Adaptive Information Belief Space Planning

Moran Barenboim[1] and **Vadim Indelman**[2]
[1]Technion Autonomous Systems Program
[2]Department of Aerospace Engineering
Technion - Israel Institute of Technology, Haifa 32000, Israel
moranbar@campus.technion.ac.il, vadim.indelman@technion.ac.il

Abstract

Reasoning about uncertainty is vital in many real-life autonomous systems. However, current state-of-the-art planning algorithms either cannot reason about uncertainty explicitly, or do so with high computational burden. Here, we focus on making informed decisions efficiently, using reward functions that explicitly deal with uncertainty. We formulate an approximation, namely an abstract observation model, that uses an aggregation scheme to alleviate computational costs. We derive bounds on the expected information-theoretic reward function and, as a consequence, on the value function. We then propose a method to refine aggregation to achieve identical action selection in a fraction of the computational time.[1]

1 Introduction

Planning under uncertainty is a recurrent aspect in many practical autonomous systems. The uncertainty stems from the fact that in many real-life scenarios the state of the agent and the surrounding world is not known precisely. Some of the most common sources for uncertainty include sensor noise, modeling approximations and the unknown environment of the agent. A common way to make decisions while considering uncertainty is to formulate the problem as partially observable Markov decision process (POMDP).

Since in POMDPs states are not directly observed, all past action-observation pairs serve as an alternative for making decisions. Storing all previous actions and observations over long trajectories may be expensive in terms of space. Instead, a common approach is to calculate a distribution over the unobserved states, also known as belief states. In most POMDP formulations, belief states are assumed to be sufficient statistics, such that they are equivalent to the observed history of actions and observations when considering decision making [Thrun *et al.*, 2005]. An optimal solution to the POMDP is a policy which maximizes some objective function, usually defined as the sum of expected future reward values over the unknown states. A policy then maps each belief state to an action. Unfortunately, the exact solution of a POMDP is computationally infeasible for all but very small POMDPs [Papadimitriou and Tsitsiklis, 1987].

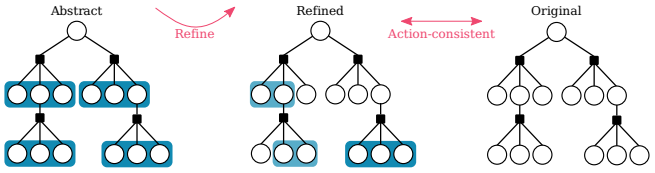

Figure 1: An illustration of our approach. The blue clusters correspond to a single evaluation of the reward function across different posterior nodes, which is faster to compute. Then, the algorithm initiates a refinement procedure; the refined clusters guarantee the same action selection as the original reward evaluation.

A POMDP formulation that is defined with state-dependent reward, i.e., a reward function that is defined over states, can only reason about uncertainty implicitly. However, it is insufficient for problems where the goal is uncertainty reduction or information gathering. Instead, it was previously shown that a belief-dependent reward, which is a reward over the state distribution, is a more natural definition for uncertainty reduction tasks. For instance, consider the active localization problem [Burgard *et al.*, 1997], where the goal of the agent is not to reach a certain destination, but to reduce uncertainty about its state. Other motivating examples may include search and rescue missions [Moon *et al.*, 2022], object search and active SLAM [Kim and Eustice, 2014]. With a belief-dependent reward, the problem formulation is considered an extension to POMDP, which exists in the literature under different names, such as ρ-POMDP [Araya *et al.*, 2010], POMDP-IR [Spaan *et al.*, 2015] or Belief Space Planning (BSP) [Platt *et al.*, 2010; Van Den Berg *et al.*, 2012; Indelman *et al.*, 2015].

Common approaches for measuring uncertainty are information theoretic functions, such as entropy, information gain and mutual information. For continuous distributions, which are of interest in this paper, calculating the exact values of information theoretic rewards involves intractable integrals in the general case. Thus, they are amenable to different approximations, such as kernel density estimation (KDE), Voronoi diagrams [Miller, 2003] or sampling based approximations [Boers *et al.*, 2010]. Unfortunately, all such approximations are expensive to compute, as they require quadratic costs in

[1]For the appendix and the code please visit https://github.com/moranbar/Adaptive-Information-BSP

the number of samples and are usually the bottleneck of planning algorithms.

Since finding an exact solution to a POMDP is intractable, approximate algorithms have been developed. Tree search algorithms, which are the main focus of this paper, are a prominent approach for such approximation; instead of considering all possible belief states, tree search algorithms reduce the belief space to a reachable subset, starting from the prior belief node. In this paper, we consider an online paradigm, where instead of calculating a policy in advance, the planner needs to find an (approximately-) optimal action, by building a tree every time-step. POMCP [Silver and Veness, 2010], is a tree search algorithm, which extends MCTS [Browne *et al.*, 2012] to the POMDP framework; it is considered a state-of-the-art (SOTA) algorithm, however it is limited to discrete state, action and observation spaces. DESPOT [Somani *et al.*, 2013] and its descendants [Ye *et al.*, 2017; Garg *et al.*, 2019] consider α-vectors in their derivations and are thus limited to reward functions that are linear functions in the belief. However, information-theoretic functions are usually not linear with respect to the belief and thus not supported by these approaches.

POMCPOW [Sunberg and Kochenderfer, 2018] is an extension of the POMCP algorithm, to deal with continuous action and observation spaces. In POMCPOW, each belief node is represented by a set of particles. The number of particles in each node is determined by the number traversals within this node in planning. Due to the exploratory nature of the algorithm, most belief nodes contain only a few particles, which are unsuitable to approximate belief dependent reward. In PFT-DPW algorithm [Sunberg and Kochenderfer, 2018], each expanded node in the belief tree contains the same number of particles and is better suited for belief dependent rewards. A recent result by [Hoerger and Kurniawati, 2021] shows comparable or better performance than current SOTA algorithms, by considering trajectories of particles and using a weighted particle filter to extract trajectories for any observation, but suffers from the same problem as POMCPOW. [Flaspohler *et al.*, 2019] uses MCTS with information-theoretic reward as heuristic over state dependent objective function for information gathering.

