Original Article

The International Journal of Robotics Research
2024, Vol. 0(0) 1–39
© The Author(s) 2024
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/02783649241261398
journals.sagepub.com/home/ijr
Sage

No compromise in solution quality: Speeding up belief-dependent continuous partially observable Markov decision processes via adaptive multilevel simplification

Andrey Zhitnikov[1], Ori Sztyglic[2] and Vadim Indelman[3]

Abstract
Continuous Partially Observable Markov Decision Processes (POMDPs) with general belief-dependent rewards are notoriously difficult to solve online. In this paper, we present a complete provable theory of adaptive multilevel simplification for the setting of a given externally constructed belief tree and Monte Carlo Tree Search (MCTS) that constructs the belief tree on the fly using an exploration technique. Our theory allows to accelerate POMDP planning with belief-dependent rewards without any sacrifice in the quality of the obtained solution. We rigorously prove each theoretical claim in the proposed unified theory. Using the general theoretical results, we present three algorithms to accelerate continuous POMDP online planning with belief-dependent rewards. Our two algorithms, SITH-BSP and LAZY-SITH-BSP, can be utilized on top of any method that constructs a belief tree externally. The third algorithm, SITH-PFT, is an anytime MCTS method that permits to plug-in any exploration technique. All our methods are guaranteed to return exactly the same optimal action as their unsimplified equivalents. We replace the costly computation of information-theoretic rewards with novel adaptive upper and lower bounds which we derive in this paper, and are of independent interest. We show that they are easy to calculate and can be tightened by the demand of our algorithms. Our approach is general; namely, any bounds that monotonically converge to the reward can be utilized to achieve a significant speedup without any loss in performance. Our theory and algorithms support the challenging setting of continuous states, actions, and observations. The beliefs can be parametric or general and represented by weighted particles. We demonstrate in simulation a significant speedup in planning compared to baseline approaches with guaranteed identical performance.

Keywords
Decision-making under uncertainty, belief space planning, partially observable Markov decision processes, belief-dependent rewards, planning with imperfect information

Received 25 June 2023; Revised 16 January 2024; Accepted 21 May 2024

1. Introduction

Efficiently solving Partially Observable Markov Decision Processes (POMDPs) implies enabling autonomous agents and robots to plan under uncertainty (Garg et al., 2019; Kurniawati et al., 2008; Silver and Veness, 2010; Smith and Simmons, 2004; Sunberg and Kochenderfer, 2018; Ye et al., 2017). Typical sources of uncertainty are the imprecise actions, sensor type, sensor noise, imprecise models, and unknown agent surroundings. However, solving a POMDP is notoriously hard. Specifically, it was proven to be PSPACE-complete (Papadimitriou and Tsitsiklis, 1987).

The actual POMDP state is hidden. Instead, at each time step, the robot shall decide which action to take based on the distribution over the state, given the corresponding history of performed actions and observations received so far. Such a distribution received the name "belief." In a planning session, the robot has to take into account all possible future actions interleaved with possible observations. Each such future history of the length of predefined horizon defines a lace of the future beliefs (blue lace in Figure 1) and corresponding cumulative rewards named return. Solving

[1]Technion Autonomous Systems Program (TASP), Technion - Israel Institute of Technology, Haifa, Israel
[2]Department of Computer Science, Technion - Israel Institute of Technology, Haifa, Israel
[3]Department of Aerospace Engineering, Technion - Israel Institute of Technology, Haifa, Israel

Corresponding author:
Andrey Zhitnikov, Technion Autonomous Systems Program (TASP), Technion - Israel Institute of Technology, Haifa 3200003, Israel.
Email: andreyz@campus.technion.ac.il

POMDP in the most common sense means finding a mapping from belief to action called policy, which maximizes the expected return.

Earlier *offline* solvers such as Kurniawati et al. (2008) and Smith and Simmons (2004) are applicable to small or moderately sized discrete POMDPs. These methods require passage over all possible states and observations (Kochenderfer et al., 2022) since they are built on value iteration of α-vectors, so called full-width methods (Silver and Veness, 2010). More recent *online* solvers are suitable for POMDPs with large but discrete action, state, and observation spaces (Silver and Veness, 2010; Ye et al., 2017). Still, continuous state, action, and observation spaces remain to be an open problem (Sunberg and Kochenderfer, 2018). Another challenging aspect of solving POMDP and the subject of interest in this paper is general belief distributions represented by weighted particles. Further in the manuscript we will regard the combination of both, nonparametric beliefs and a fully continuous POMDP as a **nonparametric fully continuous** setting.

In a fully continuous setting with parametric or general beliefs one shall resort to sampling of future possible actions and observations. In a sampled form, this abundance of possible realizations of action–observation pairs constitutes a *belief tree*. Building the full belief tree is intractable since each node in the tree repeatedly branches with all possible actions and all possible observations as illustrated in Figure 1. The number of nodes grows exponentially with the horizon. This problem is known as the *curse of history.*

The reward function in a classical POMDP is assumed to have a specific structure, namely, to be the expectation with respect to the belief of the state-dependent reward function. While alleviating the solution, this formulation does not support more general, belief-dependent reward functions, such as information-theoretic rewards.

However, POMDP planning with belief-dependent rewards is essential for various problems in robotics and Artificial Intelligence (AI), such as informative planning (Hollinger and Sukhatme, 2014), active localization (Burgard et al., 1997), active Simultaneous Localization and Mapping (SLAM) (Stachniss et al., 2005), and Belief Space Planning (BSP) (Indelman et al., 2015; Platt et al., 2010; Van Den Berg et al., 2012). Araya et al. (2010) provide an extensive motivation for general belief-dependent rewards. One of the widely used such rewards is Information Gain, which involves the difference between differential entropies of two consecutive in time beliefs. Such a reward is crucial in exploration tasks because, in these tasks, the robot's goal is to decrease uncertainty over the belief. For instance, uncertainty measures such as differential entropy and determinant of the covariance matrix of the belief cannot be represented as expectation over a state-dependent reward with respect to the belief. Another example of a belief-dependent reward is entropy over discrete variables that correspond to data association hypotheses (Pathak et al., 2018). Computationally-efficient information-theoretic BSP approaches have been investigated in recent years, considering Gaussian distributions (Elimelech and Indelman, 2022; Kitanov and Indelman, 2024; Kopitkov and Indelman, 2017, 2019).

Yet, POMDP planning with general belief-dependent rewards in particular, when the beliefs are represented by particles exacerbate the computational challenge of the solution even more. For example information theoretic rewards such as differential entropy, are computationally expensive.

Let us focus for the moment on differential entropy. Even if the belief is parametric but not Gaussian, calculating the exact value of differential entropy involves intractable integrals. This fact also motivates to use a weighted particles representation for the belief. In this case differential entropy

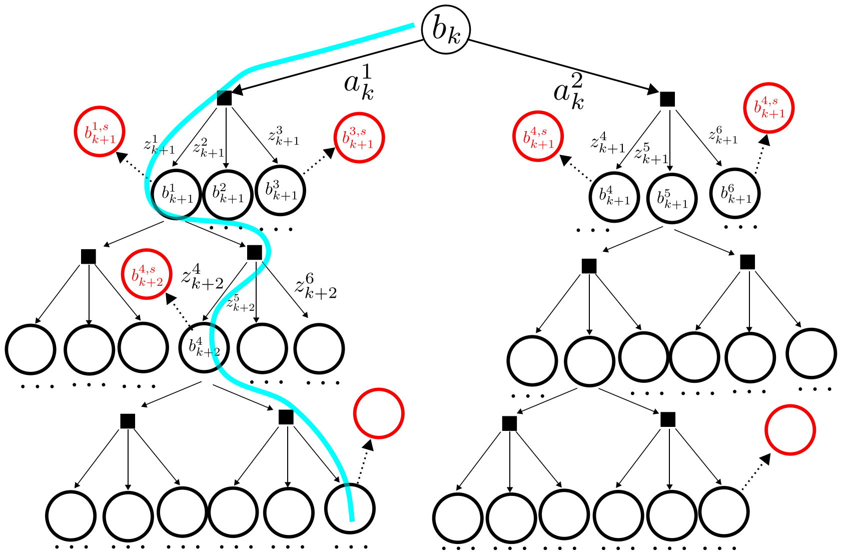


Figure 1. Schematic visualization of the belief tree and the inplace simplification. The superscript in this visualization denotes the index in the belief tree. By b^s we denote the simplified version of the belief b.

can be estimated, for instance by Kernel Density Estimation (KDE) (Fischer and Tas, 2020) or a model-based estimator (Boers et al., 2010). However, these estimators have quadratic cost in the number of samples and are usually the bottleneck of planning algorithms. The reason is that this increased computational burden is incurred for all nodes in the belief tree. Importantly, the estimation errors of these estimators with respect to differential entropy over theoretical belief are out of the reach due to the unavailability of both, the theoretical belief and the entropy on top of it. Yet, due to the convergence of the belief represented by particles to the theoretical belief (almost sure convergence (Crisan and Doucet, 2002)), the mentioned above estimators converge to the exact differential entropy. This prompts us to use **as many belief particles as possible** to get closer to the theoretical belief. Nevertheless, increasing the number of belief particles severely impacts planning time.

In this paper, we accelerate online decision making in the setting of nonparametric fully continuous POMDPs with general belief-dependent rewards. Crucially, planning performance of our accelerated approach is the same as that of the baseline approaches without our acceleration. Before stating our contributions, we review the most relevant works in this context.

1.1. Related work

Allowing general belief-dependent rewards in POMDP while solving such a problem efficiently is a long standing effort. Some previous seminal works such as ρ-POMDP (Araya et al., 2010; Fehr et al., 2018) as well as Dressel et al. (2017) have focused on discrete domains, small sized spaces and have tackled the offline solvers. Furthermore, these approaches are limited to piecewise linear and convex or Lipschitz-continuous rewards. Another work named POMDP-IR (Spaan et al., 2015) suggests an interesting framework for specific form of information rewards involving manipulations on the action space. Still, in Araya et al. (2010), Dressel et al. (2017), and Fehr et al. (2018), the state, action, and observation spaces are discrete and small sized. Another line of works is Belief Space Planning (BSP) (Indelman et al., 2015; Platt et al., 2010; Van Den Berg et al., 2012). These approaches are designed for fully continuous POMDPs, but limited to Gaussian beliefs. In striking contrast, our approach is centered in the more challenging fully continuous domain and nonparametric general beliefs represented by particles while at the same time our framework is general and supports also exact parametric beliefs.

One way to tackle a nonparametric fully continuous setting with belief-dependent rewards is to reformulate POMDP as a Belief-MDP (BMDP). On top of this reformulation one can utilize MDP sampling-based methods such as Sparse Sampling (SS) proposed by Kearns et al. (2002). However, this algorithm still suffers from the curse of history and such that increasing the horizon is still problematic.

Monte Carlo Tree Search (MCTS) made a significant breakthrough in overcoming the course of history by building the belief tree incrementally and exploring only the "promising" parts of the tree using the exploration strategy. An inherent part of MCTS-based algorithms is the exploration strategy designed to balance exploration and exploitation while building the belief tree. Most widely used exploration technique is Upper Confidence Bound (UCB) (Kocsis and Szepesvári, 2006).

MCTS algorithms assume that calculating the reward over the belief node does not pose any computational difficulty. Information-theoretic rewards violate this assumption. When the reward is a general function of the belief, the origin of the computational burden is shifted towards the reward calculation. Moreover, in a nonparametric setting, belief-dependent rewards require a complete set of belief particles at each node in the belief tree. Therefore, algorithms such as POMCP (Silver and Veness, 2010) and its numerous predecessors are inapplicable since they simulate each time a single particle down the tree when expanding it. DESPOT-based algorithms behave similarly (Ye et al., 2017), with the DESPOT-α as an exception (Garg et al., 2019). DESPOT-α simulates a complete set of particles. However, the DESPOT-α tree is built using α-vectors, such that they are an indispensable part of the algorithm. The standard α-vectors technique requires that the reward is state dependent, and the reward over the belief is merely expectation over the state reward. In other words, DESPOT-α does not support belief-dependent rewards since it contradicts the application of the α-vectors.

The only approach posing no restrictions on the structure of belief-dependent reward and not suffering from limiting assumptions is Particle Filter Tree (PFT). The idea behind PFT is to apply MCTS over Belief-MDP (BMDP). Sunberg and Kochenderfer (2018) augmented PFT with Double Progressive Widening (DPW) to support continuous spaces in terms of actions, states, and observations, and coined the name PFT-DPW. PFT-DPW utilizes the UCB strategy and maintains a complete belief particle set at each belief tree node. Recently, Fischer and Tas (2020) presented Information Particle Filter Tree (IPFT), a method to incorporate information-theoretic rewards into PFT. The IPFT simulates small subsets of particles sampled from the root of the belief tree and averages entropies calculated over these subsets, enjoying a fast runtime. However, differential entropy estimated from a small-sized particle set can be significantly biased. This bias is unpredictable and unbounded, therefore, severely impairs the performance of the algorithm. In other words, celerity comes at the expense of quality. Oftentimes, the policy defined by this algorithm is very far from optimal given a time budget. Fischer and Tas (2020) provide guarantees solely for the asymptotic case, that is, the number of subsampled from the root belief state samples (particles) tends to infinity. Asymptotically their algorithm behaves precisely as the PFT-DPW in terms of running speed and performance. Yet, in practice the performance of IPFT in terms of optimality can degrade severely compared to PFT-DPW. Moreover, Fischer and Tas (2020) do not provide any study of comparison of IPFT against PFT-DPW with an information-theoretic reward. Another undesired characteristic of IPFT is that the averaging of the differential entropies is done implicitly and the number of averaged entropies per belief is the visitation count of the corresponding belief. Therefore, to properly compare IPFT with PFT-DPW one shall increase the number of simulations inside the IPFT algorithm. We explain this aspect more thoroughly in Section 8.3.5. Prompted by these insights, we chose the PFT-DPW as our *baseline* approach, which we aim to accelerate. In contrast to IPFT designed specifically for differential entropy, our approach is suitable for any belief-dependent reward and explicitly guarantees an *identical* solution to PFT-DPW with an information-theoretic reward, for *any* size of particle set representing the belief and serving as input to PFT-DPW.

The computational burden incurred by the complexity of POMDP planning inspired many research works to focus on approximations of the problem on top of existing solvers, for example, multilevel successive approximation of a motion model (Hoerger et al., 2019), lazy belief extraction on top of a particle-based representation (Hoerger and Kurniawati, 2021), linearity-based solvers (Hoerger et al., 2020), and averaging differential entropy estimated from tiny subsets of particles (Fischer and Tas, 2020). Typically, these works provide only asymptotical guarantees (Fischer and Tas, 2020; Hoerger et al., 2019) or no guarantees at all. In addition many of these approximations leverage the assumption that the belief-dependent reward is an averaged state-dependent reward (e.g, Hoerger et al., 2019; Hoerger and Kurniawati, 2021), and therefore cannot accommodate belief-dependent rewards with general structure (e.g., do not support information-theoretic rewards such as differential entropy).

Recently, the novel paradigm of *simplification* has appeared in literature (Barenboim and Indelman, 2022, 2023; Elimelech and Indelman, 2022; Kitanov and Indelman, 2024; Lev-Yehudi et al., 2024; Shienman and Indelman, 2022; Sztyglic and Indelman, 2022; Zhitnikov and Indelman, 2022b, 2024). The simplification is concerned with carefully replacing the nonessential elements of the decision-making problem and quantifying the impact of this relaxation. Specifically, simplification methods are accompanied by stringent guarantees. A prominent aspect of a simplification paradigm is the usage of the bounds over the reward or the objective function. As opposed to approximations, the simplification framework always keeps some sort of connection to the original unsimplified problem and by that provides deterministic guarantees relative to the given solver. Despite that various objective function bounds have been practiced in Kochenderfer et al. (2022), Smith and Simmons (2004), Walsh et al. (2010), and Ye et al. (2017), these techniques are not applicable in the realm of belief-dependent rewards and a fully continuous setting. In addition, commonly these approaches assume that the state dependent reward is trivially bounded from below and above by some constant.

