(f,a) < \sqrt[n]{\tau N - w^-} \tag{14}$$

which guarantees that an inactive feature does not have any actions such that $v(f,a) \geqslant \sqrt[n]{\tau N - w^-}$. In a given setting, if n, τ and N act as constants the main influence in the activation or deactivation of a feature is w^-. This means inactive features are those with actions that yield significantly (n-degree polynomial) larger negative reward, than they do positive reward. Modifying τ online would either relax or restrict feature activation, and perhaps such a variable policy might be useful in some domains.

4.4.2 Estimation Errors

From eqs. 2 and 14 it follows that $\mathbb{E}[q(\cdot,a)] < \sqrt[n]{\tau N - w^-}$ for inactive features, so let $\mathcal{U} = \sqrt[n]{\tau N - w^-}$ be the upper bound of the expected action value. When a feature is active, it doesn't affect an action selection policy so the base convergence theorems hold (e.g. for UCT, etc.). The same applies when a feature is inactive, and its actions are not part of a (sub) optimal policy (i.e. an action selection policy with decreasing exploration rate would eventually ignore them).

Non feature-related actions, such as navigation, are not affected by feature activation and are always available in their corresponding states. We then have two cases of interest: the first occurs when an action is part of an ϵ-optimal policy but is unavailable during action selection due to a value approximation error (should be active instead). The second case is when the action-value approximation is correct but very low, and so the feature-value is below the activation threshold. This means the action is required but cannot be selected.

In the first case, the action value should satisfy $v \geqslant \mathcal{U}$ but due to errors, it does not. We can then express the probability of the approximation $v + \epsilon$ falling under the upper bound as $P\{v < \mathcal{U} - \epsilon\}$ which can be estimated using Hoeffding's theorem, that states:

$$P\{\bar{X} \leqslant \mu - a\} \leqslant e^{-2a^2/n_X}$$

where $\bar{X}$ is the empirical mean from n_X elements, μ is the distribution mean and a an upper bound. In our case $\bar{X} = \hat{q}(s,a)$ is our empirical mean, $\mu = q(s,a)$ is the true mean, and $a = \mathcal{U} - \epsilon$ the upper bound. We then get:

$$\begin{aligned} P\{\bar{X} \leqslant \mu - a\} &\leqslant e^{-2(\mathcal{U}-\epsilon)^2/n_q} \\ &\leqslant \exp\left\{-2\frac{\left(\sqrt[n]{\tau N - w^-} - \epsilon\right)^2}{N}\right\} \end{aligned} \tag{15}$$

This probability is maximal when $\left(\sqrt[n]{\tau N - \omega^-} - \epsilon\right) \to 0$, which can happen under two circumstances:

1. When a very large error ϵ approaches the bound $\mathcal{U}$. This reflects a poor overall approximation, often the result of insufficient *budget* (eg.: Monte Carlo simulations), which also determines asymptotic convergence. This can be addressed by increasing the approximation budget or implementing more sample-efficient action selection techniques such as PGS.
2. If $\epsilon \to 0$ and $\omega^- \approx \tau N$, meaning the expected reward of the feature's actions is very close to the activation threshold τ. Keep in mind that an inactive feature also satisfies $\omega^+ < |\omega^-|$, so the reward of the *good* actions is in the interval $[0, |\tau|)$ suggesting this case is more likely when actions don't yield much reward. A simple way to address this issue is using a lower activation threshold, or perhaps even a variable threshold.

Additionally, if an action is not available (because its feature is inactive) and it is in fact optimal, then very likely no other feature-related action is available either (unless approximation errors occur in practice). A potential solution (apart from using a lower threshold) is to stochastically activate a feature, introducing an exploration pattern, or to limit the number of features that can be inactive, allowing an agent to always select among the most promising actions.

This analysis underlines the importance of a suitable activation threshold, which under appropriate circumstances will simplify a problem while preserving the inherent convergence properties of a planning algorithm. Without

further analysis, an easy recommendation is to choose a τ that is somewhat below the reward of expensive necessary actions but above other costly actions.

4.5 PLANNING WITH INCREMENTAL REFINEMENT

IRE-based POMDP planning is shown in algorithm 1. Line 3 corresponds to an IRE-enabled (or feature-aware) Monte-Carlo planner, where matching entries in $\mathbb{T}$ update their feature-values and feature-action visit counts when their corresponding action is executed. After the simulation budget is exhausted, the best action is selected in line 4 from the subset in eq. 8 (e.g.: with IRE-enabled UCB1), executed, and its corresponding results received. The POMDP simulator is updated to reflect these changes and features are activated or deactivated to reflect their current values (function BELIEFREVISION).