More closely related to our work is [Thomas *et al.*, 2021], which interleave MCTS with ρ-POMDP. Their approach considers a discrete observation space. In each traversal of the tree, their algorithm adds a fixed set of particles propagated from the root node, which results in an increased number of samples at each node as the algorithm progresses. According to the authors, the main motivation is a reduced asymptotic bias. Given a time budget, a significant reduction in the number of nodes explored is observed by the authors of [Thomas *et al.*, 2021], which in turn impacts the quality of the policy. [Fischer and Tas, 2020] consider a belief dependent reward, in which they build upon PFT-DPW [Sunberg and Kochenderfer, 2018]. Instead of maintaining the same particle set in each posterior node, they reinvigorate particles in every traversal of the posterior node. Then, they propose to average over different estimations of the reward function. [Fischer and Tas, 2020] suggest to estimate the reward function using KDE, which scales quadratically with the number of state samples. [Sztyglic and Indelman, 2021] consider a sampling-based approximation to evaluate differential entropy. They propose a simplification procedure that alters the number of state samples and prune sub-optimal action branches, using bounds relative to the non-simplified estimator. In a follow-up work, [Sztyglic *et al.*, 2021] propose an approach to interleave simplification with MCTS while maintaining tree-consistency, thereby increasing computational efficiency. Our work is complementary to these works and can be combined.

In this paper, we show an approach to alleviate the computational burden of calculating reward at each belief node of the tree, while guaranteeing identical solution. We focus on Shannon's entropy and differential entropy, alongside state-dependent reward functions. Our approach relies on clustering different nodes and evaluating an approximated belief dependent reward once on this entire cluster, that is, all the nodes within a cluster share the same reward value, see figure 1. As a result, the estimated value function is affected. To relate the approximated value function to the one that would originally be calculated, each node maintains a lower and upper bound on the value function.

Consequently, our main contributions are as follows. First, we introduce an abstract observation model. We use the model to form an abstraction of the expected reward, namely weighted average of state-dependent reward and entropy. Then, using the abstract model, we show how computational effort is alleviated. Second, we derive deterministic lower and upper bounds with respect to the underlying expected reward values and the value function. Third, we introduce a new algorithm, which is able to tighten the bounds upon demand, such that the selected action is guaranteed to be identical to the non-simplified algorithm. Last, we evaluate our algorithm in an online planning setting, and show that our algorithm outperforms the current state-of-the-art.

2 Preliminaries

2.1 Problem Formulation

POMDP is defined as a 7-tuple, (S, A, O, T, Z, R, b_0), where, S, A, O denotes the state, action and observation spaces respectively. $T(s' \mid s, a)$ is the transition density function, $Z(o \mid s)$ is the observation density function and b_0 denotes the initial belief distribution. A history, H_t, is a shorthand for all past action-observation sequences $(a_0, o_1, \ldots, a_{t-1}, o_t)$ up to current time. We use H_t^- to denote the same history, without the last observation, i.e. $H_t^- = (a_0, o_1, \ldots, a_{t-1})$. Similarly, b_t^- is a belief conditioned on H_t^-.

In this work we focus on a reward function defined as a weighted sum of state-dependent reward and entropy,

$$R(b, a, b') = \omega_1 \mathbb{E}_{s\sim b'}\left[r(s, a)\right] + \omega_2 \mathcal{H}(b'), \qquad (1)$$

where b' is the subsequent belief to b and $\mathcal{H}(\cdot)$ is either differential entropy or Shannon's entropy. The dependence of (1) on both b and b' stems from the definition of the differential estimator, as will be shown in Section 3.2. A policy, $\pi(\cdot)$, maps a belief to an action to be executed. Given a belief at time t, each policy corresponds to a value function,

$$V^{\pi}(b_t) = \mathbb{E}_o\left[\sum\nolimits_{\tau=t}^{\mathcal{T}-1} R(b_\tau, \pi_\tau(b_\tau), b_{\tau+1})\right], \qquad (2)$$

which is the expected cumulative reward following the policy, π. Similarly, an action-value function,

$$Q^{\pi}(b_t, a_t) = \mathbb{E}_{o_{t+1}}\left[R(b_t, a_t, b_{t+1}) + V^{\pi}(b_{t+1})\right], \quad (3)$$

is the value of executing action a_t in b_t and then following the policy π.

For a given policy, it is possible to compute the value function by applying equations (1) and (2). That is, for every possible observation and time step, compute the reward value. Clearly, the number of times each reward value is calculated depends on the number of observation instances. Generally, evaluating a belief-dependent reward is computationally expensive. This paper is concerned with minimizing the number of times the expensive-to-compute reward, equation (1), is evaluated while retaining guarantees on the solution quality. As a result of our approach, planning time is diminished or, equivalently, planning performance increases under a given time budget.

2.2 Belief-MDP

A belief-MDP is an augmentation of POMDP to an equivalent MDP, by treating the belief-states in a POMDP as states in the Belief-MDP. Subsequently, algorithms developed originally for MDPs can be used for solving a POMDPs or BSP problems with slight modifications, a property which we exploit in this paper.

3 Expected Reward Abstraction

In this section we introduce the notion of an abstract observation model and show how this model can be utilized to ease the computational effort. We then derive bounds on the expected reward and, as a consequence, on the value function.