1.2. Contributions

This work is about accelerating online decision making while obtaining exactly the same solution as without acceleration. Specifically, we contribute an adaptive multilevel simplification framework that accounts for belief-dependent rewards, possibly nonparametric beliefs, and continuous state, observation and action spaces.

Our framework accepts as input adaptive monotonical and computationally inexpensive bounds over the exact or estimated reward. Given such reward bounds, it accelerates online decision making. Specifically, given such adaptive monotonical reward bounds, it is possible to adaptively bound the value function for any given policy and expedite decision making. If the value function bounds for different candidate policies do not overlap, we do not pay in terms of quality, namely, we obtain the same solution as the equivalent unsimplified method. In the case these bounds do overlap, then we can progressively tighten them by invoking a process that we shall call simplification adaptation or *resimplification* until they no longer overlap.

Our techniques return exactly the same solution as the unsimplified equivalent. Such an unsimplified baseline can correspond to decision-making problems where the reward can be exactly calculated (analytically), or where the reward is estimated. In either case, if the bounds over the corresponding reward are provided and satisfy the assumptions stated in Section 3.3, one can apply our framework to speed up the decision-making process while obtaining the same best action as with the original rewards instead of the bounds. Such a capability is therefore particularly appealing in light of the information-theoretic rewards that are essential in numerous problems in robotics, but are often the computational bottleneck.

Further, there are two settings that we separately and explicitly discuss in this paper. We start from a given belief tree, that can be constructed by a POMDP solver that is not coupled with the solution, for example, SS. In this setting, we can prune branches of the belief tree whenever the mentioned objective bounds for different candidate policies or actions do not overlap.

We then discuss an anytime setting of MCTS, where the belief tree construction is coupled with the solution due to an exploration strategy (e.g., UCB). The exploration strategy builds upon an exploratory objective. Since the exploratory objective typically requires access to the objective estimates to select an action at each arrival to a belief node, we cannot prune suboptimal candidate actions. Instead, we can only dismiss them until the next arrival to this belief node. The simplification and reward bounds are used here to bound the exploratory objective and the value function at the root of the belief tree.

Finally, we focus on a specific simplification of nonparametric beliefs represented by particles and a differential entropy estimator as the reward function. Our simplification is subsampling of the original belief to a smaller sample size. We contribute novel computationally cheaper bounds over the differential entropy estimator on top of such a simplified belief and incorporate these bounds into our framework. By that, we produce a specific embodiment of the general framework presented earlier.

To summarize, we list down the contributions of this work, in the order they are presented in the manuscript.

1. Building on **any** adaptive monotonically convergent bounds over belief-dependent reward, we present in this paper a **provable** general theory of adaptive multilevel simplification with deterministic performance guarantees.
2. For the case of a given belief tree as in Sparse Sampling, we develop two algorithms, Simplified Information Theoretic Belief Space Planning (SITH-BSP) and a faster variant, LAZY-SITH-BSP. Both are complementary to any POMDP solver that does not couple belief tree construction with an objective estimation while exhibiting a significant speedup in planning with a guaranteed same planning performance.
3. In the context of MCTS, we embed the theory of simplification into the PFT-DPW algorithm and introduce SITH-PFT. We provide stringent guarantees that exactly the same belief tree is constructed by SITH-PFT and PFT-DPW. We focus on a UCB exportation technique, but with minor adjustments, an MCTS with any exploration method will be suitable for acceleration.
4. We derive novel lightweight adaptive bounds on the differential entropy estimator of (Boers et al., 2010) and prove the bounds presented are monotonic and convergent. Moreover, these bounds can be incrementally tightened. We believe these bounds are of interest on their own. The bounds are calculated using the simplified belief (See Figure 1). We emphasize that any other bounds fulfilling assumptions declared in Section 3.3 can be utilized within our framework.
5. We present extensive simulations that exhibit a significant improvement in planning time without any sacrifice in planning performance.

This paper is an extension of the work presented in Sztyglic and Indelman (2022), which proposed novel adaptive bounds on the differential entropy estimator of Boers et al. (2010) and introduced the simplification paradigm in the context of a given belief tree. To be precise we explicitly clarify how this work differs from the conference version of this paper (Sztyglic and Indelman, 2022). In this version, we extend the simplification framework to the rewards depending on a pair of consecutive-in-time beliefs, for example, Information Gain as opposed to the conference version where such an extension was only mentioned. In this version, we provide alternative proof of these bounds and prove that these reward bounds are monotonic. In the setting of a given belief tree, we present an additional algorithm, that we call LAZY-BSP. This algorithm is faster than SITH-BSP suggested in Sztyglic and Indelman (2022). Importantly, we extend our simplification framework to support also anytime MCTS planners. Additionally, we provide extensive performance evaluation of our methods in simulations.

1.3. Paper organization

The remainder of this paper is structured as follows. Section 2 provides background in terms of POMDPs, theoretical objective and commonly used objective estimators. We devote Section 3 to our general adaptive multi-level simplification framework. In Section 4, we consider a given belief tree setting in which the belief tree construction is not coupled with the solution. In Section 5, we delve into the MCTS approach in the context of our multilevel simplification. In Section 6, we consider a specific simplification and develop novel bounds on an information-theoretic reward function. Section 7 assesses the general adaptation overhead of our methodology. Finally, Section 8 presents simulations and results corroborating our ideas. In order not to disrupt the flow of the presentation, proofs are presented in appropriate Appendices.

2. Background

In this section, we present the background. To elaborate, we present a POMDP with belief-dependent rewards followed by theoretical and estimated objectives that correspond to different online POMDP solvers. Our techniques work with estimated objectives.

2.1. POMDPs with belief-dependent rewards

A POMDP is a tuple

$$\langle \mathcal{X}, \mathcal{A}, \mathcal{Z}, T, \mathcal{O}, \rho, \gamma, b_0 \rangle, \tag{1}$$

where $\mathcal{X}, \mathcal{A}, \mathcal{Z}$ are state, action, and observation spaces, respectively. In this paper, we consider continuous state, observation and action spaces. $T(x, a, x') = \mathbb{P}_T(x'|x, a)$ is the stochastic transition model from the past state x to the subsequent x' through action a, $\mathcal{O}(z, x) = \mathbb{P}_O(z|x)$ is the stochastic observation model, $\gamma \in (0, 1]$ is the discount factor, b_0 is the belief over the initial state (prior), and ρ is the reward function. Let $h_k = \{b_0, a_0, z_1, \ldots, a_{k-1}, z_k\}$ denote *history* of actions and observations obtained by the agent up to time instance k and the prior belief. The posterior belief at time instant k is given by $b_k(x_k) = \mathbb{P}(x_k|h_k)$.

In our generalized formulation, the reward is a function of two subsequent in time beliefs, an action and an observation:

$$\rho(b_k, a_k, z_{k+1}, b_{k+1}) = (1 - \lambda)\rho^x(b_k, a_k, b_{k+1}) + \tag{2}$$

$$+\lambda\rho^I(b_k, a_k, z_{k+1}, b_{k+1}), \tag{3}$$

where $\lambda \geq 0$. The first reward component $\rho^x(b_k, a_k, b_{k+1})$ is the expectation over the state and action-dependent reward r

(x_k, a_k) or r (a_k, x_{k+1}). Correspondingly, these two possibilities yield

$$\rho^x(b_k,a_k,b_{k+1}) = \underset{x_k \sim b_k}{\mathbb{E}}[r(x_k,a_k)] \approx \frac{1}{n_x}\sum_{\xi=1}^{n_x} r\left(x_k^{\xi},a_k\right), \quad (4)$$

or

$$\rho^x(b_k,a_k,b_{k+1}) = \underset{x_{k+1} \sim b_{k+1}}{\mathbb{E}}[r(a_k,x_{k+1})] \approx \frac{1}{n_x}\sum_{\xi=1}^{n_x} r\left(a_k,x_{k+1}^{\xi}\right). \quad (5)$$

which is commonly approximated by sample mean using n_x samples of the belief. The second reward component $\rho^I(b_k,a_k,z_{k+1}b_{k+1})$ is an information-theoretic reward weighted by λ, which in general can be dependent on consecutive beliefs and the elements relating them, for example, information gain or specific estimators as Boers et al. (2010) for nonparametric beliefs represented by particles. For instance, in Section 6.1, we consider the entropy estimator introduced by Boers et al. (2010). As will be seen in the sequel, although the theoretical entropy is only a function of a single belief b_{k+1}, the mentioned estimator utilizes b_k, a_k, z_{k+1}, and b_{k+1}; hence the second reward component, $\rho^I(b_k,a_k,z_{k+1}b_{k+1})$, depends on these quantities.

The *policy* is a mapping from belief to action spaces $a_k = \pi_k$ (b_k). Let $\pi_{\ell+}$ be a shorthand for policy for $\ell - k + L$ consecutive steps ahead starting at index ℓ, namely, $\pi_{\ell:k+L-1}$ for $\ell \geq k$.

2.2. Theoretical objective

The decision-making goal is to find an optimal policy π_{k+} maximizing the value function as such:

$$V(b_k,\pi_{k+}) \text{ s.t. } b_{\ell+1} = \psi(b_\ell,\pi_\ell(b_\ell),z_{\ell+1}), \quad (6)$$

where V (b_k, π_k) is defined by

$$\underset{z_{k+1:k+L}}{\mathbb{E}}\left[\sum_{\ell=k}^{k+L-1}\gamma^{\ell-k}\rho(b_\ell,\pi_\ell(b_\ell),z_{\ell+1},b_{\ell+1})\middle| b_k,\pi_{k+}\right] \quad (7)$$

and ψ is the Bayesian belief update method. Utilizing the Bellman formulation (7) takes the form of

$$\begin{aligned} V(b_k,\pi_{k+}) &= \underset{z_{k+1}}{\mathbb{E}}[\rho(b_k,\pi_k(b_k),z_{k+1},b_{k+1})|b_k,\pi_k] \\ &+\gamma\underset{z_{k+1}}{\mathbb{E}}[V(\psi(b_k,a_k,z_{k+1}),\pi_{(k+1)+})|b_k,\pi_k]. \end{aligned} \quad (8)$$

The action-value function under arbitrary policy is given by

$$\begin{aligned} Q(b_k,\{a_k,\pi_{(k+1)+}\}) &= \underset{z_{k+1}}{\mathbb{E}}[\rho(b_k,a_k,z_{k+1},b_{k+1})|b_k,a_k] \\ &+\gamma\underset{z_{k+1}}{\mathbb{E}}[V(\psi(b_k,a_k,z_{k+1}),\pi_{(k+1)+})|b_k,a_k]. \end{aligned} \quad (9)$$

The relation between (8) and (9) is V (b_k, π_{k+}) = Q (b_k, {π_k (b_k), $\pi_{(k+1)+}$}). If π is the optimal policy, we denote it by π^*. For clarity, let us designate for action-value function under optimal future policy $Q(b_k,\{a_k,\pi^*_{(k+1)+}\})$ a short notation Q (b_k, a_k). If Q (b_k, a_k) can be calculated, the online POMDP solution for the current belief b_k will be

$$\pi_k^*(b_k) \in \underset{a_k}{\text{argmax}}\, Q(b_k,a_k). \quad (10)$$

Linearity of the expectation and the structure displayed by equations (2) and (3) lead to a similar decomposition of action-value function (9) as such

$$Q(\cdot) = (1-\lambda)Q^x(\cdot) + \lambda Q^I(\cdot), \quad (11)$$

where Q^x is induced by state-dependent rewards and Q^I by the information-theoretic rewards.

From here on, for the sake of clarity, we will use the notation of history h_k and the belief b_k interchangeably for any time k. In a similar manner, we shall use the notations b_k, a_k and $h_k a_k$ interchangeably.

2.3. Estimated objective

The continuous observation space makes the theoretical expectations in (7) and (9) attainable in very limited cases. Generally we shall resort to estimators. Similar to theoretical counterparts, the relation between the estimated optimal value and action-value function reads

$$\widehat{V}(b_k,\pi^*_{k+}) = \max_{a_k}\widehat{Q}(b_k,a_k). \quad (12)$$

Also in equation (10), the theoretical Q (b_k, a_k) is substituted by the estimator $\widehat{Q}(b_k,a_k)$. Naturally, we expect from the estimator to admit the decomposition

$$\widehat{Q}(b_k,a_k) = (1-\lambda)\widehat{Q}^x(b_k,a_k) + \lambda\widehat{Q}^I(b_k,a_k). \quad (13)$$

Typically the $\widehat{Q}^x$ element is easy to calculate, thus it is out of our focus, whereas $\widehat{Q}^I$ is computationally expensive to compute.

Below we present two common sample based estimators that will be used in this paper.

2.3.1.Objective estimator in case of a given belief tree. We turn to the setting of a given externally-constructed belief tree, for example, by a SS algorithm. For the sake of clarity and to ease the explanation, we assume that the number of child posterior beliefs is constant at each nonterminal belief and denoted by n_z. Relaxing this assumption is straightforward. The Bellman form representation of (7) using such an estimator is

$$\begin{aligned} \widehat{V}(b_k,\pi_{k+}) &= \frac{1}{n_z}\sum_{i=1}^{n_z}\rho\left(b_k,\pi_k(b_k),z^i_{k+1},b^i_{k+1}\right) \\ &+\gamma\frac{1}{n_z}\sum_{i=1}^{n_z}\widehat{V}\left(\psi\left(b_k,\pi_k(b_k),z^i_{k+1}\right),\pi_{(k+1)+}\right), \end{aligned} \quad (14)$$

and the corresponding estimator for (9) under an optimal future policy reads

$$\widehat{Q}(b_k, a_k) = \frac{1}{n_z}\sum_{i=1}^{n_z}\rho\left(b_k, a_k, z^i_{k+1}, b^i_{k+1}\right) + \gamma\frac{1}{n_z}\sum_{i=1}^{n_z}\widehat{V}\left(\psi\left(b_k, a_k, z^i_{k+1}\right), \pi^*_{(k+1)+}\right), \tag{15}$$

where n_z is the number of children of b_ℓ under the execution policy $\pi_{\ell+}$ and i is the child index.

2.3.2. Interchangeability between the history and belief. The purpose of this section is to clarify why further we will use interchangeably belief and the history. The belief is merely a reinterpretation of the knowledge about the POMDP state stored in history in the form of a PDF. The belief b_k is a function of the history h_k. Therefore, different histories may yield the same belief. To avoid ambiguity and relate the objectives and their position in the belief tree with some abuse of notation, we sometimes switch the dependence on the belief to dependence on corresponding history. In general, we can write $b_\ell\,(h_\ell)$.

2.3.3. Coupled action-value function estimation and belief tree construction. The estimator presented above leverages symmetric, in terms of observations, Bellman form. However, in MCTS methods, due to exploration driven by, for example, UCB (16), the estimators are assembled from laces of the returns. In each simulation, a single lace is added to the estimator at each posterior belief. Whenever a new posterior belief node is expanded, a rollout is commenced such that the lace is complemented to the whole horizon.

MCTS repetitively descends down the tree, adding a lace of cumulative rewards (or updates visitation counts of an existing lace) and ascends back to root. On the way down it selects actions according to an exploration strategy, for example, (16). This results in a policy tree that represents a stochastic policy represented by visitation counts $N\,(ha)/N(h)$. Further we will focus on UCB exploration strategy, however, all derivations of our approach are general and are valid for any exploration strategy, for example, P-UCT (Auger et al., 2013) or ϵ-greedy exploration (Sutton and Barto, 2018).

