

nent 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 as a state-of-the-art (SOTA) algorithm, however, it is limited to a 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 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 of 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 as heuristic over state-dependent objective function for information gathering.

More closely related to our work is [Thomas *et al.*, 2021], which interleaves 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 estimating the reward function using KDE, which scales quadratically with the number of state samples. [Szttyglic 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, [Szttyglic *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 an 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. 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, a 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 for 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 POMDP 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. $T(s' | s, a)$ is the transition density function, $Z(o | s)$ 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 $(a_0, o_1, \dots, a_{t-1}, o_t)$ up to current time. We use H_t^- to denote the same history, without the last observation, i.e. $H_t^- = (a_0, o_1, \dots, a_{t-1})$. 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,

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

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,

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

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

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

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