We start this section with a theoretical derivation, where we assume the reward can be calculated analytically. This appears in special cases, e.g. when the reward is solely entropy and the belief is parametrized as a Gaussian or when the state space is discrete. In this part we assume the observation space is discrete and thus the expected reward can be calculated analytically. In the second part of this section, we relax those assumptions. Generally, the observation and state spaces can be continuous and the belief may be arbitrarily distributed. To deal with such cases, we derive an estimator of the expected reward, in which the belief is approximated by utilizing a particle filter.

For both the discrete and continuous observation spaces, we present an abstract observation model,

Definition 1 (Abstract observation model)**.** *An abstract observation model assigns a uniform probability to all observations within a single set,*

$$\bar{Z}(o^j \mid s) \doteq \frac{1}{K}\sum_{k=1}^{K} Z\left(o^k \mid s\right) \quad \forall j \in [1, K] \quad (4)$$

where $Z\left(o^k \mid s\right)$ corresponds to the original observation model over different observation realizations, o^k, and K denotes the cardinality of observations within that set.

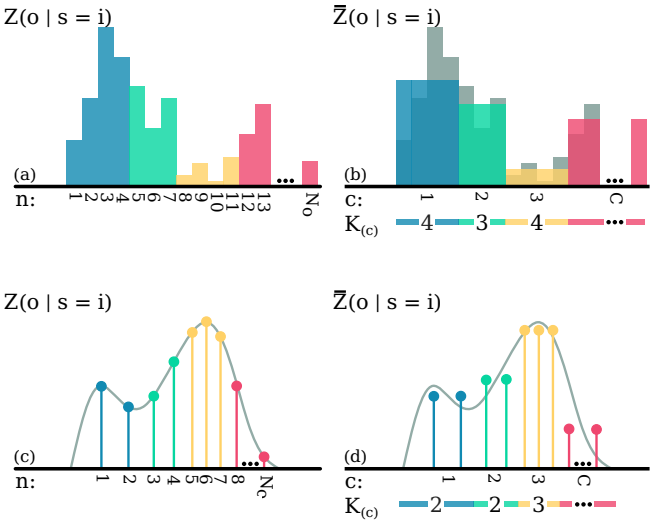

Figure 2: An abstraction of the observation model: (a) original discrete observation model with N_o observations. (b) Abstract discrete observation model with C clusters. (c) Sample set from the original continuous observation model, with N_o observations. (d) Abstract sample set with C clusters.

The abstract model aggregates K different observations and replaces the original observation model when evaluating the reward, see Figure 2. In the continuous case we revert to observation samples, thus the summation in (4) corresponds to different observation samples. A more precise explanation of the continuous case will be given in Section 3.2.

3.1 Discrete Observation Space

Since calculating the exact value of a reward function in every belief node is expensive, we now formulate an approach to evaluate rewards once for an entire set of K posterior beliefs. Such an abstraction results in a decreased number of reward evaluations in planning. We show that when constructing the aggregation scheme as a uniform distribution over a set of observations, one can achieve tight upper and lower bounds on the expected entropy, defined as $-\mathbb{E}\left[\sum_s b(s) log(b(s))\right]$. Moreover, we show that abstraction for the expected state-dependent reward does not affect its value, which remains identical with and without abstraction.

Denote the cardinality of observation space with N_o, we partition the observations to C clusters and denote the number of observations within each cluster as K, see Figure 2. Generally, each cluster may contain a different number of observations, $N_o = \sum_{c=1}^{C} K(c)$; For clarity, we assume K is identical for all clusters, but the results below are easily extended to the more general case.

Our *key result*, stated in the lemma below, corroborates the intuition that utilizing an abstract observation model results in a reduced number of reward evaluations.

Lemma 1. *Evaluation of the expected reward with an abstract observation model,* (4)*, requires only C evaluations of*

the reward, instead of N_o, where $C = \frac{N_o}{K}$. That is,

$$\sum_{n=1}^{N_o} \bar{\mathbb{P}}\left(o_{t+1}^n \mid H_{t+1}^-\right) R\left(b_t, a_t, \bar{b}_{t+1}\right) = \quad (5)$$
$$K \sum_{c=1}^{C} \bar{\mathbb{P}}\left(o_{t+1}^{c \cdot K} \mid H_{t+1}^-\right) R\left(b_t, a_t, \bar{b}_{t+1}\right).$$

Proof. see appendix A.1. □

Here, $o_{t+1}^{c \cdot K}$ denotes a single representative observation of the cluster c and

$$\bar{\mathbb{P}}\left(o^n \mid H^-\right) \doteq \sum_{s \in S} \bar{Z}(o^n \mid s) b^-, \quad (6)$$

$$\bar{b} \doteq \frac{\bar{Z}(o^n \mid s) b^-}{\sum_{s' \in S} \bar{Z}(o^n \mid s') b^-}. \quad (7)$$

For the continuous state case, simply replace summations with integrals. Furthermore, the expected state-dependent reward remains unchanged when evaluated over abstract belief and expectation,

Lemma 2. *The value of the expected state-dependent reward is not affected by the abstraction shown in* (4)*, i.e.,*

$$\mathbb{E}_o\left[\mathbb{E}_{s \sim b}\left[r\left(s, a\right)\right]\right] = \bar{\mathbb{E}}_o\left[\mathbb{E}_{s \sim \bar{b}}\left[r\left(s, a\right)\right]\right]. \quad (8)$$

Proof. see appendix A.2. □

Note that $\{\mathbb{E}_o, \mathbb{E}_b\}$ and $\{\bar{\mathbb{E}}_o, \mathbb{E}_{\bar{b}}\}$ correspond to expectations with the original and abstract observation models, (6) and (7).