A UCB-based MCTS over a Belief-MDP (BMDP) (Auer et al., 2002; Sunberg and Kochenderfer, 2018) constructs a policy tree by executing multiple simulations. Each simulation adds a single belief node to the belief tree or terminates by terminal state or action. To steer toward more deeper and more beneficial simulations, MCTS selects an action $a*$ at each belief node according to the following rule $a* = \underset{a\in\mathcal{A}}{\text{argmax}}\text{UCB}(ha)$, where

$$\text{UCB}(ha) = \widehat{Q}(ha) + c\cdot\sqrt{\log(N(h))/N(ha)}, \tag{16}$$

where $N\,(h)$ is the visitation count of the belief node defined by history h, $N\,(ha)$ is the visitation count of the belief–action node, c is the exploration parameter and, $\widehat{Q}(ha)$ is the estimator of the action-value function Q for node ha obtained by simulations. The rule described by (16) is a result of modeling exploration as a Multi-Armed Bandit (MAB) problem (Auger et al., 2013; Kocsis and Szepesvári, 2006; Munos, 2014). When the action is selected, a question arises either to open a new branch in terms of observation and posterior belief or to continue through one of the existing branches. In continuous action, and observation spaces, this can be resolved by the Double Progressive Widening (DPW) technique (Auger et al., 2013; Sunberg and Kochenderfer, 2018). If a new branch is expanded, an observation z' is created from state x' drawn from the belief b *propagated with an action a*.

Let the return, corresponding to lace i starting from some belief b^i_ℓ at depth $\ell - k$, be $g\left(b^i_\ell, a_\ell, z^i_{\ell+1:k+L}\right)$ for $\ell\in[k : k+L-1]$. More specifically, suppose the new posterior belief was expanded at depth d^i of the belief tree such that $d^i > \ell$. We have that $g\left(b^i_\ell, a_\ell, z^i_{\ell+1:k+L}\right)$ is composed from two parts, the already expanded tree part and the rollout added part such that

$$\underbrace{g\left(b^i_\ell, a_\ell, z^i_{\ell+1:k+L}\right) = \rho\left(b^i_\ell, a_\ell, z^i_{\ell+1}, b^i_{\ell+1}\right) + \sum_{l=\ell+1}^{k+d^i-1}\gamma^{l-\ell}\rho\left(b^i_l, \pi^{*,i}_l\left(b^i_l\right), z^i_{l+1}, b^i_{l+1}\right) +}_{\text{belief tree}} \tag{17}$$

$$\underbrace{+\sum_{l=k+d^i}^{k+L-1}\gamma^{l-\ell}\rho\left(b^i_l, \mu\left(b^i_l\right), z^i_{l+1}, b^i_{l+1}\right)}_{\text{rollout}}, \tag{18}$$

where L is the horizon (tree depth), $\pi^{*,i}$ is an optimal tree policy depending on the number of the simulation i through $\widehat{Q}$ and visitation counts in (16) and μ is the rollout policy. Importantly, in rollout the observations are drawn randomly and since we are in continuous spaces the beliefs in the rollouts are unique. A new belief node is added for $l = k + d^i$. If due to DPW no new belief node was added to the belief tree, no rollout depicted by (18) is commenced and the return sample takes the form of

$$g\left(b^i_\ell, a_\ell, z^i_{\ell+1:k+L}\right) = \rho\left(b^i_\ell, a_\ell, z^i_{\ell+1}, b^i_{\ell+1}\right) + \sum_{l=\ell+1}^{k+L-1}\gamma^{l-\ell}\rho\left(b^i_l, \pi^{*,i}_l\left(b^i_l\right), z^i_{l+1}, b^i_{l+1}\right). \tag{19}$$

The estimate for (9) under optimal future policy is assembled from laces in accordance to

$$\widehat{Q}(h_\ell a_\ell) = \frac{1}{N(h_\ell a_\ell)}\sum_{i=1}^{N(h_\ell a_\ell)} g\left(b^i_\ell, a_\ell, z^i_{\ell+1:k+L}\right), \tag{20}$$

where each reward $\rho(b, a, z', b')$ in the belief tree appears the number of times according to the visitation count of the node b', namely, $N\,(h')$. We note that for both estimators (15) and (20), the formulation in (13) holds.

Now we move to the details of our general approach.

3. Our approach

This section is the core of our general approach. We first describe bounds over the theoretical and the estimated objectives. We then endow the rewards bounds with discrete simplification levels. Finally, instead of calculating rewards, we calculate the bounds over them and if they are not tight enough we tighten them so we can make faster decisions with bounds over the objectives instead of objectives themselves.

3.1. Theoretical simplification formulation

Simplification is any kind of relaxation of POMDP tuple (1) elements, accompanied by guarantees that quantify the (worst-case or potential) impact of a particular simplification technique on planning performance. In this section, we present a general simplification framework that is applicable to any reward bounds that satisfy the assumptions stated in Section 3.3.

Our framework applies without any change to parametric and non-parametric beliefs, and to closed-form belief-dependent rewards (that can be calculated exactly, i.e., analytically), as well as to estimated rewards. Therefore, in this paper, we do not differentiate between these cases and denote the belief-dependent reward by ρ (b_ℓ, a_ℓ, $z_{\ell+1}$, $b_{\ell+1}$), without using the notation $\hat{\square}$ for estimators. In other words, depending on the setting, $\rho()$ and b_ℓ can represent, respectively, an analytical reward and a parametric belief, or a reward estimator and a nonparametric belief. In all cases, if one can provide monotonically adaptive bounds on the reward, our framework will return an identical solution as if the decision making was performed with original reward calculations (i.e., depending on the setting, either an analytical reward calculation or reward estimator calculation). In Section 6, we provide a specific incarnation of the framework considering non-parametric beliefs represented by a set of weighted samples and a reward estimator, and where the simplification corresponds to utilizing only a subset of the samples.

As mentioned, we aim to simplify the belief-dependent reward ρ (b_ℓ, a_ℓ, $z_{\ell+1}$, $b_{\ell+1}$) calculations. Namely, the original reward ρ is bounded using the simplified belief b^s instead of original belief b. This operation materializes in the form of following inequality

$$\begin{aligned}&\underline{\rho}\left(b_\ell^s, b_\ell, a_\ell, z_{\ell+1}, b_{\ell+1}, b_{\ell+1}^s\right)\\ &\leq \rho\left(b_\ell, a_\ell, z_{\ell+1}, b_{\ell+1}\right)\\ &\leq \overline{\rho}\left(b_\ell^s, b_\ell, a_\ell, z_{\ell+1}, b_{\ell+1}, b_{\ell+1}^s\right),\end{aligned} \tag{21}$$

where $\underline{\rho}$ and $\overline{\rho}$ are the corresponding lower and upper bounds, respectively. The superscript s denotes the fact that the corresponding belief was simplified as we depict in Figure 1. Notice that in (21) the pair of consecutive beliefs, b_ℓ and $b_{\ell+1}$, can be simplified differently.

Henceforth, in order to avoid unnecessary clutter we will omit the dependence on the observation and denote the bounds over the reward using simplified beliefs as follows

$$\underline{\rho}^s(b, a, b') \leq \rho(b, a, b') \leq \overline{\rho}^s(b, a, b'). \tag{22}$$

It should be stressed that since in the belief tree b' always has a single parent b, the reader should think about such a reward as one corresponding to b'.

A key requirement is reduced computational complexity of these bounds compared to the complexity of the original reward. Instead of calculating the expensive reward ρ (b, a, b') for each pair of beliefs b, b', we first obtain the corresponding simplified beliefs b^s and b'^s, as illustrated in Figure 1, and then formulate the bounds $\underline{\rho}^s$ and $\overline{\rho}^s$ from (22). However, we note that the form (22) is actually more general and not limited to belief simplification.

Further we formulate bounds over the value function Equation 8 and action-value function Equation 9, both under the optimal policy. In fact, our bounds hold under an arbitrary policy. We narrow the discussion to optimal polices solely for the clarity of the explanation and this is not a limitation of our approach.

Suppose inequality (22) holds for any possible pair of consecutive beliefs, for example, these are analytical bounds, as opposed to (Zhitnikov and Indelman, 2022b). A direct consequence of this fact, alongside the structure of (7), is that

$$\underline{V}\left(b_\ell, \pi_{\ell+}^*\right) \leq V\left(b_\ell, \pi_{\ell+}^*\right) \leq \overline{V}\left(b_\ell, \pi_{\ell+}^*\right), \tag{23}$$

holds for any belief b_ℓ and $\ell \in [k, k + L - 1]$. Using the Bellman representation as in (8) the bounds (23) take the form of

$$\begin{aligned}\overline{V}\left(b_\ell, \pi_{\ell+}^*\right) &= \mathop{\mathbb{E}}_{z_{\ell+1}}\left[\overline{\rho}^s\left(b_\ell, \pi_\ell^*(b_\ell), b_{\ell+1}^i\right) + \overline{V}\left(b_{\ell+1}^i, \pi_{(\ell+1)+}^*\right)\right]\\ \underline{V}\left(b_\ell, \pi_{\ell+}^*\right) &= \mathop{\mathbb{E}}_{z_{\ell+1}}\left[\underline{\rho}^s\left(b_\ell, \pi_\ell^*(b_\ell), b_{\ell+1}^i\right) + \underline{V}\left(b_{\ell+1}^i, \pi_{(\ell+1)+}^*\right)\right].\end{aligned} \tag{24}$$

The bounds over the value function Equation 8 in (24) are initialized at the Lth time step in the planning horizon as $\overline{V}(b_{k+L}, \pi_{k+L}) = 0$ and $\underline{V}(b_{k+L}, \pi_{k+L}) = 0$. Similarly the bounds over the action-value function (9) under an optimal future policy are

$$\underline{Q}\left(b_\ell, \left\{a_\ell, \pi_{(\ell+1)+}^*\right\}\right) \leq Q(b_\ell, a_\ell) \leq \overline{Q}\left(b_\ell, \left\{a_\ell, \pi_{(\ell+1)+}^*\right\}\right), \tag{25}$$

where the policy $\pi_{(\ell+1)+}^*$ is optimal. Note, as we observe in (24), the simplification assumed herein does not affect the distribution of future observations with respect to which the expectation is taken.

3.1.1. Bounding the belief-dependent element of the reward. At this point, we want to recall that commonly, the state-dependent element (2) is much easier to calculate than the belief-dependent one. Leveraging the structure manifested by (11) the immediate bounds over (3) induce bounds over $Q^I(\cdot)$ as such

$$\underline{Q}^I(b_k, a_k) \le Q^I(b_k, a_k) \le \overline{Q}^I(b_k, a_k), \tag{26}$$

and utilizing (11), we arrive at

$$\overline{Q}(b_k, a_k) = (1-\lambda)Q^x(b_k, a_k) + \lambda\overline{Q}^I(b_k, a_k) \tag{27}$$

$$\underline{Q}(b_k, a_k) = (1-\lambda)Q^x(b_k, a_k) + \lambda\underline{Q}^I(b_k, a_k). \tag{28}$$

Importantly, the belief-dependent element (3) does not have to be information-theoretic. The simplification paradigm is general and works for any belief-dependent operator given appropriate bounds.

3.2. Bounds over the estimated objective

As we explained in Section 2.3 in practice the value and action-value function are estimated. Instead of using (23) and (25), we have

$$\underline{\widehat{V}}\left(b_\ell, \pi^*_{\ell+}\right) \le \widehat{V}\left(b_\ell, \pi^*_{\ell+}\right) \le \overline{\widehat{V}}\left(b_\ell, \pi^*_{\ell+}\right), \tag{29}$$

and

$$\underline{\widehat{Q}}\left(b_\ell, \left\{a_\ell, \pi^*_{(\ell+1)+}\right\}\right) \le \widehat{Q}(b_\ell, a_\ell) \le \overline{\widehat{Q}}\left(b_\ell, \left\{a_\ell, \pi^*_{(\ell+1)+}\right\}\right), \tag{30}$$

respectively.

The bounds, in case of symmetric estimators from Section 2.3.1, are

$$\begin{aligned}\overline{\widehat{V}}\left(b_\ell, \pi^*_{\ell+}\right) &= \frac{1}{n_z}\sum_{i=1}^{n_z}\overline{\rho}^s\left(b_\ell, \pi^*(b_\ell), b^i_{\ell+1}\right) \\ &\quad + \gamma\frac{1}{n_z}\sum_{i=1}^{n_z}\overline{\widehat{V}}\left(b^i_{\ell+1}, \pi^*_{(\ell+1)+}\right) \\ \underline{\widehat{V}}\left(b_\ell, \pi^*_{\ell+}\right) &= \frac{1}{n_z}\sum_{i=1}^{n_z}\underline{\rho}^s\left(b_\ell, \pi^*(b_\ell), b^i_{\ell+1}\right) \\ &\quad + \gamma\frac{1}{n_z}\sum_{i=1}^{n_z}\underline{\widehat{V}}\left(b^i_{\ell+1}, \pi^*_{(\ell+1)+}\right),\end{aligned} \tag{31}$$

where to clarify we repeat that n_z is the number of children of b_ℓ under the execution policy $\pi_{\ell+}$ and i is the child index. The bounds over the estimated value function in (31) are initialized at the Lth time step in the planning horizon as $\overline{\widehat{V}}(b_{k+L}, \pi_{k+L}) = 0$ and $\underline{\widehat{V}}(b_{k+L}, \pi_{k+L}) = 0$.

In a similar manner, we define also bounds over (15) as such

$$\begin{aligned}\overline{\widehat{Q}}\left(b_\ell, \left\{a_\ell, \pi^*_{(\ell+1)+}\right\}\right) &= \frac{1}{n_z}\sum_{i=1}^{n_z}\overline{\rho}^s\left(b_\ell, a_\ell, b^i_{\ell+1}\right) \\ &\quad + \gamma\frac{1}{n_z}\sum_{i=1}^{n_z}\overline{\widehat{V}}\left(b^i_{\ell+1}, \pi^*_{(\ell+1)+}\right) \\ \underline{\widehat{Q}}\left(b_\ell, \left\{a_\ell, \pi^*_{(\ell+1)+}\right\}\right) &= \frac{1}{n_z}\sum_{i=1}^{n_z}\underline{\rho}^s\left(b_\ell, a_\ell, b^i_{\ell+1}\right) \\ &\quad + \gamma\frac{1}{n_z}\sum_{i=1}^{n_z}\underline{\widehat{V}}\left(b^i_{\ell+1}, \pi^*_{(\ell+1)+}\right),\end{aligned} \tag{32}$$

We emphasize that the superscript i in (31) and (32) denotes the child posterior nodes of b_ℓ.