Algorithm 1 Online planning & acting with IRE

Input: Generative model $\mathcal{G}$, initial belief state b_0
Output: Policy π_f

```
1:  procedure ONLINE PLANNER
2:      repeat
3:          f-MCTS from s ≈ B                ▷ Updates feature-values
4:          a ← π_f(s)                       ▷ e.g.: f-UCB1
5:          Execute a                        ▷ "real-world" action
6:          Receive s', o, r
7:          Update G with a, o, r
8:          BELIEFREVISION(B(h))
9:      until s is terminal
10: end procedure

11: function BELIEFREVISION(b)
12:     for all f ∈ F do
13:         If V(f) < τ, deactivate f ∀s ∈ b
14:         Otherwise activate f ∀s ∈ b
15:         ∀⟨f, a⟩ N(f, a) ← 1
16:     end for
17:     If ∀f ∈ F are inactive, activate random f
18: end function
```

5 RESULTS

We chose two challenging POMDP's that represent actual robotic tasks, with many objects and realistic reward distributions. Our performance results show that an agent can considerably improve its performance in complex, practical problems through feature-relevance estimation.

The first problem is the cellar domain from (Saborío and Hertzberg, 2019), inspired by the well known Rocksample but with added obstacles that increase complexity. In this problem, the robot must collect valuable wine bottles while avoiding or interacting with crates and shelves. In some cases a promising bottle might be behind a crate, and the agent might have to push the crate aside. Pushing crates yields a small punishment of -2, so the agent must consider whether a bottle is in fact valuable beforehand. Shelves cannot be pushed and yield a punishment of -10, so in addition the agent must also make sure an object is a crate before pushing it. Movement incurs in a penalty of -1, while using the object scanner yields a reward of -0.5 but returns noisy information about a bottle's value and an object's type. Collecting valuable bottles awards $+10$, while non-valuable bottles cost -10. The agent must leave with at least one valuable bottle to complete the task, receiving an additional $+10$. In the experiments we used cellar[5,2,6,4], a medium-sized problem in a 5×5 grid with 2 bottles, 6 crates, 4 shelves and approximately 10^8 states, cellar[7,8,7,8] with over 10^{15} states and cellar[11,11,15,15], a large POMDP with more than 10^{30} states.

We also introduce another interesting task-planning problem, in which an unmanned aerial vehicle (UAV) must navigate a grid-shaped forest to observe and document a number of creatures. The forest has bears, trees and one or more specimens of Big Foot, the apelike creature of North American folklore. The UAV doesn't know in advance which type creatures are, so it has to identify them using a noisy sensor and update the corresponding probability. In order to do so, the UAV needs to know their most recent location, but all of them (except trees) move to adjacent locations stochastically. A grid region can be scanned with another sensor (based on Rocksample's virtual sensor), and if there is a creature its location is updated (erroneous readings are possible). Finally the UAV may photograph any creature, which also reveals their type, but photos of anything other than Big Foot are punished. Identifying and taking pictures succeeds only when the UAV is directly above a creature. We assume all features are *tagged* in advance, so their tag number or *id* is known but their location and type is not. The task is completed by taking a specified amount of pictures and leaving the area.

The reward distribution is -0.5 for sensing actions (location check and identify), -1 for navigation (including waiting and leaving) and misuse of identify (eg.: creatures with unknown location), $+5$ and -10 for good and bad photos respectively, $+10$ for terminating successfully and -100 in case of failure (leaving too soon, etc.). In our experiments, creatures move to an adjacent cell with probability 0.15 and identifications are correct with probability 0.9. Observations include the ground, Big Foot and non-Big-Foot creatures, and the tag number of each object in the domain. If we refer to the general problem as BigFoot$[n, c, t]$, for an $n \times n$ grid with c creatures and t trees, there is a total of $6 + n^2 + 2(c + t)$ actions and $3 + c + t$ observations. The problems below

are BigFoot[3,3,2], with approximately 10^6 states, BigFoot[4,3,2] with around 10^8 states and BigFoot[5,8,8] with over 10^{16} states.