We now transition to the main theorem of this paper. Using (4) as the abstraction mechanism, we show that,

Theorem 1. *The expected entropy is bounded from above and below by,*

$$0 \leq \bar{\mathbb{E}}_o\left[\mathcal{H}\left(\bar{b}\right)\right] - \mathbb{E}_o\left[\mathcal{H}\left(b\right)\right] \leq log(K). \quad (9)$$

Proof. see appendix A.3. □

A direct implication of this result is a bounded sub-optimality, which we state explicitly in corollary 1.1, while increasing computational efficiency by a factor of K. The bounds hold in the worst-case sense, i.e. regardless of the choice of which observations one chooses to cluster together. Note that the difference between the expected entropy and the abstracted one is bounded from below by zero. Since the entropy evaluates the uncertainty, the interpretation of this result is quite intuitive; the uncertainty cannot reduce when using abstracted models. The upper bound depends on the number of observations we choose to abstract, K. When $K = 1$, that is, each cluster contains a single observation, both the upper and lower bounds are zero and the abstract expected entropy equals the original expected entropy. From (1), (8) and (9) it follows that,

$$0 \leq \bar{\mathbb{E}}_o\left[R\left(b, a, \bar{b}'\right)\right] - \mathbb{E}_o\left[R\left(b, a, b'\right)\right] \leq \omega_2 log(K). \quad (10)$$

We now generalize those results and show that the value function is bounded. An abstract value function is defined as,

$$\bar{V}^\pi(b_t) = \bar{\mathbb{E}}_{o_{t+1}}\left[R(b_t, \pi_t(b_t), \bar{b}_{t+1})\right] + \mathbb{E}_{o_{t+1}}\left[\bar{V}^\pi(b_{t+1})\right]. \quad (11)$$

As a direct consequence of (10) and (11),

Corollary 1.1. *The difference between the original value function and the abstract value function is bounded by,*

$$0 \leq \bar{V}^\pi(b_t) - V^\pi(b_t) \leq \mathcal{T} \cdot \omega_2 log(K). \quad (12)$$

Proof. see appendix A.4. □

This result allows us to bound the loss when applying observation abstraction. In Section 4, we devise an algorithm that adapts the bounds so that the same best action will be chosen, with and without abstraction, while expediting planning time.

3.2 Continuous Observation Space

Planning with entropy as reward over a continuous observation space is more cumbersome as it requires calculation of,

$$\mathbb{E}[\mathcal{H}(b_t)] = -\int_{o_t} \mathbb{P}(o_t \mid H_t^-) \int_{s_t} b(s_t) log(b(s_t)). \quad (13)$$

First, the probability density function of the belief may be arbitrary, so the integral over the state has no closed-form expression. Moreover, usually, there is no access to the density functions due to the difficulty of exact Bayesian inference, but only to samples. Second, even if the differential entropy could be evaluated, integrating over the observation space makes this calculation intractable. Consequently, it is only feasible to estimate the value function, here denoted by $\hat{V}^\pi(\cdot)$. To approximate the posterior belief at each time step, we employ the commonly used particle filter, see e.g. [Thrun *et al.*, 2005]. Inspired by derivations in [Boers *et al.*, 2010], we estimate (13) using state samples obtained via a particle filter. We then sample observations using the given observation model, conditioned on the state particles. This is a common procedure in tree search planning, see for example [Sunberg and Kochenderfer, 2018]. Using the state and observation samples, we derive an estimator to (13),

$$\hat{\mathbb{E}}[\mathcal{H}(\hat{b}_t)] = -\hat{\eta}_t \sum_{m=1}^{M} \sum_{i=1}^{N} Z\left(o_t^m \mid s_t^i\right) q_{t-1}^i \cdot \quad (14)$$
$$log\left(\frac{Z\left(o_t^m \mid s_t^i\right) \sum_{j=1}^{N} T\left(s_t^i \mid s_{t-1}^j, a_{t-1}\right) q_{t-1}^j}{\sum_{i'=1}^{N} Z\left(o_t^m \mid s_t^{i'}\right) q_{t-1}^{i'}}\right),$$

where $\hat{b} \doteq \{q^i, s^i\}_{i=1}^N$ denotes the belief particles with weights q^i; M, N are the number of observation and state samples accordingly and $\hat{\eta}_t = \frac{1}{\sum_{m=1}^{M} \sum_{i=1}^{N} Z\left(o_t^m \mid s_t^i\right) q_{t-1}^i}$. See appendix A.5 for the full derivation, and [Boers *et al.*, 2010] for a discussion about convergence of the differential entropy estimator to the true differential entropy value. The estimator for the expected entropy of $\hat{b}_t$, (14), is also a function of $\hat{b}_{t-1}$, hence the reward structure $R(\hat{b}, a, \hat{b}')$.

Similar to the discrete case, we use an abstract observation model (4), where instead of discrete observations, summation is done over observation samples, see Figure 2. We obtain upper and lower bounds that resemble the results (8), (9) and (12) but depend on the approximate expected reward value. Combining results on the expected state-dependent reward and expected entropy,

Theorem 2. *The estimated expected reward is bounded by,*

$$0 \leq \hat{\bar{\mathbb{E}}}_o \left[R\left(\hat{b}, a, \hat{\bar{b}}'\right)\right] - \hat{\mathbb{E}}_o \left[R\left(\hat{b}, a, \hat{b}'\right)\right] \leq \omega_2 log(K). \quad (15)$$

Proof. see appendix A.6. □

Here, $\hat{\bar{b}}' \doteq \{\bar{q}^i, s^i\}_{i=1}^N$ denotes a particle set with abstract weight, $\bar{q}^i$, due to the abstract observation model (4), and

$$\hat{\bar{\mathbb{E}}}_o [\cdot] \doteq \sum_{m=1}^{M} \sum_{i=1}^{N} \bar{Z}(o^m \mid s^i) q_{t-1}^i [\cdot]. \quad (16)$$

$\hat{\mathbb{E}}_o[\cdot]$ and $\hat{b}'$ are defined similarly by replacing the abstract model with the original one. As a result of Theorem 2, the estimated value function is bounded,

Corollary 2.1. *The difference between the estimated value function and the abstracted value function bounded by,*

$$0 \leq \hat{\bar{V}}^{\pi}(b_t) - \hat{V}^{\pi}(b_t) \leq \mathcal{T} \cdot \omega_2 log(K). \quad (17)$$

Proof. see appendix A.7. □

The computational complexity of the expected reward in (15) is dominated by the complexity of the expected entropy, (14), which is $O(MN^2)$. By simply choosing $K = M$ the time complexity of the abstract expected reward diminishes to $O(N^2)$ with bounded loss. Utilizing our result directly induces a trade-off between computational speed and approximation loss of the value function. In the next section we derive an adaptive algorithm that gains computational efficiency *without any loss* in terms of the selected action.