The bounds over MCTS estimator (20) are

$$\begin{aligned}\overline{\widehat{Q}}(ha) &= \frac{1}{N(ha)}\sum_{i=1}^{N(ha)}\Bigg(\overline{\rho}^s\left(b^i_\ell, a_\ell, b^i_{\ell+1}\right) \\ &\quad + \sum_{l=\ell+1}^{k+d^i-1}\gamma^{l-\ell}\overline{\rho}^s\left(b^i_l, \pi^{*,i}_l\left(b^i_l\right), b^i_{l+1}\right) \\ &\quad + \sum_{l=k+d^i}^{k+L-1}\gamma^{l-\ell}\overline{\rho}^s\left(b^i_l, \mu\left(b^i_l\right), b^i_{l+1}\right)\Bigg) \\ \underline{\widehat{Q}}(ha) &= \frac{1}{N(ha)}\sum_{i=1}^{N(ha)}\Big(\underline{\rho}^s\left(b^i_\ell, a_\ell, b^i_{\ell+1}\right) \\ &\quad + \sum_{l=\ell+1}^{k+d^i-1}\gamma^{l-\ell}\underline{\rho}^s\left(b^i_l, \pi^{*,i}_l\left(b^i_l\right), b^i_{l+1}\right) \\ &\quad + \sum_{l=k+d^i}^{k+L-1}\gamma^{l-\ell}\underline{\rho}^s\left(b^i_l, \mu\left(b^i_l\right), b^i_{l+1}\right)\Big).\end{aligned} \tag{33}$$

Let us clarify again that in (33) the superscript i denotes the number of the simulation. Moreover, the reward bounds within the tree repeat in more than a single simulation according to the visitation count of the corresponding posterior belief. Clearly, the decomposition displayed by equations (27) and (28) is valid for both bounds (32) and (33). We have that

$$\overline{\widehat{Q}}(b_k, a_k) = (1-\lambda)\widehat{Q}^x(b_k, a_k) + \lambda\overline{\widehat{Q}}^I(b_k, a_k) \tag{34}$$

$$\underline{\widehat{Q}}(b_k, a_k) = (1-\lambda)\widehat{Q}^x(b_k, a_k) + \lambda\underline{\widehat{Q}}^I(b_k, a_k). \tag{35}$$

3.2.1. Impact of the information weight λ. Allow us to linger on the λ from equations (11) and (13). It is hard to predict how the objective will behave with various values of λ. Nevertheless, if the bounds are over the belief-dependent element of the reward, by subtracting (35) from (34), we arrive at

$$\overline{\widehat{Q}}(b_k, a_k) - \underline{\widehat{Q}}(b_k, a_k) = \lambda\left(\overline{\widehat{Q}}^I(b_k, a_k) - \underline{\widehat{Q}}^I(b_k, a_k)\right). \tag{36}$$

The width of the bounds is monotonically increasing with λ. Of course, it will also happen to a theoretical analog of such a bounds displayed by equations (27) and (28). We can envision more speedup from applying the simplification paradigm with lower values of λ and will see it in the simulations.

Further we will consider the estimated action-value or value functions and therefore omit the word “estimated.” We will also omit mentioning each time that our bounds are under the optimal policy.

3.3. Multi-level simplification

We now extend the definition of simplification as we envision it to be an *adaptive paradigm*. We denote *level of simplification* as how “aggressive” the suggested simplification is. Observe an illustration in Figure 2.

With this setting, we can naturally define many discrete levels such that $s \in \{1, 2, \ldots, n_{\max}\}$ represents the simplification level, where 1 and $n_{\max}$ correspond to the coarsest and finest simplification levels, respectively. For instance, suppose the belief is represented by a set of samples (particles), as in Section 6. Taking a small subset of particles to represent the simplified belief corresponds to a *coarse* simplification. If one takes many particles, this corresponds to a *fine* simplification.

Remark. From now on the superscript s denotes the discrete simplification level. Importantly we always have a **finite** number, denoted by $n_{\max}$, of simplification levels.

Further, we assume bounds monotonically become tighter as the simplification level is increased and that the bounds for the finest simplification level $n_{\max}$ converge to the original reward without simplification. More formally, denote $\overline{\Delta}^s(b,a,b') \triangleq \overline{\rho}^s(b,a,b') - \rho(b,a,b')$ and $\underline{\Delta}^s(b,a,b') \triangleq \rho(b,a,b') - \underline{\rho}^s(b,a,b')$.

Assumption 1. Monotonicity. Let $n_{\max} \geq 2$, $\forall s \in [1, n_{\max} - 1]$ we get: $\overline{\Delta}^s(b,a,b') \geq \overline{\Delta}^{s+1}(b,a,b')$ and $\underline{\Delta}^s(b,a,b') \geq \underline{\Delta}^{s+1}(b,a,b')$.

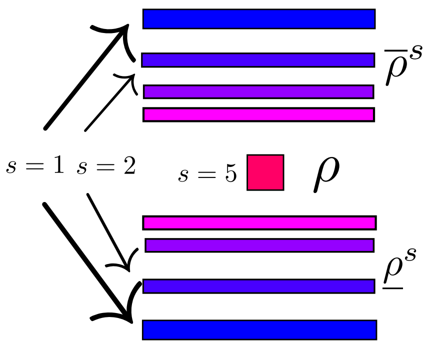


Figure 2. Reward bounds and different levels of the simplification. Here, $n_{\max} = 5$. Warmer colors visualize tighter bounds, whereas colder colors (blue) indicate looser bounds and cheaper to calculate.

Assumption 2. Convergence. $\forall b$, a, b' we get: $\overline{\rho}^{s=n_{\max}}(b,a,b') = \underline{\rho}^{s=n_{\max}}(b,a,b') = \rho(b,a,b')$.

In Section 6, we derive novel bounds on top of a particular simplification that takes a subset of belief samples instead of a complete set. We prove that these bounds indeed satisfy both assumptions.

The simplification levels of the reward bounds for different posterior belief nodes in the belief tree determine how tight the bounds over the value or action-value function are. To tighten the bounds over the objective, we have the freedom to select any rewards the belief tree and tighten the bounds over these selected rewards by increasing their simplification levels; this, in turn, would contract the bounds over the objective.

We call a particular algorithmic scheme to select the rewards a **resimplification strategy**. A general valid resimplificaiton strategy is defined as follows.

Definition 1. Resimplification strategy. Given a pair of lower $\underline{\widehat{V}}(b_\ell, \pi_{\ell+})$ $\left(\underline{\widehat{Q}}(b_\ell, \{a_\ell, \pi^*_{(\ell+1)+}\})\right)$ and upper bounds $\overline{\widehat{V}}(b_\ell, \pi_{\ell+})$ $(\overline{\widehat{Q}}(b_\ell, \{a_\ell, \pi^*_{(\ell+1)+}\}))$ over the estimated objective, the resimplification strategy is a rule to promote one or more simplification levels of the rewards in the the subtree rooted at b_ℓ and defined by the mentioned above estimated objective. If the resimplification does not promote the simplification level for any reward, so $\overline{\widehat{Q}}(b_\ell, \{a_\ell, \pi^*_{(\ell+1)+}\}) - \underline{\widehat{Q}}(b_\ell, \{a_\ell, \pi^*_{(\ell+1)+}\}) = 0$.

Note that, all the rewards within a subtree defined by $\overline{\widehat{Q}}(b_\ell, \{a_\ell, \pi^*_{(\ell+1)+}\}), \underline{\widehat{Q}}(b_\ell, \{a_\ell, \pi^*_{(\ell+1)+}\})$ are being at the maximal simplification level implies $\overline{\widehat{Q}}(b_\ell, \{a_\ell, \pi^*_{(\ell+1)+}\}) - \underline{\widehat{Q}}(b_\ell, \{a_\ell, \pi^*_{(\ell+1)+}\}) = 0$, but the inverse implication is not necessarily true. Once initiated, a **valid** strategy can select no reward for simplification level promotion only if $\overline{\widehat{Q}}(b_\ell, \{a_\ell, \pi^*_{(\ell+1)+}\}) - \underline{\widehat{Q}}(b_\ell, \{a_\ell, \pi^*_{(\ell+1)+}\}) = 0$.

Theorem 1. Monotonicity and Convergence of Estimated Objective Function Bounds. *If the bounds over the reward are monotonic (assumption* 1) *and convergent (assumption* 2), *for both estimators* (32) *and* (33), *the bounds on the sample approximation* (30) *are monotonic as a function of the number of resimplifications and convergent after at most* $n_{\max} \cdot M$ *resimplifications for* ***any*** *resimplification strategy. Here M is the number of posterior beliefs in* (32) *or* (33). *Namely, if all the rewards are at the maximal simplifciation level* $n_{\max}$ *we have to reach*

$$\underline{\widehat{Q}}(\cdot) = \widehat{Q}(\cdot) = \overline{\widehat{Q}}(\cdot). \tag{37}$$

Similarly for optimal value function the equality $\underline{\widehat{V}}(\cdot) = \widehat{V}(\cdot) = \overline{\widehat{V}}(\cdot)$ *holds.*

The reader can find the proof in the Appendix 11.1. Theorem 1 ensures that if the resimplification strategy is valid (Definition 1), we do not get stuck in an infinite loop of

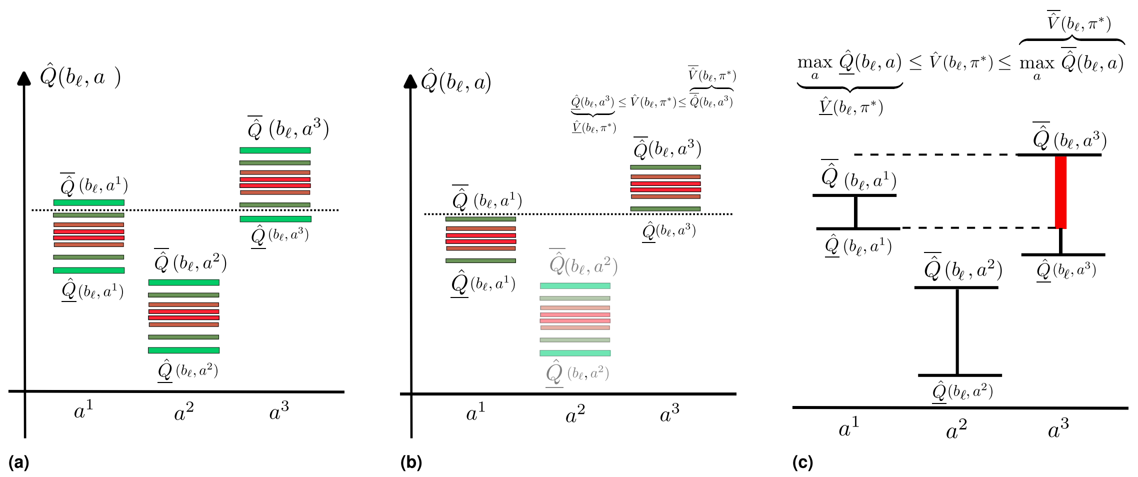

$\hat{Q}(b_\ell, a\ )$
$\overline{\widehat{Q}}\ (b_\ell, a^1)$
$\widehat{\underline{Q}}\ (b_\ell, a^1)$
$\overline{\widehat{Q}}\ (b_\ell, a^2)$
$\widehat{\underline{Q}}\ (b_\ell, a^2)$
$\overline{\widehat{Q}}\ (b_\ell, a^3)$
$\widehat{\underline{Q}}\ (b_\ell, a^3)$
a^1
a^2
a^3
(a)
$\hat{Q}(b_\ell, a)$
$\overbrace{\widehat{\underline{Q}}(b_\ell,a^3)}^{}_{\underline{V}(b_\ell,\pi^*)} \leq \hat{V}(b_\ell,\pi^*) \leq \overbrace{\overline{\widehat{Q}}(b_\ell,a^3)}^{\overline{V}(b_\ell,\pi^*)}$
(b)
$\underbrace{\max_a \widehat{\underline{Q}}(b_\ell,a)}_{\underline{V}(b_\ell,\pi^*)} \leq \hat{V}(b_\ell,\pi^*) \leq \overbrace{\max_a \overline{\widehat{Q}}(b_\ell,a)}^{\overline{V}(b_\ell,\pi^*)}$
(c)

Figure 3. In this illustration we have three candidate actions $\{a^1, a^2, a^3\}$ that can possibly be taken by the robot from the belief node b_ℓ. (a) We observe that $\overline{\widehat{Q}}(b_\ell, a^1_\ell) > \widehat{\underline{Q}}(b_\ell, a^3_\ell)$ and prune action a^2. (b) After the resimplification no overlap an we can safely decide that a^3_ℓ is optimal. Moreover we prune the withered interval corresponding to the a^2_ℓ. (c) Another situation where we are not concerned about optimal action, we solely want to send up to the tree the bounds over optimal value function.

resimplifications if instead of $\widehat{Q}(\cdot)$ we use its bounds. In particular, if (37) is reached, there is no reason to activate the resimplification routine.

Importantly, as we discuss next and corroborate by simulations in many cases we can identify the optimal action before reaching the maximal number of resimplificaitons.

3.4. *Adaptive simplification mechanics*

Our adaptive simplification approach is based on two key observations. The *first key observation* is that we can compare bounds over (30) constituted by rewards at different levels of simplification. Our *second key observation* is that we can reuse calculations between different simplification levels avoiding recalculation of the simplification from scratch.

Naturally we do not want to reach (37). Let us begin by explaining how we determine an optimal action by using bounds over the action-value function instead of its explicit calculation and obtain a significant speedup in planning time. If there is no overlap between the intervals originated from the upper and lower bounds (30) of each candidate action, we can determine the optimal action and therefore there is no reason to call the resimplification routine.

Contemplate about some belief b_ℓ in the belief tree. We annotate by superscript j candidate actions emanating from b_ℓ, such that the index j corresponds to the jth candidate action. We first select a candidate action using the lower bound (30) over $\widehat{Q}(b_\ell, a^j_\ell)$ as

$$j^\dagger(b_\ell(h_\ell)) = \arg\max_j \Big\{ \widehat{\underline{Q}}\left(b_\ell(h_\ell), \{a^j_\ell, \pi^*_{(\ell+1)+}\}\right) + c^j(h_\ell a^j) \Big\}, \tag{38}$$

where c^j is an action-dependent constant. In case of a given belief tree $c^j = 0\ \forall j$, whereas in case of MCTS, it is a constant originated from UCB as in (16).

We then ask the question whether or not an overlap with another candidate action exists,

$$\widehat{\underline{Q}}\ \left(b_\ell, \{a^{j^\dagger}_\ell, \pi^*_{(\ell+1)+}\}\right) + c^{j^\dagger} \overbrace{\geq}^{?}$$
$$\geq \max_{j \in \{1\ldots\} \setminus \{j^\dagger\}} \Big\{ \overline{\widehat{Q}}\left(b_\ell, \{a^j_\ell, \pi^*_{(\ell+1)+}\}\right) + c^j \Big\} \tag{39}$$

See a visualization in Figure 3(a).

If the condition displayed by equation (39) is not fulfilled, as depicted in Figure 3(a), we shall tighten the bounds (30) by calling a **resimplification strategy**. Importantly, in case of a given belief tree, even if an overlap is present similar to branch-and-bound technique (Kochenderfer et al., 2022) we can prune any subtree obtained with action j satisfying

$$\widehat{\underline{Q}}\left(b_\ell, \{a^{j^\dagger}_\ell, \pi^*_{(\ell+1)+}\}\right) + c^{j^\dagger} \geq \overline{\widehat{Q}}\left(b_\ell, \{a_\ell{}^j, \pi^*_{(\ell+1)+}\}\right) + c^j. \tag{40}$$

We illustrated this aspect in Figure 3(b). If the belief tree is constructed gradually as in MCTS based methods and anytime setting, instead of pruning, we still can use (40) to dismiss suboptimal, at current simulation of MCTS, actions (See Figure 4).