We compared the performance of IRE and of regular POMCP, both with a uniformly random rollout policy (legal actions only) and with PGS. To the best of our knowledge there are no other online planners that can handle POMDP's this large,[1] so we did not perform an exhaustive comparative analysis. Statistics were compiled from experiments running on several Intel Xeon E5-2660 CPU's, but a single instance runs similarly on standard desktop hardware. All results are averaged over 100 runs and use up to 2^{16} Monte-Carlo simulations per step, but in the plots we skip the results with very few simulations to better appreciate the region with significant differences. In practice, given the size of these problems, using more simulations might be desirable (and attainable since we've achieved some speedup). Note that for the purpose of testing we quickly selected parameter values that showed promising results, so performance is limited by these choices and may be improved with further analysis for specific, real-life scenarios.

Figure 1 shows the average discounted returns in the cellar domain. In cellar[5,2,6,4] we used an activation threshold of $\tau = -6$ and $\gamma = 0.99$, while the two others used $\tau = -5$ and $\gamma = 0.95$. All three used $\gamma_f = 0.5$ and binary entropy threshold (PGS parameter) of 0.5. The maximum number of steps for each problem was 150, 250 and 350 respectively.

All plots consistently show that IRE-based methods eventually match or outperform non-relevance-aware planning, regardless of the chosen policy. PGS is already effective at gathering larger rewards using less simulations, but its performance can be greatly improved with relevance estimation. Cellar[5,2,6,4] is deceivingly complicated: it contains "only" two valuable bottles but both are out of reach, so the agent must necessarily push at least one crate. This difficulty is reflected in the performance difference of fig. 1a, where PGS's advantage is exploited by IRE to achieve even higher rewards. The performance increase is particularly noticeable in cellar[7,8,7,8] (fig. 1b), where regular PGS is matched by the IRE-enabled random policy and doubled by its IRE-based version. The largest problem has several easily accessible bottles, so the challenge is simply focusing on the bottles while determining which (if any) non-bottle objects are useful. In this case, the performance difference of feature-relevance estimation is still noticeable but not as dramatic. Planning with IRE was much faster in all cases (see table 1), with speedups of 1.7, 2.62 and 1.41 respectively.

[1] Performance with DESPOT is very close to POMCP.

In the Big Foot domain we used $\tau = -7$, $\gamma = 0.95$ and $\gamma_f = 0.5$ in all problems, with a PGS entropy threshold of 0.5 in BigFoot[5,8,8] and 0.4 in the rest, and a maximum of 150, 250 and 450 steps respectively. Figure 2 shows the average discounted returns.

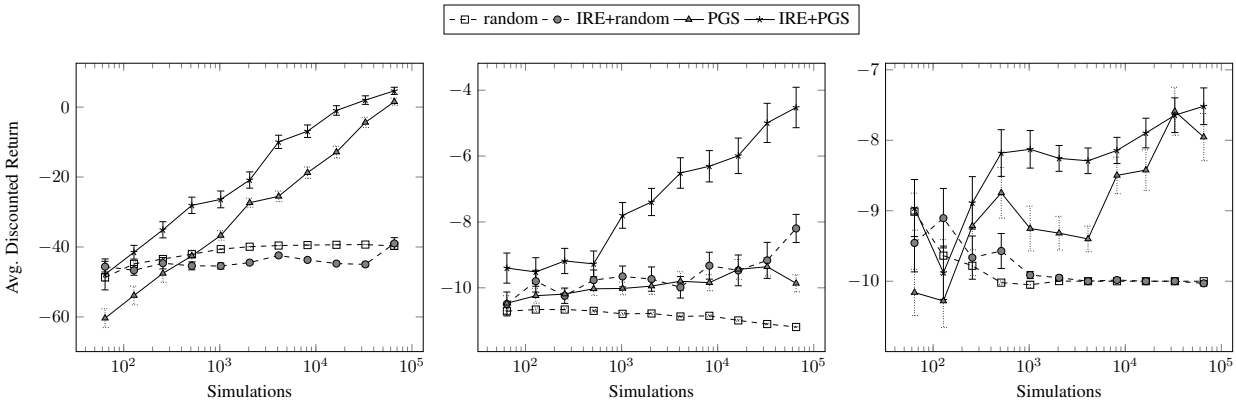
Rewards in the Big Foot domain have very long delays, collected only after the agent locates, identifies and photographs valuable creatures. Consequently, the accuracy of relevance estimation is affected by the quality of its underlying policy. Despite such limitations, incremental refinement outperforms non-IRE methods, in some instances by a significant margin. Some parameters in the Big Foot domain might require fine-tuning but, as table 1 shows, IRE already provides a clear advantage in both cumulative reward and runtime. In BigFoot[3,3,2] IRE-PGS collected almost twice the reward of standard PGS very quickly. A similar pattern can be observed in BigFoot[4,3,2]. BigFoot[5,8,8] is considerably more difficult and this difference is reflected in runtimes and rewards, both of which were improved with IRE. We achieved speedups of 1.74, 1.66 and 1.21 respectively.