4 Algorithms

Since the derivations in previous sections are agnostic to which algorithm is being used, we begin this section by presenting the contribution of our work to an existing algorithm. Then, based on insights gained from the examined algorithm, we propose modifications to improve the current algorithm. In the following section, we show that the changed algorithm empirically surpasses the current SOTA in performance throughout our experiments by a significant margin.

4.1 Baseline Algorithms

Sparse sampling (SS) algorithm, introduced in [Kearns *et al.*, 2002], provides ϵ-accuracy guarantee on the solution at a finite time. However, since it searches the tree exhaustively, the convergence is quite slow in practice. On the other hand, MCTS algorithm [Browne *et al.*, 2012] has the desirable property of focusing its search on the more promising parts of the tree, but was shown to have poor finite-time performance, requiring an $exp(exp(...exp(1)...))$[2] iterations in the worst-case scenario [Munos, 2014]. To combat the worst-case running time of MCTS and the slow running time of SS, Forward Search Sparse Sampling (FSSS) [Walsh *et al.*, 2010] was introduced. It was shown to achieve comparable performance to MCTS with performance guarantees under finite computational budget as in SS.

[2] A composition of D-1 exponentials, where D denotes tree depth.

Algorithm 1 AI-FSSS

Procedure: SIMULATE(b,d)

if d = 0 **then**
  return 0, 0
else if $|C(b)| < |\mathcal{A}|$ **then**
  $ba, a, o_{\{1,...,K\}} \leftarrow$GEN$(b, K)$
  $\bar{P}_{o|s} \leftarrow$ ABSTRACTOBS$(ba, o_{\{1,...,K\}})$ // eq.(4)
  $\bar{\mathbb{E}}[\mathcal{R}(ba)] \leftarrow$EXPECTEDREWARD$(b, a, ba, \bar{P}_{o|s})$
else
  $a \leftarrow$ SELECTACTION(b)
end if
$lb \leftarrow \bar{\mathbb{E}}[\mathcal{R}(ba)]$
$ub \leftarrow \bar{\mathbb{E}}[\mathcal{R}(ba)] + log(K)$
if $0 < N(ba) < K$ **then**
  $o \leftarrow$ POP $(o_{\{1,...,K\}})$
  $b' \leftarrow$ POSTERIOR(b, a, o)
  $V_{LB}, V_{UB} \leftarrow$SIMULATE$(b', d-1)$
else if $N(ba) = K$ **then**
  $b' \leftarrow \arg\min_{b'} N(b')$
  $V_{LB}, V_{UB} \leftarrow$SIMULATE$(b', d-1)$
else if $N(ba) = 0$ **then**
  $V_{LB}, V_{UB} \leftarrow$ROLLOUT$(ba, d-1)$
end if
$LB(ba) \leftarrow lb + \frac{V_{LB} + (|C(ba)|-1)(LB(ba)-lb)}{|C(ba)|}$
$UB(ba) \leftarrow ub + \frac{V_{UB} + (|C(ba)|-1)(LB(ba)-ub)}{|C(ba)|}$
$a^* \leftarrow \arg\max_a UB(ba)$
$LB(b) \leftarrow LB(ba^*)$
$UB(b) \leftarrow UB(ba^*)$
$N(b) \leftarrow N(b) + 1$
$N(ba) \leftarrow N(ba) + 1$
return $LB(b), UB(b)$

4.2 FSSS with Information-Theoretic Rewards

In its original version, FSSS introduced lower and upper bounds on the estimate of the $Q^d(s, a)$ function. In contrast to SS, FSSS builds the tree incrementally, where each iteration begins at the root node and proceeds down to horizon H, to obtain an estimate for the action-value function. Whenever a new node is expanded, its direct action-nodes are created alongside M randomly sampled children for each of them. The branching factor, M, is a predefined hyper-parameter. After performing $(|A| \cdot M)^d$ iterations, FSSS builds the same tree as SS, but enjoys anytime properties. Moreover, FSSS may benefit from reduced computation by utilizing upper and lower bounds and pruning actions that are sub-optimal.
When the reward is defined as an information-theoretic function, such as differential entropy, upper and lower bounds on the value function cannot be determined a-priori, thus no pruning can be made. Nonetheless, our experimental evaluations suggest that FSSS still serves as a strong baseline.