Once no overlap is present (the condition (39) is fulfilled) we can declare that the selected action is optimal $(\pi^*_\ell(b_\ell) = a^{j^\dagger(b_\ell)}_\ell)$. Utilizing the optimal action we can bound the *optimal* value function $\widehat{V}(b_\ell, \pi^*_{\ell+})$ as such

$$\overline{\widehat{V}}\left(b_\ell, \{\pi^*_\ell, \pi^*_{(\ell+1)+}\}\right) \triangleq \overline{\widehat{Q}}\left(b_\ell, \{a^{j^\dagger(b_\ell)}_\ell, \pi^*_{(\ell+1)+}\}\right), \tag{41}$$

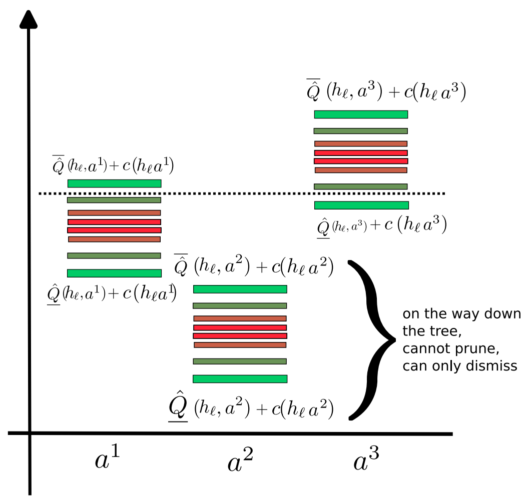

$\overline{Q}(h_\ell,a^3)+c(h_\ell a^3)$
$\overline{Q}(h_\ell,a^1)+c(h_\ell a^1)$
$\hat{\underline{Q}}(h_\ell,a^3)+c(h_\ell a^3)$
$\overline{Q}(h_\ell,a^2)+c(h_\ell a^2)$
$\hat{\underline{Q}}(h_\ell,a^1)+c(h_\ell a^1)$
on the way down the tree, cannot prune, can only dismiss
$\hat{\underline{Q}}(h_\ell,a^2)+c(h_\ell a^2)$
a^1
a^2
a^3

Figure 4. Demonstration of our approach in the setting of MCTS. In contrast to Figure 3(b), we cannot prune action a^2 and can only dismiss it to not participate in resimplifications. This is because, in the next tree queries, a^2 may be the best action for the robot to take.

$$\underline{\widehat{V}}(b_\ell,\{\pi^*_\ell,\pi^*_{(\ell+1)+}\}) \triangleq \underline{\widehat{Q}}(b_\ell,\{a_\ell^{j^\dagger(b_\ell)},\pi^*_{(\ell+1)+}\}). \qquad (42)$$

Let us recite that the bounds (41) and (42) are conditioned on the fact that there is no overlap of the bounds intervals that correspond to different candidate actions, namely the condition (39) is met for *each* belief b_ℓ in the belief tree. This situation is visualized in Figure 3(b).

On the other hand, to identify the optimal immediate action a_k^*, we require no overlap between bounds of different actions only at the root of the belief tree (where the belief is b_k). This means that at each belief node b_ℓ in the tree, besides the root, we only want to bound the value function for the optimal action (and under optimal future policy). While it is possible to do so by first determining the optimal action, as in (41) and (42), we can bypass this step and directly bound the value function over the optimal action as follows,

$$\overline{\widehat{V}}(b_\ell,\{\pi^*_\ell,\pi^*_{(\ell+1)+}\}) \triangleq \max_j \overline{\widehat{Q}}(b_\ell,\{a^j,\pi^*_{(\ell+1)+}\}), \qquad (43)$$

$$\underline{\widehat{V}}(b_\ell,\{\pi^*_\ell,\pi^*_{(\ell+1)+}\}) \triangleq \max_j \underline{\widehat{Q}}(b_\ell,\{a^j,\pi^*_{(\ell+1)+}\}), \qquad (44)$$

that is, relaxing the requirement of no overlap between bounds for different actions at any node b_ℓ besides b_k. See illustration of (43) and (44) in Figure 3(c). In turn, eliminating a single overlap at the root results in lower rewards simplification levels in the tree, although such a value bounds may be looser. As we shall see, this approach would typically yield more speedup.

Nevertheless, when we need a policy tree we still have to obtain an optimal action at each belief node within the tree. This requires no bounds overlap at each node, as in the former setting. This situation arises for example when the action and observation spaces are large but discrete. In this case the robot sometimes does not do re-planning at each time step. Instead the robot uses the policy tree as a representation of the policy and selects an optimal action that corresponds to the received observation. In addition, such a strategy accommodates possible reuse calculations in such a solved belief tree (Farhi and Indelman, 2019, 2021).

To conclude this section, let us summarize. As discussed, we have the following two variants:

- The resimplification is initiated at each nonterminal posterior belief node b_ℓ up until no overlap between candidate actions is present and the optimal action $\pi^*_\ell(b_\ell)$ is selected. This way we bound the optimal value function of the descendant to b_k nodes using an optimal action according to (41) and (42). We named this approach Policy Tree (PT).
- The resimplification is commenced solely at the root b_k of the whole belief tree. We eliminate the overlap and obtain an optimal action only at b_k. This way we use (43) and (44) to bound the optimal value function of the descendant to b_k nodes. We shall refer to this variant of our approach as LAZY.

3.5. Specific resimplification strategies

In this paper we consider two specific resimplification strategies that are elaborated in the next sections: Simplification Level (SL) and Gap. We note that additional valid resimplification strategies exist and can be plugged-in into the above-proposed general theory.

3.5.1. Simplification level. The resimplification strategy can be directly tied to the simplification level. In this situation the resimplifcation strategy promotes simplification level of the rewards inside the belief tree corresponding to bounds in (29) or (30) based on the simplification level itself. We provide further details in the setting of a given belief tree, considering a PT variant in Section 4.

3.5.2. Gap. Another possibility is that the resimplification is tied to the gap $\overline{\rho}^s - \underline{\rho}^s$. Such a resimplification promotes the simplification level if the reward bounds gap satisfies a certain condition. We describe thoroughly this resimplification flavor in the setting of a given belief tree, considering LAZY variant in Section 4.2, and in MCTS setting, considering a PT variant in Section 5.4.

Each of these strategies can be used in conjunction with any of the variants PT and LAZY. In the sequel, we shall denote these combinations explicitly, for example, PT-SL, LAZY-Gap and PT-Gap.

The preceding discussion raises the question of how do we actually incorporate the proposed bounds into online decision making. This brings us to the next section. We first consider a given belief tree and then coupled belief tree construction and solution as in MCTS methods. It shall be noted that further presented resimplification strategies are also suitable for static candidate action sequences, with minor modifications.

4. Adaptive simplification in the setting of a given belief tree

We start with the assumption that the belief tree was generated in some way and that it is given, for example, Sparse Sampling (SS) algorithm introduced by Kearns et al. (2002). In other words the belief tree construction is not coupled with rewards calculation and estimation of the objective.

In this setting, we contribute two resimplification strategies. The first strategy is described in Section 4.1. The general idea is to break down recursively a given belief tree $\mathbb{T}$ into its sub-problems (subtrees), denoted as $\{\mathbb{T}^j\}_{j=1}^{|\mathcal{A}|}$ (each subtree j at the root belief has a single action j), and solve each sub-problem with its own simplification level of the corresponding belief subtree. Ultimately this would lead to the solution of the entire problem via action-value function bounds (32). This strategy is based on Simplification Level and it is a PT strategy. The action-value bounds should not overlap **at each node** in the given belief tree.

The second strategy is described in Section 4.2. This resimplification strategy is based on Gap and it is a LAZY strategy. Here, the general idea is to first substitute all the rewards in a given belief tree by bounds with the coarsest simplification level. We then eliminate an overlap between candidate actions only at the root belief node b_k by a repetitive descending to the belief tree, promoting the simplification levels along a single lace chosen according to largest gap and ascending back. We emphasize that in this setting, the action-value bounds should not overlap **only at the root node** in the given belief tree.

As mentioned in the beginning of Section 2.3.1, only for simplicity we consider a symmetric setting in terms of sampled actions and the observations, but the approach is applicable without any limitations to any given belief tree.

4.1. Resimplification strategy: PT-SL

This section presents our first resimplification strategy. We now turn to thorough description.

Not to be confused with **policy tree** represented by the (14) or (15) the **given** belief tree ($\mathbb{T}$) has more than a single action emanating from each belief node besides the leaves.

We now assign a simplification level to the bounds on the value and action-value functions. Consider again some belief node b_ℓ in the belief tree, and assume recursively for *each* of its children belief nodes $b_{\ell+1}$

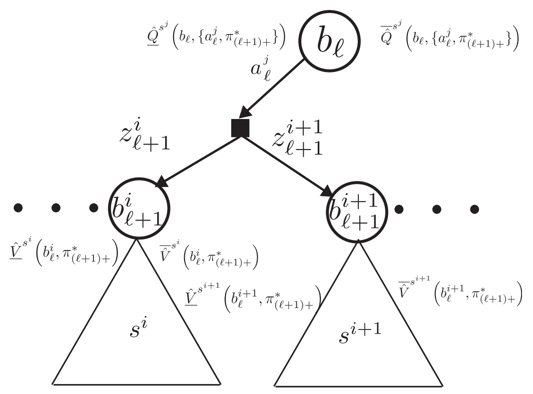

$\underline{\hat{Q}}^{s^j}\left(b_\ell,\{a^j_\ell,\pi^*_{(\ell+1)+}\}\right)$
b_ℓ
$\overline{\hat{Q}}^{s^j}\left(b_\ell,\{a^j_\ell,\pi^*_{(\ell+1)+}\}\right)$
a^j_ℓ
$z^i_{\ell+1}$
$z^{i+1}_{\ell+1}$
$b^i_{\ell+1}$
$b^{i+1}_{\ell+1}$
$\underline{\hat{V}}^{s^i}\left(b^i_\ell,\pi^*_{(\ell+1)+}\right)$
$\overline{\hat{V}}^{s^i}\left(b^i_\ell,\pi^*_{(\ell+1)+}\right)$
$\underline{\hat{V}}^{s^{i+1}}\left(b^{i+1}_\ell,\pi^*_{(\ell+1)+}\right)$
$\overline{\hat{V}}^{s^{i+1}}\left(b^{i+1}_\ell,\pi^*_{(\ell+1)+}\right)$
s^i
s^{i+1}

Figure 5. Pruning the subtrees by adaptively promoting the simplification levels of the rewards inside. Here, the simplification levels of a subtrees are not equal. It is possible that $s^i \neq s^{i+1}$. Note that here the superscripts are relative to b_ℓ as opposed to Figures 1 and 6.

we already calculated the optimal policy $\pi^*_{(\ell+1)+}(b_{\ell+1})$ and the corresponding upper and lower bounds $\underline{\hat{V}}^s(b_{\ell+1},\pi^*_{(\ell+1)+})$ and $\overline{\hat{V}}^s(b_{\ell+1},\pi^*_{(\ell+1)+})$. In general, these bounds for each child sub-policy tree of b_ℓ can correspond to different simplification levels.

From now on let the superscript s over the action-value function bounds from (32) and (31) denote the simplification level stemmed from pertaining reward bounds. The bounds previously described by equation (32) for belief node b_ℓ, incorporating simplification level, are now modified to

$$\overline{\hat{Q}}^{s^j}\left(b_\ell,\{a^j_\ell,\pi^*_{(\ell+1)+}\}\right)=\frac{1}{n_z}\sum_{i=1}^{n_z}\overline{\rho}^s\left(b_\ell,a^j_\ell,b^i_{\ell+1}\right)$$
$$+\gamma\frac{1}{n_z}\sum_{i=1}^{n_z}\overline{\hat{V}}^{s^i}\left(b^i_{\ell+1},\pi^*_{(\ell+1)+}\right)$$
$$\underline{\hat{Q}}^{s^j}\left(b_\ell,\{a^j_\ell,\pi^*_{(\ell+1)+}\}\right)=\frac{1}{n_z}\sum_{i=1}^{n_z}\underline{\rho}^s\left(b_\ell,a^j_\ell,b^i_{\ell+1}\right)$$
$$+\gamma\frac{1}{n_z}\sum_{i=1}^{n_z}\underline{\hat{V}}^{s^i}\left(b^i_{\ell+1},\pi^*_{(\ell+1)+}\right), \tag{45}$$

as illustrated in Figure 5. We shall pinpoint the abuse of notation here. In contrast to (32) the superscript s over the immediate reward bounds denotes a specific simplification level instead of indicating a general simplification.

Note equation (45) applies for each $a^j_\ell \in \mathcal{A}$, and as mentioned, each belief node $b^i_{\ell+1}$ (one for each observation $z^i_{\ell+1}$) has, in general, its own simplification level s^i. In other words, for each $b^i_{\ell+1}$, s^i is the simplification level that was sufficient for calculating the bounds $\left\{\overline{\hat{V}}^{s^i}(b^i_{\ell+1},\pi^*_{(\ell+1)+}),\ \underline{\hat{V}}^{s^i}(b^i_{\ell+1},\pi^*_{(\ell+1)+})\right\}$ and the corresponding optimal policy $\pi^*_{(\ell+1)+}$. Thus, when addressing belief node b_ℓ in (45), for each belief node $b^i_{\ell+1}$ and its corresponding simplification level s^i, these bounds are already available.

Further, as seen in (45), the immediate reward and the corresponding bounds $\overline{\rho}$ and $\underline{\rho}$, in general, can be calculated with their own simplification level s. In particular, when starting calculations, s could correspond to a default coarse simplification level, for example, coarsest level $s = 1$. Another possibility is to set $s = s^i$ for corresponding simplification level of value function bounds of the ith child belief.

To define simplification level s^j of the bounds (45), we leverage the recursive nature of the Bellman update and define

$$s^j \triangleq \min\{\underbrace{s}_{\substack{\overline{\rho}^s \\ \underline{\rho}^s}}, \underbrace{s^{i=1}, s^{i=2} \ldots s^{i=n_z}}_{\substack{\overline{\widehat{V}}^{s^i}\left(b^i_{\ell+1}, \pi^*_{(\ell+1)+}\right) \\ \underline{\widehat{V}}^{s^i}\left(b^i_{\ell+1}, \pi^*_{(\ell+1)+}\right)}}\}, \tag{46}$$

where $\{s^{i=1}, s^{i=2}, \ldots, s^{i=n_z}\}$ represent the (generally different) simplification levels of optimal value functions of belief nodes $b^i_{\ell+1}$ considered in the expectation approximation in (45).

We now wish to decide which action $a_\ell^{j^\dagger(b_\ell)} \in \mathcal{A}$ is optimal from belief node b_ℓ; the corresponding optimal policy would then be $\pi^*_{\ell+} = \{a^*_\ell, \pi^*_{(\ell+1)+}\}$, where $\pi^*_{(\ell+1)+}$ is the already-calculated optimal policy for belief nodes $\{b^i_{\ell+1}\}_{i=1}^{n_z}$ that a^*_ℓ leads to. See illustration in Figure 5.

Let us utilize now a general simplification approach described in Section 3.4. Overall in each belief node we have n_a candidate actions indexed by superscript j in (45).

At each belief node we first select an optimal action candidate according to (38) with a nullified action-dependent constant ($\forall j\ c^j = 0$). Further, in any PT resimplification strategy there are three possible scenarios.

- No overlap is present ((39) is satisfied) and we are at the root, that is, $b_\ell = b_k$. In this case the optimal action shall be returned.
- No overlap is present ((39) is satisfied) and we not at the root b_k. In this case, using the optimal action we bound optimal value function using the (41) and (42).
- Equation (39) is not satisfied, meaning an overlap is present. In the presence of overlap we shall prune actions according to (40) and commence resimplification routine based on resimplification strategy.

We now discuss how the simplification level is updated recursively from the simplification level of pertaining reward bounds, and revisit the process to calculate the optimal policy and the corresponding bounds. For some belief node b_ℓ in the belief tree, consider the bounds $\overline{\widehat{Q}}^{s^j}(b_\ell, \{a_\ell^j, \pi^*_{(\ell+1)+}\})$ and $\underline{\widehat{Q}}^{s^j}(b_\ell, \{a^j_\ell, \pi^*_{(\ell+1)+}\})$ from (45) for different actions $a^j_\ell \in \mathcal{A}$, that partially overlap and therefore could not be pruned. Each policy tree corresponding to action a^j_ℓ can generally have its own simplification level s^j. We now iteratively increase the simplification level by 1. This can be done for each of the branches, if s^j is identical for all branches, or only for the branch with the coarsest simplification level.