In some cases the mean discounted return decreased with more simulations using the random policy. It is possible that the agent is more likely to select more useful and "expensive" actions with a larger planning budget, instead of executing only "cheaper", less useful actions, but nonetheless runs out of time. Despite this limitation, IRE seems to also improve the random policy (an exemplar case in fig. 1b).

Table 1 compiles the average running times and standard error of the two best performing policies in each problem (PGS and PGS with IRE) with 2^{16} simulations over 100 runs. IRE consistently collects higher rewards in considerably less time. These runtimes are for complete episodes with many consecutive actions; planning for a single action requires only a fraction of the time.

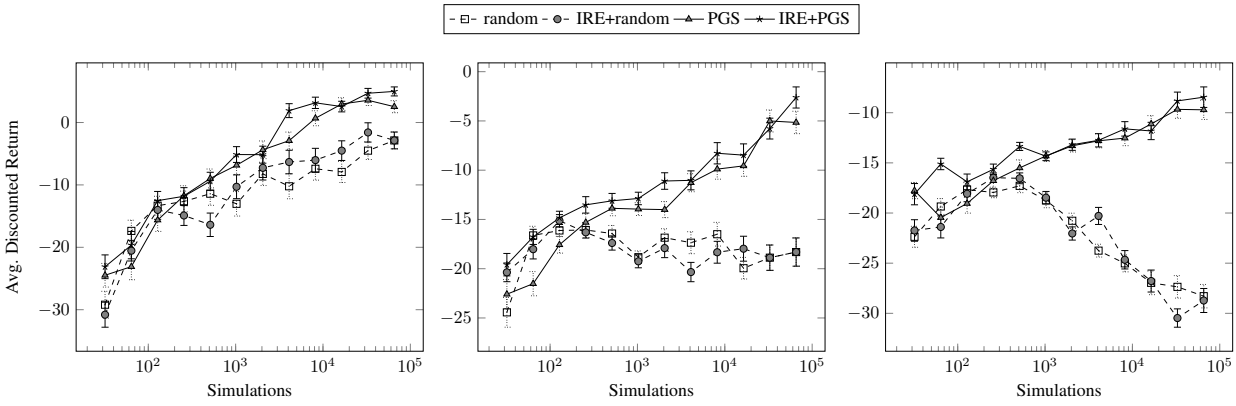
Table 1: Performance with PGS and 2^{16} simulations

Problem	**IRE**	**Reward**	**Time** (s.)
Cellar[5,2,6,4]	✓	4.645 ± 1.02	175.1
	✗	1.532 ± 0.99	310
Cellar[7,8,7,8]	✓	-4.525 ± 0.61	2169
	✗	-9.864 ± 0.26	5693
Cellar[11,11,15,15]	✓	-7.517 ± 0.26	9029
	✗	-7.954 ± 0.33	12790
BigFoot[3,3,2]	✓	4.983 ± 0.72	20.45
	✗	2.539 ± 0.97	35.54
BigFoot[4,3,2]	✓	-2.619 ± 1.07	129.8
	✗	-5.154 ± 1.14	215.5
BigFoot[5,8,8]	✓	-8.45 ± 1.02	2264
	✗	-9.711 ± 0.96	2735



(a) Cellar[5,2,6,4] (b) Cellar[7,8,7,8] (c) Cellar[11,11,15,15]

Figure 1: Performance in the cellar domain over 100 runs



(a) BigFoot[3,3,2] (b) BigFoot[4,3,2] (c) BigFoot[5,8,8]

Figure 2: Performance in the Big Foot domain over 100 runs

6 CONCLUSIONS

We designed and implemented a relevance-estimation method for online POMDP planning, that approximates feature relevance by a weighted combination of action values. Features that satisfy the threshold criteria are considered relevant, and the rest are ignored by the planner. Our theoretical results show that there exists a threshold value that separates valuable features from those with overall low expected rewards, effectively simplifying large problems. The experimental results confirm that this approach significantly improves runtime and expected returns, without the use of detailed domain knowledge. Efficient, online POMDP planning opens up many opportunities in a variety of challenging scenarios that combine exploration, manipulation and information-gathering in dynamic environments.