4.3 Adaptive Information-FSSS

We coin our new algorithm Adaptive Information FSSS (AI-FSSS). Given the same number of iterations, the actions obtained by the two algorithms are identical. The pseudo-code

Algorithm 2 SOLVE

Procedure: SOLVE

for $i \in 1:n$ **do**
  SIMULATE(b_{init}, d_{max})
end for
$action \leftarrow$ ADAPTBOUNDS(b_{init})
return $action$

Algorithm 3 REFINE

Procedure: REFINE(b,ba,d)

if IsLeaf(b) **then**
  return $0, 0$
else if ABSTRACT(ba) **then**
  $r_{old} \leftarrow$REUSEREWARD(ba)
  $P_{o|s} \leftarrow$ ORIGINALOBSMODEL$(ba, o_{\{1,\dots,K\}})$
  $\mathbb{E}[\mathcal{H}(ba)] \leftarrow$EXPECTEDENTROPY$(b, ba, P_{o|s})$
  $r \leftarrow r_{old} + \omega_2(\mathbb{E}[\mathcal{H}(ba)] - \bar{\mathbb{E}}[\mathcal{H}(ba)])$
else
  $r \leftarrow$REUSEREWARD(ba)
end if
$b' \leftarrow \arg\max_{b'}(UB(b') - LB(b'))$
$a' \leftarrow \arg\max_{a'}(UB(b'a') - LB(b'a'))$
$V_{LB}, V_{UB} \leftarrow$REFINE$(b', b'a', d-1)$
$LB(ba) \leftarrow lb + \frac{V_{LB} + (|C(ba)|-1)(LB(ba)-lb)}{|C(ba)|}$
$UB(ba) \leftarrow ub + \frac{V_{UB} + (|C(ba)|-1)(LB(ba)-ub)}{|C(ba)|}$
$a^* \leftarrow \arg\max_a UB(ba)$
$LB(b) \leftarrow LB(ba*)$
$UB(b) \leftarrow UB(ba*)$
return $LB(b), UB(b)$

Procedure: ADAPTBOUNDS(b_{init})

while $\max_{a^+ \in \mathcal{A}} LB(b_{init}a^+) < \max_{a \in \mathcal{A} \setminus a^+} UB(b_{init}a)$ **do**
  $a^* \leftarrow \arg\max_{a \in \mathcal{A}} LB(b_{init}a)$
  REFINE$(b_{init}, b_{init}a^*, d)$
end while
return a^*

presented in appendix B. As in FSSS, we build the tree incrementally, where each iteration adds a new trajectory to the tree. The algorithm constructs an abstract belief tree, where a set of K posterior beliefs share the same reward upper and lower bounds relating it to the underlying reward value. This is done by immediately sampling K observation samples whenever a new action node expanded, followed by a computation of the abstract reward. Based on Theorem 2 we derive an ADAPTBOUNDS procedure, that adapts the number of aggregated observations. Bounds adaptation halts whenever the highest lower bound, $\max_a LB(b_{init}a)$, is higher than the upper bound of any other action. This results in the same action selection for the full FSSS and our adaptation, AI-FSSS.

4.4 Introducing Rollouts to AI-FSSS

A direct adaptation of FSSS to AI-FSSS would abstract K observations in each new action node up to the full depth of the tree, d_{max}. However, when the time budget is limited, it might not be the best strategy, since the abstraction of deeper belief nodes of the tree may never be visited twice, but might need to be refined afterward. Instead, we propose to perform rollout whenever a new action node is met for the first time. This is similar to MCTS, where rollouts are used to get an estimate of the action-value function. This approach will lead to abstraction only for expanded nodes, which are the ones in proximity to the root node. As the number of iterations grows, action nodes are expanded gradually and more abstract belief nodes are added to the tree. Given that the number of iterations equals the number of action nodes in the original Sprase-Sampling tree, followed by ADAPTBOUNDS procedure, both algorithms converge to the same solution.

4.5 Implementation

In this section we present the main building blocks to derive our algorithm. The variables used in Algorithm 1 are b, ba, and b' which represent a belief node, a predicted belief node, i.e. after performing an action and posterior belief, after incorporating a measurement. $C(\cdot)$ denotes a list of their corresponding children. a and $o_{\{1,\dots,K\}}$ denote an action and a list of K sampled observations respectively. $\bar{P}_{o|s}$ is a list holding the abstract probability values of the measurement model, as in equation (4). $R_{state}(\cdot,\cdot)$ denotes a state-dependent reward function, which may be defined arbitrarily. LB, UB and N are all initialized to zero. ROLLOUT performs a predefined policy. In our experiments, we chose uniform distribution over all actions for the rollout policy. Algorithm 2 uses b_{init}, which represents the initial belief at the root node, n is the number of iterations and d_{max}, the maximum depth of the planning tree.

5 Experiments

The goal of the experiment section is to evaluate the influence of the abstraction mechanism on the planning performance. We examined both the time difference and the total return. All algorithms use a particle filter for inference, the choice of the particle filter variant is independent of our contribution. All experiments were performed on the common 2D Light Dark benchmark, where both the state and observation spaces are continuous; see an illustration in Figure 4. In this problem, the agent is required to reach the goal while reducing localization uncertainty using beacons scattered across the map. The reward function defined as a weighted sum of distance to goal, which is state dependent reward and entropy, as in (1). Due to space limitations, domain and implementation details are deferred to appendix C.

5.1 Time Performance Evaluation

We compared the basic FSSS with our adaptation, AI-FSSS, in terms of time efficiency. As stressed in previous sections, both algorithms guaranteed to select the same action. To ensure that both algorithms built the same tree, rollouts were avoided and each iteration proceeded until the maximum depth of the tree. We note that the expected return was inferior to our full algorithm, which is evaluated next. Technically, we also fixed the random numbers by selecting the same seed in both algorithms.