Consider now any such branch whose simplification level needs to be adapted from s^j to $s^j + 1$. Recall, that at this point, the mentioned bounds were already calculated, thus their ingredients, in terms of $\left\{\overline{\rho}^s(b_\ell, a^j_\ell, b^i_{\ell+1}), \underline{\rho}^s(b_\ell, a^j_\ell, b^i_{\ell+1})\right\}_{i=1}^{n_z}$ and $\left\{\overline{\widehat{V}}^{s^i}(b^i_{\ell+1}, \pi^*_{(\ell+1)+}), \underline{\widehat{V}}^{s^i}(b^i_{\ell+1}, \pi^*_{(\ell+1)+})\right\}_{i=1}^{n_z}$, involved in approximating the expectation in (45), are available. Recall also (46), that is, each element in $\{s, s^{i=1}, s^{i=2}, \ldots, s^{i=n_z}\}$ is either equal or larger than s^j. We now discuss both cases, starting from the latter.

As we assumed bounds to improve monotonically as simplification level increases, see Assump. 1, for any $s^i > s^j + 1$ we already have readily available bounds $\overline{\widehat{V}}^{s^i}(b^i_{\ell+1}, \pi^*_{(\ell+1)+}), \underline{\widehat{V}}^{s^i}(b^i_{\ell+1}, \pi^*_{(\ell+1)+})$ which are tighter than those that would be obtained for simplification level $s^j + 1$. Thus, we can *safely skip* the calculation of the latter and use the existing bounds from level s^i as is.

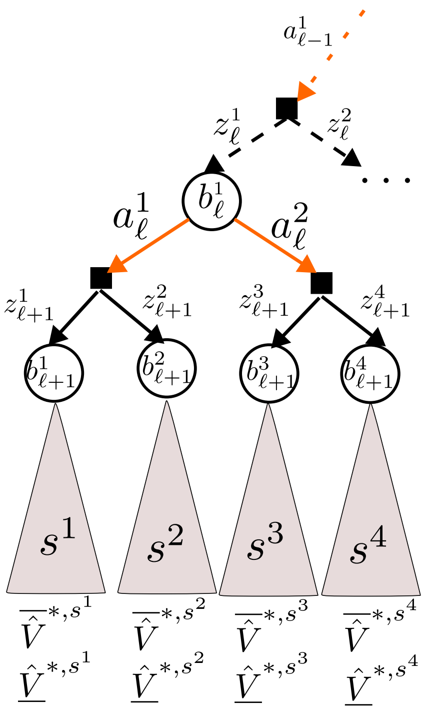
For the former case, that is, $s^i = s^j$, we now have to adapt the simplification level of a child tree i to $s^j + 1$ by calculating the bounds $\overline{\widehat{V}}^{s^i+1}(b^i_{\ell+1}, \pi^*_{(\ell+1)+}), \underline{\widehat{V}}^{s^i+1}(b^i_{\ell+1}, \pi^*_{(\ell+1)+})$. Here, our *key insight* is that, instead of calculating these bounds from scratch, we can re-use calculations between different simplification levels, in this case, from level s^i. As the bounds from that level are available, we can identify only the incremental part that is "missing" to get from simplification level s^i to $s^i + 1$, and update the existing bounds $\overline{\widehat{V}}^{s^i}(b^i_{\ell+1}, \pi^*_{(\ell+1)+}), \underline{\widehat{V}}^{s^i}(b^i_{\ell+1}, \pi^*_{(\ell+1)+})$ to recover $\overline{\widehat{V}}^{s^i+1}(b^i_{\ell+1}, \pi^*_{(\ell+1)+}), \underline{\widehat{V}}^{s^i+1}(b^i_{\ell+1}, \pi^*_{(\ell+1)+})$ exactly. The same argument applies also for bounds over momentary rewards. In Section 6.2.3, we apply this approach to a specific simplification and reward function.

We can repeat iteratively the above process of increasing the simplification level until we can prune all branches but one. This means each subtree will be solved maximum once, per simplification level. Since we assumed the reward bounds converge monotonically to the original reward for the finest level $s = n_{\max}$ (See Figure 2), from Theorem 1, we are guaranteed to eventually disqualify all sub-optimal branches. Our described approach is summarized in Algs. 1 and 2.

4.1.1. Illustrative example.

We now illustrate the described above resimplification strategy in a toy example. Before we start this section, let us clarify that in the example the superscripts are global over the belief tree in contrast to the previous section. Consider Figure 6 and assume the subtrees to b_ℓ^1 were solved using simplification levels that hold $s^2 = s^1 + 1$, $s^2 < s^3, s^4$. Further assume the immediate reward simplification is $s = s^1$. According to definitions above this means that for subtree starting at b_ℓ^1 and action a_ℓ^1 the simplification level is $\min\{s^1, s^2\}$ and for action a_ℓ^2 the simplification level is $\min\{s^3, s^4\}$. Now, we consider the case the existing bounds of the subtrees were not tight enough to prune, we adapt simplification level starting from b_ℓ^1 and promote $s \leftarrow s^1 + 1$. Since $s^1 < s^1 + 1$ we re-simplify the subtree corresponding to simplification level of s^1 to simplification level $s^1 + 1$, that is, to a finer simplification.

However we do not need to re-simplify subtrees corresponding to s^2, s^3, s^4: The tree corresponding to s^2 is already simplified to the currently desired level; thus we can use its existing bounds. For the two other trees, their current simplification levels, s^3 and s^4, are higher (finer) than the desired $s^1 + 1$ level, and since the bounds are tighter as simplification level increases we can use their existing tighter bounds without the need to "go-back" to a coarser level of simplification. If we can now prune one of the actions, we keep pruning up the tree. If pruning is still not possible, we need to adapt simplification again with simplification level $s^1 + 2$.



$a_{\ell-1}^1$
z_ℓ^1
z_ℓ^2
b_ℓ^1
a_ℓ^1
a_ℓ^2
$z_{\ell+1}^1$
$z_{\ell+1}^2$
$z_{\ell+1}^3$
$z_{\ell+1}^4$
$b_{\ell+1}^1$
$b_{\ell+1}^2$
$b_{\ell+1}^3$
$b_{\ell+1}^4$
s^1
s^2
s^3
s^4
$\overline{\hat{V}}^{*,s^1}$ $\overline{\hat{V}}^{*,s^2}$ $\overline{\hat{V}}^{*,s^3}$ $\overline{\hat{V}}^{*,s^4}$
$\underline{\hat{V}}^{*,s^1}$ $\underline{\hat{V}}^{*,s^2}$ $\underline{\hat{V}}^{*,s^3}$ $\underline{\hat{V}}^{*,s^4}$

Figure 6. An example of the simplification paradigm. The superscript here denotes **global** number of the belief, observation or action in the belief tree as opposed to equation (45) and Figure 5.

Algorithm 1 Simplified Information Theoretic Belief Space Planning (SITH-BSP)

```
1:  procedure SolveBeliefTree(belief-tree: $\mathbb{T}$)
2:      if $\mathbb{T}$ is a leaf then
3:          //Corresponds to a single belief node.
4:          return 0, 0
5:      end if
6:      for all subtrees $\mathbb{T}^j \in \{\mathbb{T}^j\}_{j=1}^{|\mathcal{A}|}$ do        // Actions
7:          //Observations
8:          for all subtrees $\mathbb{T}^{j,i} \in \{\mathbb{T}^{j,i}\}_{i=1}^{n_z}$ do
9:              //Returns Optimal Value bounds and prune suboptimal branches of $\mathbb{T}^{j,i}$.
10:             SolveBeliefTree($\mathbb{T}^{j,i}$)
11:             Set the simplification level of $\underline{\rho}^s(b, a^j, b'^i)$ and $\overline{\rho}^s(b, a^j, b'^i)$ as in (46)
12:         end for
13:         Calculate $\underline{\hat{Q}}^{s^j}, \overline{\hat{Q}}^{s^j}$ according to (45)
14:     end for
15:     Prune($\{\underline{\hat{Q}}^{s^j}, \overline{\hat{Q}}^{s^j}\}_{j=1}^{|\mathcal{A}|}$)        // Alg. 2
16:     while not all subtrees $\mathbb{T}^j \in \{\mathbb{T}^j\}_{j=1}^{|\mathcal{A}|}$ but 1 pruned do
17:         Find minimal simplification level $s_{\min}$ between all $\underline{\hat{Q}}^{s^j}, \overline{\hat{Q}}^{s^j}$ corresponding to not pruned $\mathbb{T}^j$
18:         // Can be more than single subtree
19:         select subtree $s^j == s_{\min}$
20:         ResimplifyTree($\mathbb{T}^j$)
21:         Prune($\{\underline{\hat{Q}}^{s^j}, \overline{\hat{Q}}^{s^j}\}_{j=1}^{|\mathcal{A}|}$)        // Alg. 2
22:     end while
23:     return optimal action branch that left $a^*$ and $\underline{\hat{Q}}^{s^j}, \overline{\hat{Q}}^{s^j}$.
24: end procedure
25: procedure ResimplifyTree($\mathbb{T}^j$)
26:     for all subtrees $\mathbb{T}^{j,i} \in \{\mathbb{T}^{j,i}\}_{i=1}^{n_z}$ do
27:         ResimplifyReward($\mathbb{T}^j$, $b$, $a^j$, $b^i$)        // Alg. 3
28:         if $b^i$ has children then
29:             // $s^i$ is a simplification level of corresponding optimal value function (policy tree)
30:             if $s^i \leq s_{\min}$ then
31:                 // Alg. 4
32:                 ResimplifySubtree($\mathbb{T}^{j,i}$, $b$, $b^i$)
33:                 $s^i \leftarrow s^i + 1$
34:             end if
35:         end if
36:     end for
37:     $s^j \leftarrow s^j + 1$
38: end procedure
```

Algorithm 2 Pruning of trees

```
1: procedure Prune
2:     Input: (belief-tree root, $b$; bounds of root's children, $\{\underline{\hat{Q}}^j, \overline{\hat{Q}}^j\}_{j=1}^{n_a}$)        // $n_a$ is the number of child branches (candidate actions) going out of $b$.
3:     $\underline{\hat{Q}}^* \leftarrow \max_j \{\underline{\hat{Q}}^j\}_{j=1}^{n_a}$
4:     for $j \in 1 : n_a$ do
5:         if $\underline{\hat{Q}}^* > \overline{\hat{Q}}^j$ then
6:             prune child $j$ from the belief tree
7:         end if
8:     end for
9: end procedure
```

Algorithm 3 ResimplifyReward

```
1: procedure ResimplifyRewad($\mathbb{T}^j$, $b$, $a$, $b'$)
2:     Obtain corresponding to the $\mathbb{T}^j$ bounds $\overline{\hat{V}}$, $\underline{\hat{V}}$
3:     $\overline{\hat{V}} \leftarrow \overline{\hat{V}} - \frac{\overline{\rho}^s(bab')}{n_z}$
4:     $\underline{\hat{V}} \leftarrow \underline{\hat{V}} - \frac{\underline{\rho}^s(bab')}{n_z}$
5:     Advance level of simplification of $b'$
6:     $\overline{\hat{V}} \leftarrow \overline{\hat{V}} + \frac{\overline{\rho}^s(bab')}{n_z}$
7:     $\underline{\hat{V}} \leftarrow \underline{\hat{V}} + \frac{\underline{\rho}^s(bab')}{n_z}$
8: end procedure
```

Algorithm 4 ResimplifySubtree

```
1: procedure ResimplifySubtree($\mathbb{T}^{j,i}$,$b$ $b'$)
2:     $\overline{\hat{V}}(b) \leftarrow \overline{\hat{V}}(b) - \gamma\frac{\overline{\hat{V}}(b')}{n_z}$
3:     $\underline{\hat{V}}(b) \leftarrow \underline{\hat{V}}(b) - \gamma\frac{\underline{\hat{V}}(b')}{n_z}$
4:     ResimplifyTree($\mathbb{T}^{j,i}$)
5:     $\overline{\hat{V}} \leftarrow \overline{\hat{V}}(b) + \gamma\frac{\overline{\hat{V}}(b')}{n_z}$
6:     $\underline{\hat{V}} \leftarrow \underline{\hat{V}}(b) + \gamma\frac{\underline{\hat{V}}(b')}{n_z}$
7: end procedure
```

Algorithm 5 Lazy Simplified Information Theoretic Belief Space Planning (LAZY-BSP)

```
1: procedure Plan(belief: $b$, belief-tree: $\mathbb{T}$)
2:     BoundOptimalValue(belief: $b$, belief-tree: $\mathbb{T}$)
3:     $a^* \leftarrow$ ActionSelection($b$, $L$)                // Alg. 6
4:     return $a^*$
5: end procedure
6: procedure BoundOptimalValue(belief-tree: $\mathbb{T}$)
7:     if $\mathbb{T}$ is a leaf then
8:         //Corresponds to a single belief node.
9:         return $0, 0$
10:    end if
11:    for all subtrees $\mathbb{T}^j \in \{\mathbb{T}^j\}_{j=1}^{|\mathcal{A}|}$ do
12:        for all subtrees $\mathbb{T}^{j,i} \in \{\mathbb{T}^{j,i}\}_{i=1}^{n_z}$ do
13:            $\underline{\hat{V}}(b'), \overline{\hat{V}}(b') \leftarrow$ BoundOptimalValue($b$, $\mathbb{T}^{j,i}$)
14:            Set the simplification level of $\underline{\rho}^s(b, a^j, b'^i)$ and $\overline{\rho}^s(b, a^j, b'^i)$ to coarsest possible
15:        end for
16:        Calculate $\underline{\hat{Q}}^j, \overline{\hat{Q}}^j$
17:    end for
18:    $\underline{\hat{V}}(b) \leftarrow \max_j\{\underline{\hat{Q}}^j\}$
19:    $\overline{\hat{V}}(b) \leftarrow \max_j\{\overline{\hat{Q}}^j\}$
20:    return $\underline{\hat{V}}(b), \overline{\hat{V}}(b)$
21: end procedure
```