Future work includes designing and experimenting with different scaling functions and activation policies, for instance based on value histories, general prior knowledge or on predefined preference criteria. In this paper we showed that a fixed threshold is sufficient to achieve promising results, but in practice more finely tuned parameters might perform even better. We will continue to work towards transferring these methods onboard mobile robots, focusing on practical tasks that require efficiently interleaving planning and acting.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback, as well as Felix Igelbrink and Sebastian Pütz for their input in the planning problems. This work was supported by a DAAD grant.

References

Auer, P., Nicolò, C.-B., and Fischer, P. (2002). Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2):235–256.

Boutilier, C., Dean, T., and Hanks, S. (1996). Planning Under Uncertainty: Structural Assumptions and Computational Leverage. In *Proc. of the 2nd European Workshop on Planning*, pages 157–171. IOS Press.

Cassandra, A. R., Kaelbling, L. P., and Littman, M. L. (1994). Acting Optimally in Partially Observable Stochastic Domains. In *Proc. of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 2.*, pages 1023–1028.

Cassandra, A. R., Littman, M. L., and Zhang, N. L. (1997). Incremental Pruning: A Simple, Fast, Exact Method for Partially Observable Markov Decision Processes. In *UAI '97: Proc. of the 13th Conference on Uncertainty in Artificial Intelligence, Brown University, Providence, Rhode Island, USA, August 1-3, 1997*, pages 54–61.

Dietterich, T. G. (2000). Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research*, 13:227–303.

Hester, T. and Stone, P. (2013). TEXPLORE: Real-Time Sample-Efficient Reinforcement Learning for Robots. *Machine Learning*, 90(3).

Hsu, D., Lee, W. S., and Rong, N. (2007). What Makes Some POMDP Problems Easy to Approximate? In *Proc. of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 689–696, USA. Curran Associates Inc.

Igl, M., Zintgraf, L., Le, T. A., Wood, F., and Whiteson, S. (2018). Deep Variational Reinforcement Learning for POMDPs. In Dy, J. and Krause, A., editors, *Proc. of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2117–2126, Stockholmsmässan, Stockholm Sweden. PMLR.

Karkus, P., Hsu, D., and Lee, W. S. (2017). QMDP-Net: Deep Learning for Planning under Partial Observability. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4694–4704. Curran Associates, Inc.

Kocsis, L. and Szepesvári, C. (2006). Bandit based Monte-Carlo Planning. In *ECML-06. Number 4212 in LNCS*, pages 282–293. Springer.

Kurniawati, H., Hsu, D., and Lee, W. S. (2008). SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In Brock, O., Trinkle, J., and Ramos, F., editors, *Robotics: Science and Systems*. The MIT Press.

Pineau, J., Gordon, G., and Thrun, S. (2003). Policy-contingent Abstraction for Robust Robot Control. In *Proc. of the 19th Conference on Uncertainty in Artificial Intelligence*, UAI'03, pages 477–484, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Pineau, J., Gordon, G. J., and Thrun, S. (2006). Anytime Point-Based Approximations for Large POMDPs. *Journal of Artificial Intelligence Research*, 27:335–380.

Saborío, J. C. and Hertzberg, J. (2019). Planning Under Uncertainty Through Goal-Driven Action Selection. In van den Herik, J. and Rocha, A. P., editors, *Agents and Artificial Intelligence*, pages 182–201, Cham. Springer International Publishing.

Silver, D. and Veness, J. (2010). Monte-Carlo Planning in Large POMDPs. In *Advances in Neural Information Processing Systems 23*, pages 2164–2172.

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 *Proc. of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pages 520–527, Arlington, Virginia, United States. AUAI Press.

Somani, A., Ye, N., Hsu, D., and Lee, W. S. (2013). DESPOT: Online POMDP Planning with Regularization. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 1772–1780. Curran Associates, Inc.

Spaan, M. T. J. and Vlassis, N. A. (2005). Perseus: Randomized Point-based Value Iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220.

Sutton, R., Precup, D., and Singh, S. (1999). Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence*, 112:181–211.

Thiébaux, S. and Hertzberg, J. (1992). A Semi-Reactive Planner Based on a Possible Models Action Formalization. In *Artificial Intelligence Planning Systems: Proc. of the First International Conference (AIPS92)*, pages 228–235. Morgan Kaufmann.

Vien, N. A. and Toussaint, M. (2015). Hierarchical Monte-Carlo Planning. In *Proc. of the 29th AAAI Conference on Artificial Intelligence (AAAI 15)*.