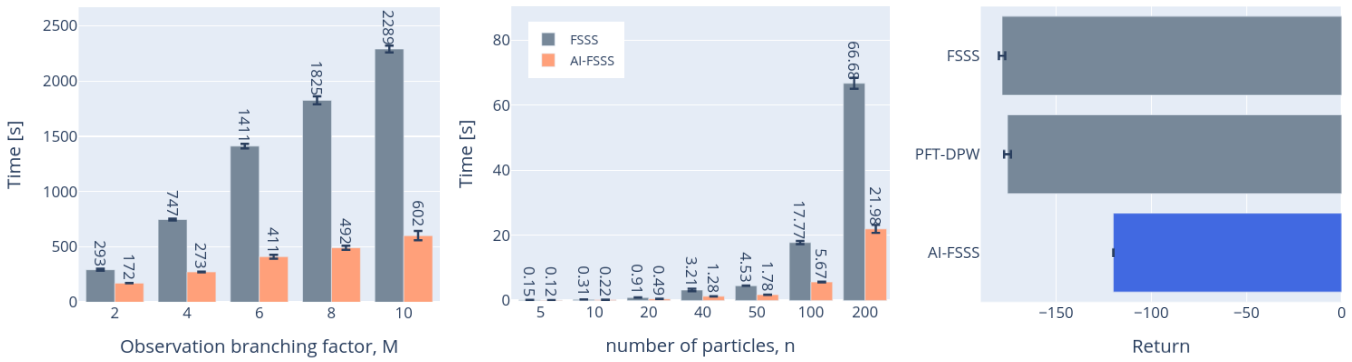

Figure 3: Evaluating performance of AI-FSSS. (a-b): Running time comparison of FSSS, and our adaptation, AI-FSSS without rollouts. Both algorithms end with the same action selection, but with increasing difference in computation time. (a) Different observation branching factor, with 20 particles. (b) Different number of state particles, with 4 observation after each action node. (c) Average total return of AI-FSSS with rollouts in 2D Light-Dark with obstacles.

Observation branching factor. In the first experiment, we fixed the number of state particles to $n = 40$, and examined the influence of different branching factors over the observation space, see Figure 3. The algorithms were limited to 20,000 iterations before performing an action. The computation time indicates an empirical average of over 1,000 simulations, approximating the mean time for a full trajectory. As one would expect, the more observations are clustered in each aggregate, the more time-efficient AI-FSSS compared to the basic FSSS. To obtain the *same* best action in both algorithms, the tree construction was followed by a refinement step, see appendix B, ADAPTBOUNDS. In the adapt-bounds step, we incrementally reduce the number of aggregates, so in the worst-case every aggregate will only hold a single observation, and thus recover the FSSS tree. This will occur only in a degenerate case, where all action-values $Q(b_0, \cdot)$ will have the same value, which is rarely the case.

Number of particles. In our second experiment, we evaluated the effect of the number of particles representing the belief. Here, the number of observations was fixed to $M = 4$. Figure 3 shows the change in computational speed with regard to the number of particles in our experiments. Both algorithms performed 1,000 simulations; The empirical running time mean and standard deviation are presented in the graph. In the experiments where only few particles were used, e.g. 5, the efficiency gain was mild. In a setting where few particles are sufficient, computing the entropy is relatively cheap compared other parts of the algorithm, which become relatively more significant (e.g. the different $\max$ operators). However, we observed that the burden became significant even in a mild number of particles, e.g. when $n = 20$ the speed-up ratio more than doubled while only a modest cluster size of 4 observations was used.

5.2 Total Return Evaluation

In contrast to the previous experiments, in this section we evaluate the full version of AI-FSSS, that is, with rollouts for every newly expanded action node. We evaluated performance against FSSS [Walsh *et al.*, 2010] and PFT-DPW

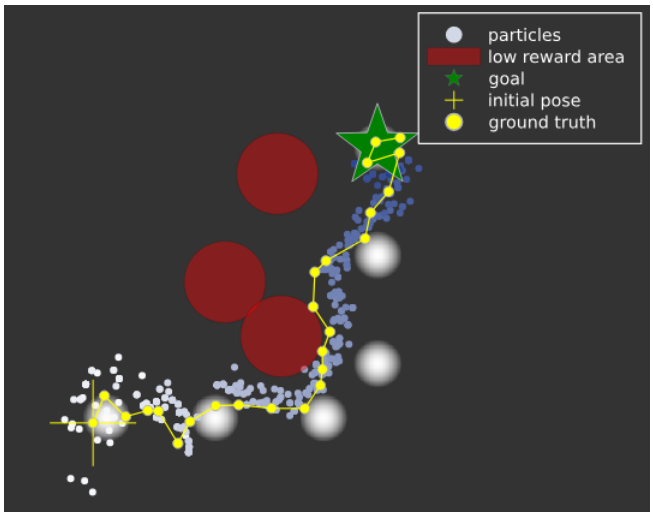

Figure 4: An illustration of the environment being used in our experiments.

[Sunberg and Kochenderfer, 2018], by augmenting a POMDP to a belief-MDP. In this setting, each node holds n particles. All algorithms had 1 second limitation for planning before each interaction with the environment. Except for PFT-DPW, the algorithms shared the same observation branching factor, $M = 4$. PFT-DPW opens new observation nodes as time progresses, depending on hyper-parameters defined by the expert. We identified k_o as the dominant hyper-parameter that controls the number of observations in our experiment. We experimented with both $k_o = 4$ and $k_o = 2$ in order to keep a comparable observation branching size. The better result is shown here, see Figure 3.

Each algorithm performed 1,000 full trajectories in the environment, each contained 25 steps. As suggested in Figure 3, our algorithm performed better than PFT-DPW and FSSS when information-gathering was an explicit part of the task. The superior results are expected due to the additional effi-

ciency of our approach. Both FSSS and PFT-DPW compute the expensive-to-evaluate reward value for every newly expanded node, whereas AI-FSSS compute the exact reward only when it is required in order to determine the best action selection. Consequently, under a given time-limit, AI-FSSS expands more posterior nodes in the belief tree, which result in better coverage of the belief tree.

6 Conclusions

This paper deals with online planning under uncertainty with information-theoretic reward functions. Information-theoretic rewards facilitate explicit reasoning about state uncertainty, contrary to the more common expected reward over the state. Due to the added computational burden of evaluating such measures, we consider an observation model abstraction that improves efficiency. We derived analytical bounds with respect to the original reward function. Additionally, we introduced a new algorithm, AI-FSSS, that contracts the bounds upon need, and is guaranteed to select identical action as the vanilla algorithm. Finally, we conducted an empirical performance study with and without observation abstraction. Our results suggest a significant speed-up as the cardinality of the particle set and the observation-branching factor increases while yielding same performance.