Algorithm 6 Action Selection for Lazy Simplified Information Theoretic Belief Space Planning

```
1: procedure ActionSelection(belief: $b$, horizon: $L$)
2:     Prune($\{\underline{\hat{Q}}^j, \overline{\hat{Q}}^j\}_{j=1}^{|\mathcal{A}|}$)                // Alg. 2
3:     $a^\dagger \leftarrow \arg\max_a \underline{\hat{Q}}\Big(b, \{a, \pi^*_{(k+1+)}\}\Big)$
4:     $\tilde{a} \leftarrow \arg\max_{a \in \mathcal{A} \setminus a^\dagger} \overline{\hat{Q}}\Big(b, \{a, \pi^*_{(k+1+)}\}\Big)$
5:     $\Delta \leftarrow \Big(\overline{\hat{Q}}(b, \{\tilde{a}, \pi^*_{(k+1+)}\}) - \underline{\hat{Q}}(b, \{a^\dagger, \pi^*_{(k+1+)}\})\Big)^+$
6:     while $\Delta > 0$ do
7:         $a^* \leftarrow$ LazyResimplify($b$, $L$)
8:     end while
9:     return $a^*$
10: end procedure
11: procedure LazyResimplify(belief: $b(h)$, depth: $d$)
12:     if $b$ is leaf then
13:         return $0, 0$
14:     end if
15:     $\tilde{a} \leftarrow \arg\max_{a \in C(h)} \overline{\hat{Q}}(b, \{a, \pi^*_{(k+1+)}\} - \underline{\hat{Q}}(b, \{a, \pi^*_{(k+1+)}\})$
                                                        // Gap as in (47)
16:     if $d == 1$ then
17:         $b' \leftarrow \arg\max_{b' \in C(h\tilde{a})} \overline{\rho}^s(b\tilde{a}b') - \underline{\rho}^s(b\tilde{a}b')$
18:     else
19:         $b' \leftarrow \arg\max_{b' \in C(h\tilde{a})} \overline{\hat{V}}(b') - \underline{\hat{V}}(b')$
20:     end if
21:     ResimplifyReward($\mathbb{T}$, $b$, $\tilde{a}$, $b'$)                // Alg. 3
22:     $\overline{\hat{Q}}(b, \{\tilde{a}, \pi^*_{(k+1+)}\} \leftarrow \overline{\hat{Q}}(b, \{\tilde{a}, \pi^*_{(k+1+)}\} - \gamma\overline{\hat{V}}(b')$
23:     $\underline{\hat{Q}}(b, \{\tilde{a}, \pi^*_{(k+1+)}\} \leftarrow \underline{\hat{Q}}(b, \{\tilde{a}, \pi^*_{(k+1+)}\} - \gamma\underline{\hat{V}}(b')$
24:     $\underline{\hat{V}}(b'), \overline{\hat{V}}(b') \leftarrow$ LazyResimplify($b'$, $d-1$)
25:     $\overline{\hat{Q}}(b, \{\tilde{a}, \pi^*_{(k+1+)}\} \leftarrow \overline{\hat{Q}}(b, \{\tilde{a}, \pi^*_{(k+1+)}\} + \gamma\overline{\hat{V}}(b')$
26:     $\underline{\hat{Q}}(b, \{\tilde{a}, \pi^*_{(k+1+)}\} \leftarrow \underline{\hat{Q}}(b, \{\tilde{a}, \pi^*_{(k+1+)}\} + \gamma\underline{\hat{V}}(b')$
27:     $\underline{\hat{V}}(b) \leftarrow \max_a\{\underline{\hat{Q}}(b, \{a, \pi^*_{(k+1+)}\}\}$
28:     $\overline{\hat{V}}(b) \leftarrow \max_a\{\overline{\hat{Q}}(b, \{a, \pi^*_{(k+1+)}\}\}$
29:     return $\underline{\hat{V}}(b), \overline{\hat{V}}(b)$
30: end procedure
```

4.1.2. A detailed algorithm description. Let us thoroughly describe Alg. 1. We are given a belief tree $\mathbb{T}$. First at the line 10, Alg. 1 recursively descend to the leaves. When the line 11 is hit for the first time the corresponding rewards are set to the initial simplification level or also possible that minimal level of child optimal value bounds is used. In our simulations, we used minimal reward level. Further the algorithm calculates bounds over action-value function represented by (45). This happens in line 15 of Alg. 1. The next step is to try to prune all subtrees but one utilizing Alg. 1. Note, at this point all the subtrees $\mathbb{T}^j$ are already policy trees, namely, only a single action emanating from each posterior belief. In there is more that single action left after pruning, at the line 20, Alg. 1 calls routine ResimplifyTree to initiate **resimplification** for selected subtree corresponding to action a^j. The simplification level of a single step ahead reward is always have to be promoted as we do in line 27. Further, Alg. 1 treats similarly subtrees, if they are present.

4.2. Resimplification strategy: LAZY-Gap

The PT resimplification strategy from the previous section assures that no overlap is present (Figure 3(b)) at each non-leaf posterior belief and we know the optimal action to take. However, it can inflict a redundant computational burden. We can handle the overlap only at the root of the belief tree and use the bounds over optimal value function according to (43) and (44). Since we already presented the resimplification strategy based on the simplification levels, our second resimplification strategy will be based on the distance between reward bounds. However, the bounds (43) and (44) can be utilized directly also with the resimplifcation strategy based on simplification levels. Yet, this is out of the scope of this paper.

In this section, we present a lazy variant of the resimplifcation strategy. In a LAZY variant, the overlap is checked solely at the root b_k of the whole belief tree. In this approach three scenarios can be encountered at each belief node.

- The belief node is not root. We bound optimal value according to (43) and (44).
- At the root b_k we shall check for overlap. If no overlap is present ((39) is satisfied) we prune all suboptimal actions according to Alg. 2 and return an optimal action as described in Section 3.4.
- In the presence of an overlap at the root b_k (equation (39) is not satisfied), we shall prune actions according to (40) and Alg. 2 and commence a resimplification routine for the nonpruned actions based on the resimplification strategy.

Having presented general steps of any LAZY variant of resimplification strategy, we are ready to delve into specific gap driven resimplfication strategy. Let us introduce the following notation

$$G(ha) \triangleq \overline{\widehat{Q}}(ha) - \underline{\widehat{Q}}(ha). \tag{47}$$

We remind the reader that sometimes, for simplicity of explanation, we will make the gap dependent on belief and an action, and denote $G(ba)$. We use this gap to steer the resimplification procedure toward more promising lace. The lace with actions inducing largest gap (47) at each belief action node along the lace will be selected to resimplification. In fact, we use similar gap for value function to select observations along the lace. Now let us proceed to the detailed algorithm description.

4.2.1. A detailed algorithm description. This approach is summarized in Alg. 5 When we apply this resimplification strategy, we first use the lowest simplification level for each pair of consecutive beliefs in the given belief tree. In other words, Alg. 5 first descends to the leaves of the given belief tree. Then it bounds each optimal value function using the initial simplification level using (43) and (44). This initial passage over the given belief tree is enclosed by routine BoundOptimalValue. In the procedure ActionSelection, we increase the simplification level of the reward bounds in the given tree until there is no overlap at the **root**, as in Figure 3(b). In this way, we can prune entire given subtrees at the root, corresponding to candidate actions. The procedure LazyResimplify descends back to some leaf through the lace with largest gaps on the way. It selects action in line 15. It then select observation/belief according to largest gap of a single step ahead rewards if these rewards are leafs (line 17) or the largest gap of the optimal value function bounds (line 19).

5. Adaptive simplification in the setting of MCTS

In the previous sections, we described the application of the adaptive simplification paradigm when the belief tree is given or its construction is not coupled with the solution. We now turn to an anytime setting where the belief tree is not given. Instead, the belief tree construction is coupled with the estimation of the action-value function (20) at each belief action node. Such an approach is commonly used in Monte Carlo tree search

(MCTS) methods based on an exploration strategy, for example, Upper Confidence Bound (UCB) as in (16). Our goal is to suggest a resimplifcation strategy so that exactly the same belief tree as without simplification would be constructed. Also the same optimal action is identified with and without simplification. To support general belief-dependent rewards, we select PFT-DPW as the baseline, as mentioned in Section 1.1.

Algorithm 7 SITH-PFT

```
1:  procedure PLAN(belief: $b$)
2:      for $i \in 1 : n$ or timeout do
3:          $h \leftarrow \emptyset$
4:          SIMULATE($b$, $d_{\max}$, $h$)
5:      end for
6:      return ACTIONSELECTION($b$, $h$)    // called with nullified exploration constant $c$
7:  end procedure
8:  procedure SIMULATE(belief: $b$, depth: $d$, history: $h$)
9:      if $d = 0$ then
10:         return 0
11:     end if
12:     $a \leftarrow$ ACTIONSELECTION($b$, $h$)
13:     if $|C(ha)| \leq k_o N(ha)^{\alpha_o}$ then
14:         $o \leftarrow$ sample $x$ from $b$, generate $o$ from $(x, a)$
15:         $b' \leftarrow G_{\mathrm{PF}(m)}(bao)$
16:         Calculate initial $\overline{\rho}^s, \underline{\rho}^s$ for $b, b'$ based on $s \leftarrow 1$ // minimal simp. level
17:         $C(ha) \leftarrow C(ha) \cup \{(\overline{\rho}^s, \underline{\rho}^s, b', o)\}$
18:         $L, U \leftarrow \overline{\rho}^s, \underline{\rho}^s + \gamma$ ROLLOUT($b'$, $hao$, $d-1$)
19:     else
20:         $(\overline{\rho}^s, \underline{\rho}^s, b', o) \leftarrow$ sample uniformly from $C(ha)$
21:         $L, U \leftarrow \overline{\rho}^s, \underline{\rho}^s + \gamma$ SIMULATE($b'$, $hao$, $d-1$)
22:     end if
23:     if deepest resimplification depth $< d$ then    // accounting for updated deeper in the tree bounds. See section 5.3
24:         reconstruct $\underline{\hat{Q}}(ha), \overline{\hat{Q}}(ha)$
25:     end if
26:     $N(h) \leftarrow N(h) + 1$
27:     $N(ha) \leftarrow N(ha) + 1$
28:     $\overline{\hat{Q}}(ha) \leftarrow \overline{\hat{Q}}(ha) + \frac{U - \overline{\hat{Q}}(ha)}{N(ha)}$
29:     $\underline{\hat{Q}}(ha) \leftarrow \underline{\hat{Q}}(ha) + \frac{L - \underline{\hat{Q}}(ha)}{N(ha)}$
30:     return $L, U$
31: end procedure
```

Common exploration strategies conform to the structure presented in (38). Without losing generality, we focus on the most advanced, to our knowledge, exploration strategy, named UCB and portrayed by (16).

5.1. UCB bounds

With this perspicuity in mind, we now introduce bounds over (16)

$$\overline{\mathrm{UCB}}(ha) \triangleq \overline{\hat{Q}}(ha) + c \cdot \sqrt{\log(N(h))/N(ha)}, \tag{48}$$

$$\underline{\mathrm{UCB}}(ha) \triangleq \underline{\hat{Q}}(ha) + c \cdot \sqrt{\log(N(h))/N(ha)}. \tag{49}$$

Similar to the given belief tree setting we now proceed to the explanation how the reward bounds (22) yield (48) and (49).

5.2. Guaranteed belief tree consistency

Algorithm 8 Action Selection for SITH-PFT

```
1:  procedure ACTIONSELECTION($b$, $h$)
2:      if $|C(h)| \leq k_a N(h)^{\alpha_a}$ then    // action Prog. Widening
3:          $a \leftarrow$ NEXTACTION($h$)
4:          $C(h) \leftarrow C(h) \cup \{a\}$
5:      end if
6:      while true do
7:          Status, $a \leftarrow$ SELECTBEST($b$, $h$)
8:          if Status then
9:              break
10:         else
11:             for all $b', o \in C(ha)$ do
12:                 RESIMPLIFY($b'$, $hao$)
13:             end for
14:             reconstruct $\overline{\hat{Q}}(ha), \underline{\hat{Q}}(ha)$
15:         end if
16:     end while
17:     return $a$
18: end procedure
19: procedure SELECTBEST($b$, $h$)
20:     Status $\leftarrow$ true
21:     $\tilde{a} \leftarrow \arg\max_a \{\underline{\mathrm{UCB}}(ha)\}$
22:     gap $\leftarrow 0$
23:     child-to-resimplify $\leftarrow \tilde{a}$
24:     for all $ha$ children of $b$ do
25:         if $\underline{\mathrm{UCB}}(h\tilde{a}) < \overline{\mathrm{UCB}}(ha) \wedge a \neq \tilde{a}$ then
26:             Status $\leftarrow$ false
27:             if $\overline{\hat{Q}}(ha) - \underline{\hat{Q}}(ha) >$ gap then
28:                 gap $\leftarrow \overline{\hat{Q}}(ha) - \underline{\hat{Q}}(ha)$
29:                 child-to-resimplify $\leftarrow a$
30:             end if
31:         end if
32:     end for
33:     return Status, child-to-resimplify
34: end procedure
```

Since the simplification paradigm substituted UCB (16) by the bounds (48) and (49), the belief tree construction is coupled with these quantities, as opposed to the situation with the given belief tree. If there is an overlap between bounds on UCB for different actions, we can no longer guarantee the same belief tree will be constructed with and without simplification.

In this and the following sections, we address this key issue. Specifically, we define the notion of Tree Consistency and prove the equivalence of our algorithm to our baseline PFT-DPW.

Definition 2. Tree consistent algorithms. Imagine two algorithms, constructing a belief tree. Assume every common sampling operation for the two algorithms uses the same seed. The two algorithms are *tree consistent* if two belief trees constructed by the algorithms are identical in terms of actions, observations, and visitation counts.

Our approach relies on a specific procedure for selecting actions within the tree. Since in each simulation the MCTS descends down the tree with a single return lace as in (20), on the way down it requires the action maximizing UCB (16) we shall eliminate overlap at each belief node as described in Section 3.4. Further we restate the action selection procedure described in Section 3.4 with particular action-dependent constant from equations (38) and (39) rendering the UCB bounds from (48) and (49).

Our action selection is encapsulated by Alg. 8, which is different from the procedure used in PFT-DPW. On top of DPW as in Sunberg and Kochenderfer (2018) with parameters k_a and α_a, instead of selecting an action maximizing the UCB (16), at every belief node, we mark as a candidate action the one that maximizes the lower bound $\underline{\text{UCB}}$ as such

$$\tilde{a} = \arg\max_{a \in C(h)} \underline{\text{UCB}}\,(ha). \tag{50}$$

If $\forall a \neq \tilde{a}$, $\underline{\text{UCB}}\,(h\tilde{a}) \geq \overline{\text{UCB}}(ha)$, there is no overlap (Figure 7(c)) and we can declare that $\tilde{a}$ is identical to a^*, that is, the action that would be returned by PFT using (16) and the tree consistency has not been affected. Otherwise, the bounds must be tightened, so ensure the tree consistency. We examine the ha siblings of $h\tilde{a}$, which satisfy $a \neq \tilde{a} : \underline{\text{UCB}}\,(h\tilde{a}) < \overline{\text{UCB}}(ha)$ (Figure 7(a)). Our next step is to tighten the bounds by resimplification (Figure 7(b)) until there is no overlap using the valid resimplification strategy according to Definition 1.

Remark. Note that here we cannot use the "lazy variant" from Section 4.2 due to the fact that the MCTS requires selecting an action going down to the tree, see line 12 of Alg. 7. Therefore, if the UCB bounds do still overlap, we cannot assure that the same act on will be selected as in case of UCB itself.