Acknowledgements

The research presented in this paper was partially funded by the Israel Science Foundation (ISF), by US NSF/US-Israel BSF, by the Israel Ministry of Science and Technology (MOST) and by the Israeli Smart Transportation Research Center (ISTRC).

References

[Araya *et al.*, 2010] Mauricio Araya, Olivier Buffet, Vincent Thomas, and Françcois Charpillet. A pomdp extension with belief-dependent rewards. In *Advances in Neural Information Processing Systems (NIPS)*, pages 64–72, 2010.

[Boers *et al.*, 2010] Y. Boers, H. Driessen, A. Bagchi, and P. Mandal. Particle filter based entropy. In *2010 13th International Conference on Information Fusion*, pages 1–8, 2010.

[Browne *et al.*, 2012] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[Burgard *et al.*, 1997] Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Active mobile robot localization. In *Intl. Joint Conf. on AI (IJCAI)*, pages 1346–1352. Citeseer, 1997.

[Fischer and Tas, 2020] Johannes Fischer and Omer Sahin Tas. Information particle filter tree: An online algorithm for pomdps with belief-based rewards on continuous domains. In *Intl. Conf. on Machine Learning (ICML)*, Vienna, Austria, 2020.

[Flaspohler *et al.*, 2019] Genevieve Flaspohler, Victoria Preston, Anna PM Michel, Yogesh Girdhar, and Nicholas Roy. Information-guided robotic maximum seek-and-sample in partially observable continuous environments. *IEEE Robotics and Automation Letters (RA-L)*, 4(4):3782–3789, 2019.

[Garg *et al.*, 2019] Neha P Garg, David Hsu, and Wee Sun Lee. Despot-α: Online pomdp planning with large state and observation spaces. In *Robotics: Science and Systems (RSS)*, 2019.

[Hoerger and Kurniawati, 2021] Marcus Hoerger and Hanna Kurniawati. An on-line pomdp solver for continuous observation spaces. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 7643–7649. IEEE, 2021.

[Indelman *et al.*, 2015] V. Indelman, L. Carlone, and F. Dellaert. Planning in the continuous domain: a generalized belief space approach for autonomous navigation in unknown environments. *Intl. J. of Robotics Research*, 34(7):849–882, 2015.

[Kearns *et al.*, 2002] Michael Kearns, Yishay Mansour, and Andrew Y Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. volume 49, pages 193–208. Springer, 2002.

[Kim and Eustice, 2014] A. Kim and R. M. Eustice. Active visual SLAM for robotic area coverage: Theory and experiment. *Intl. J. of Robotics Research*, 34(4-5):457–475, 2014.

[Miller, 2003] Erik G Miller. A new class of entropy estimators for multi-dimensional densities. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, volume 3, pages III–297. IEEE, 2003.

[Moon *et al.*, 2022] Brady Moon, Satrajit Chatterjee, and Sebastian Scherer. Tigris: An informed sampling-based algorithm for informative path planning, 2022.

[Munos, 2014] Rémi Munos. *From Bandits to Monte-Carlo Tree Search: The Optimistic Principle Applied to Optimization and Planning*. 2014.

[Papadimitriou and Tsitsiklis, 1987] C. Papadimitriou and J. Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[Platt *et al.*, 2010] R. Platt, R. Tedrake, L.P. Kaelbling, and T. Lozano-Pérez. Belief space planning assuming maximum likelihood observations. In *Robotics: Science and Systems (RSS)*, pages 587–593, Zaragoza, Spain, 2010.

[Silver and Veness, 2010] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2164–2172, 2010.

[Somani *et al.*, 2013] Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. Despot: Online pomdp planning with regularization. In *NIPS*, volume 13, pages 1772–1780, 2013.

[Spaan *et al.*, 2015] Matthijs TJ Spaan, Tiago S Veiga, and Pedro U Lima. Decision-theoretic planning under uncertainty with information rewards for active cooperative perception. *Autonomous Agents and Multi-Agent Systems*, 29(6):1157–1185, 2015.

[Sunberg and Kochenderfer, 2018] Zachary Sunberg and Mykel Kochenderfer. Online algorithms for pomdps with continuous state, action, and observation spaces. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 28, 2018.

[Sztyglic and Indelman, 2021] Ori Sztyglic and Vadim Indelman. Online pomdp planning via simplification. *arXiv preprint arXiv:2105.05296*, 2021.

[Sztyglic *et al.*, 2021] O. Sztyglic, A. Zhitnikov, and V. Indelman. Simplified belief-dependent reward mcts planning with guaranteed tree consistency. Technical report, 2021.

[Thomas *et al.*, 2021] Vincent Thomas, Geremy Hutin, and Olivier Buffet. Monte carlo information-oriented planning. *arXiv preprint arXiv:2103.11345*, 2021.

[Thrun *et al.*, 2005] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT press, Cambridge, MA, 2005.

[Van Den Berg *et al.*, 2012] J. Van Den Berg, S. Patil, and R. Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *Intl. J. of Robotics Research*, 31(11):1263–1278, 2012.

[Walsh *et al.*, 2010] Thomas Walsh, Sergiu Goschin, and Michael Littman. Integrating sample-based planning and model-based reinforcement learning. In *Nat. Conf. on Artificial Intelligence (AAAI)*, volume 24, 2010.

[Ye *et al.*, 2017] Nan Ye, Adhiraj Somani, David Hsu, and Wee Sun Lee. Despot: Online pomdp planning with regularization. *JAIR*, 58:231–266, 2017.