5.3. A detailed algorithm description

Now we introduce our efficient variant of the Particle Filter Tree (PFT) presented in Sunberg and Kochenderfer (2018). We call our approach Simplified Information-Theoretic Particle Filter Tree (SITH-PFT). SITH-PFT (Alg. 7) incorporates the adaptive simplification into PFT-DPW. We adhere to the conventional notations as in Sunberg and Kochenderfer (2018) and denote by $G_{\text{PF}(m)}(bao)$ a generative model receiving as input the belief b, an action a and an observation o *(For clarity we substituted z' by o.)*, and producing the posterior belief b'. For belief update, we use a particle filter based on n_x state samples. A remarkable property of our efficient variant is the consistency of the belief tree. In other words, PFT and SITH-PFT have the same belief tree constructed with (16), while SITH-PFT enjoys substantial acceleration. By $C(ha)$, we denote the set of the children (posterior beliefs corresponding to the myopic observations) of the belief action node uniquely

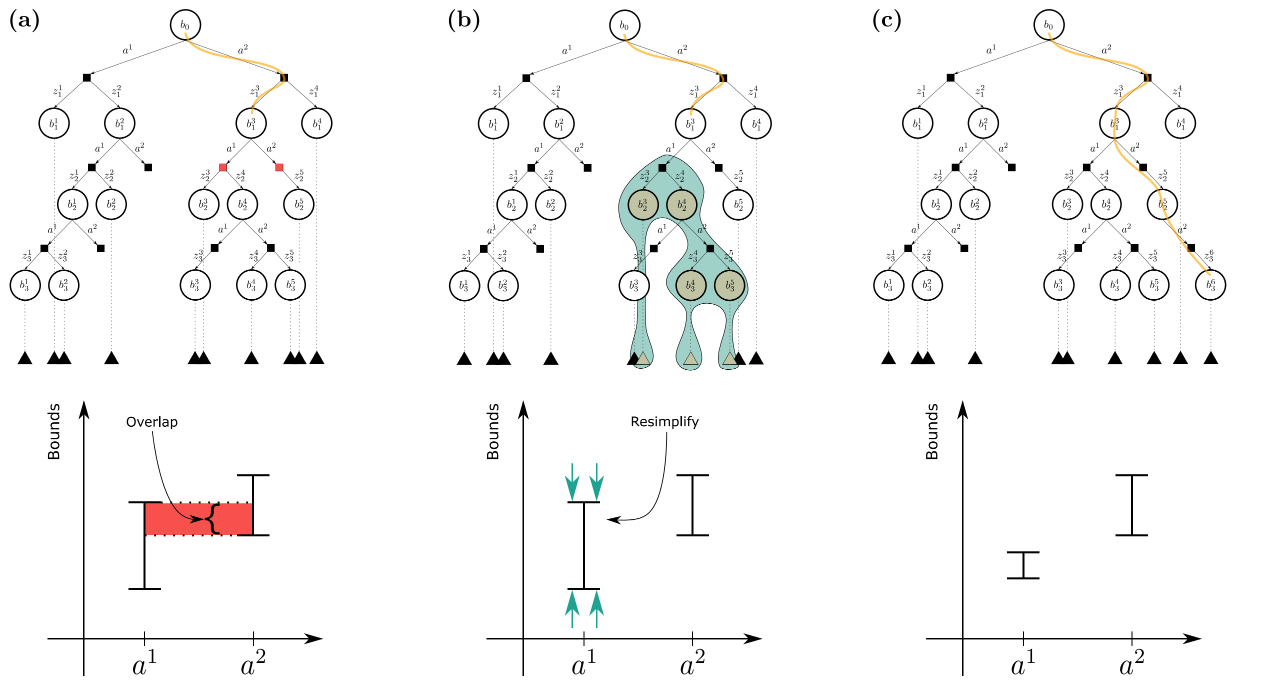


Figure 7. Illustration of our approach. The circles denote the belief nodes, and the rectangles represent the belief-action nodes. Rollouts, emanating from each belief node, are indicated by dashed lines finalized with triangles. (a) The simulation starts from the root of the tree, but at node b_1^3 it can not continue due to an overlap of the child nodes (colored red) bounds. (b) One of the red colored belief-action nodes is chosen, and resimplification is triggered from it down the tree to the leaves (shaded green area in the tree). The beliefs and rollouts inside the green area (colored by light brown) undergo resimplification if decided so. This procedure results in tighter bounds. (c) After the bounds got tighter, nothing prevents the SITH-PFT from continuing down from node b_1^3 guaranteeing the tree consistency. If needed, additional resimplifications can be commenced.

indexed by the history h with concatenated action a. Line 13 in Alg. 7 is the DPW technique from Sunberg and Kochenderfer (2018) with parameters k_o and α_o. The $N(\cdot)$ is the visitation count of belief or belief action nodes. In MCTS, the Q estimate is assembled by averaging the laces of the returns over simulations see equation (20). Each simulation yields a sum of discounted cumulative rewards. Therefore, by replacing the reward with adaptive lightweights bounds (22), we get corresponding discounted cumulative upper and lower bounds over the returns. Averaging the simulations (Alg. 7 lines 28–29), yields the bounds over the action-value function and the UCB bounds used in the routine ActionSelection () to be explained in the next paragraph.

Consider a belief-action node ha at level d with $\overline{\hat{Q}}(ha)$, $\underline{\hat{Q}}(ha)$. Suppose the algorithm selects it for bounds narrowing, as described in Section 5.2 and Alg. 8 line 7. All tree nodes of which ha is an ancestor, contribute their immediate $\overline{\rho}^s, \underline{\rho}^s$ bounds to $\overline{\hat{Q}}(ha)$, $\underline{\hat{Q}}(ha)$ computation. Thus, to tighten $\overline{\hat{Q}}(ha), \underline{\hat{Q}}(ha)$, we can potentially choose any candidate node(s) in the subtree of ha. Each child belief node of ha is sent to the resimplification routine (Alg. 8 lines $11-13$), which performs the following tasks. First, it selects the action (Alg. 9 line 7) that will participate in the subsequent resimplification call and sends all its children beliefs nodes to the recursive call further down the tree (Alg. 9 line 8–10). Second, it refines the belief node $\overline{\rho}, \underline{\rho}$ according to the specific *resimplification strategy* (Alg. 9 lines 3, 4, 12, and18). Third, it reconstructs $\overline{\hat{Q}}(ha), \underline{\hat{Q}}(ha)$ once all the child belief nodes of ha have returned from the resimplification routine (Alg. 9 line 11) as we thoroughly explain in the next section. Fourthly, it engages the rollout resimplification routine according to the specific *resimplification strategy* (Alg. 9 lines 4 and 13). Upon completion of this resimplification call initiated at ha, we obtain tighter immediate bounds of some of ha descendant belief nodes (including rollouts nodes). Accordingly, appropriate descendant of ha belief-action nodes bounds $\left(\overline{\hat{Q}}, \underline{\hat{Q}}\right)$ shall be updated.

Many resimplification strategies are possible, below we present our approach. In Section 4.2 we presented a resimplicifation strategy based on gap. Now we adapt it to the MCTS setting.

5.4. Specific resimplification strategy: PT-gap

In this section, we explain the resimplification procedure in more detail. In particular we present a specific resimplification strategy and further show that this strategy is valid according to Definition 1. When some sibling belief action nodes have overlapping bounds (Figure 3(a) and 7), we strive to avoid tightening them all at once since fewer resimplifications lead to greater acceleration (speedup). Thus, we choose a single ha-node that causes the largest "gap," denoted by G, between its bounds (see Alg. 8 lines 24–30), where G is defined by (47).

Algorithm 9 Resimplification

```
1:  procedure ResimplifY(b, h)
2:      if b is a leaf then
3:          RefineBounds(b)
4:          ResimplifyRollout(b, h)
5:          return
6:      end if
7:      $\tilde{a} \leftarrow \arg\max_a \{N(ha) \cdot (\overline{\hat{Q}}(ha) - \underline{\hat{Q}}(ha))\}$
8:      for all $b', o \in C(h\tilde{a})$ do
9:          Resimplify($b'$, $h\tilde{a}o$)
10:     end for
11:     reconstruct $\overline{\hat{Q}}(h\tilde{a}), \underline{\hat{Q}}(h\tilde{a})$
12:     RefineBounds(b)
13:     ResimplifyRollout(b, h)
14:     return
15: end procedure
16: procedure ResimplifyRollout(b, h)
17:     $b^{\text{rollout}}$ ← find weakest link in rollout
18:     RefineBounds($b^{\text{rollout}}$)
19: end procedure
20: procedure RefineBounds(b)
21:     if (51) holds for b, refine its $\overline{\rho}^{s+1}, \underline{\rho}^{s+1}$ and promote
        its simplification level
22: end procedure
```

Further, we tighten the bounds down the branch of the chosen node (see Alg. 8 lines 11–13) for each member of $C(ha)$, the set of children of ha. Since the bounds converge to the actual reward (Assumption 2) we can guarantee that Alg. 8 will pick a single action after a finite number of resimplifications; thus, tree consistency is assured.

Specifically, we decide to refine $\overline{\rho}^s, \underline{\rho}^s$ of a belief node indexed by h' at depth d' within the subtree starting from a belief action node indexed by ha at depth d when

$$\gamma^{d-d'} \cdot (\overline{\rho}^s - \underline{\rho}^s) \geq \frac{1}{d} G(ha), \tag{51}$$

where $G(ha)$ corresponds to the gap (47) of the belief-action node ha that initially triggered resimplification in Alg. 8 line 24.

The explanation of resimplification strategy based on (51) is rather simple. The right hand side of (51) is the mean gap per depth/level in the sub-tree with ha as its root and spreading downwards to the leaves. Naturally, some of the nodes in this subtree have $\overline{\rho}^s - \underline{\rho}^s$ above or equal to the mean gap and some below. We want to locate and refine all those above or equal to it. For the left side of (51), the rewards are accumulated and discounted according to their depth. Thus, we must account for the relative discount factor. Note that the depth identified with the root is the horizon $d_{\max} = L$, as seen in Alg. 7 line 4, and the leaves are distinguished by depth $d = 0$. For each rollout originating from a tree belief node, we find the rollout node with the largest $\overline{\rho} - \underline{\rho}$ satisfying (51) term locally in the rollout and resimplify it (Alg. 9 lines 4 and 13). To choose the action to continue resimplification down the tree, we take the action corresponding to the belief action node with the largest gap, weighted by its visitation count (Alg. 9 line 7). With this strategy, we aim to keep the belief tree at the lowest possible simplification level while maintaining belief tree consistency.

If the action selection procedure triggers resimplification, it modifies the bounds through the tree. Since the resimplification works recursively, it reconstructs the belief-action node bounds coming back from the recursion (Alg. 9 line 11). Similarly, the action dismissal procedure reconstructs $\overline{\widehat{Q}}$ and $\underline{\widehat{Q}}$ of the belief-action node at which the action dismissal is performed (Alg. 8 line 14). Moreover, on the way back from the simulation, we shall update the ancestral belief-action nodes of the tree. Specifically, we need to reconstruct each $\overline{\widehat{Q}}$ and $\underline{\widehat{Q}}$ that is higher than the deepest starting point of the resimplification (Alg. 7 line 23–25). The reconstruction is essentially a double loop. To reconstruct $\overline{\widehat{Q}}(ha), \underline{\widehat{Q}}(ha)$ we first query for all belief children nodes hao. We then query all belief-action nodes that are children to the hao, that is, $haoa'$. The possibly modified immediate bounds $\underline{\rho}$ and $\overline{\rho}$ are taken from hao nodes and the $\overline{\widehat{Q}}(\cdot)$, $\underline{\widehat{Q}}(\cdot)$ bounds are taken from the $haoa'$ nodes. Importantly, each of the bounds is weighted according to the proper visitation count.

5.5. Guarantees

In this section, we first show that the resimplification strategy suggested in the previous section is valid.

Lemma 1. Validity of the suggested resimplification strategy. *The resimplification strategy presented in Section 5.4 promotes the simplification level of at least one reward in the rollout or belief tree. Alternatively, all the rewards are at the maximal simplification level $n_{\max}$. In other words, the suggested resimplifcation strategy is valid.*

We provide the complete proof in Appendix 11.2. Having proved the validity of the suggested resimplification strategy, we proceed to the monotonicity and convergence of UCB bounds from (48) and (49).

Lemma 2. Monotonicity and convergence of UCB bounds. *The UCB bounds are monotonic as a function of the number of resimplifications and after at most $n_{\max} \cdot M$ resimplifications we have that*

$$\overline{\mathrm{UCB}}(ha) = \underline{\mathrm{UCB}}(ha) = \mathrm{UCB}(ha). \tag{52}$$

We provide the proof in Appendix 11.3. Now, using Lemma 2, we prove that SITH-PFT (Alg. 7) yields the same belief tree and the same best action as PFT.

Theorem 2. *SITH-PFT and PFT are Tree Consistent Algorithms for **any** valid resimplification strategy.*

Theorem 3. *SITH-PFT provides the same solution as PFT for **any** valid resimplification strategy.*

We provide the full proofs of Theorems 2 and 3 in Appendix 11.4 and 11.5, respectively. We showed that for any valid resimplification strategy SITH-PFT is guaranteed to construct the same belief tree as PFT and select the same best action at the root. From Lemma 1, our resimplification strategy is valid. Thus, we achieved the desired result.

6. Specific simplification and information-theoretic bounds

In this section, we focus on a specific simplification in the context of a continuous state space and nonparametric beliefs represented by n_x weighted particles,

$$b \triangleq \{w^i, x^i\}_{i=1}^{n_x}. \tag{53}$$

Suggested Simplification: Given the belief representation (53), the simplified belief is a subset of n_x^s particles, sampled from the original belief, where $n_x^s \leq n_x$. More formally:

$$b_k^s \triangleq \left\{ \left(x_k^i, w_k^i\right) \middle| i \in A_k^s \subseteq \{1, 2, \dots, n_x\}, \left|A_k^s\right| = n_x^s \right\}, \tag{54}$$

where A_k^s is the set of particle indices comprising the simplified belief b_k^s for time k.

Increasing the level of simplification is done *incrementally*. Specifically, when resimplification is carried out, new indices are drawn from the sets $\{1, 2, \dots, n_x\} \backslash A_k^s$ and and included to the set A_k^s. This operation promotes the simplification level to $s + 1$ and defines A_k^{s+1}.

6.1. Novel bounds over differential entropy estimator

As one of our key contributions, we now derive novel analytical bounds for the differential entropy estimator from Boers et al. (2010). These bounds can then be used within our general simplification framework presented in the previous sections. To calculate differential entropy

$$\mathcal{H}(b(x_k)) \triangleq -\int b(x_k) \cdot \log(b(x_k)) \mathrm{d}x_k,$$

One must have access to the manifold representing the belief. In a nonparametric setting this manifold is out of reach. We have to resort to approximations. Several approaches exist. One of them is using Kernel Density Estimation (KDE) as done, for example, by Fischer and Tas (2020). Here, however, we consider the method proposed by Boers et al. (2010). This method builds on top of usage of motion and observation models such that

$$\widehat{\mathcal{H}}(b_k, a_k, z_{k+1}, b_{k+1}) \triangleq \log\left[\sum_{i=1}^{n_x} \mathbb{P}_O\left(z_{k+1} \middle| x_{k+1}^i\right) w_k^i\right] - \sum_{i=1}^{n_x} w_{k+1}^i \cdot \log\left[\mathbb{P}_O\left(z_{k+1} \middle| x_{k+1}^i\right) \sum_{j=1}^{n_x} \mathbb{P}_T\left(x_{k+1}^i \middle| x_k^j, a_k\right) w_k^j\right]. \tag{55}$$

One can observe this method requires access to samples representing both b_k and b_{k+1}; thus, this corresponds to an information-theoretic reward of the form $\rho^I(b_k, a_k, z_{k+1}, b_{k+1})$. Note that as explained in Section 3 such a reward is tied to b_{k+1}.

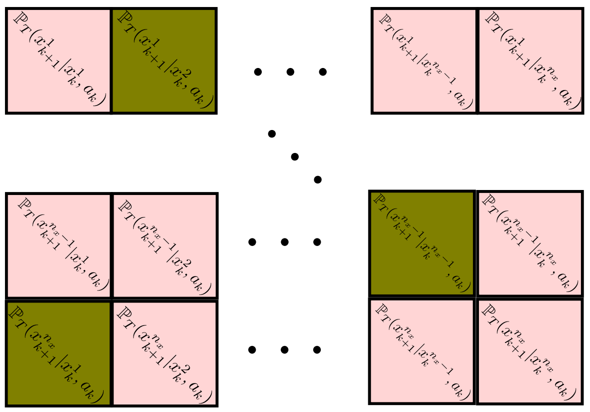

$\mathbb{P}_T(x^1_{k+1}|x^1_k, a_k)$
$\mathbb{P}_T(x^1_{k+1}|x^2_k, a_k)$
$\mathbb{P}_T(x^1_{k+1}|x^{n_x-1}_k, a_k)$
$\mathbb{P}_T(x^1_{k+1}|x^{n_x}_k, a_k)$
$\mathbb{P}_T(x^{n_x-1}_{k+1}|x^1_k, a_k)$
$\mathbb{P}_T(x^{n_x-1}_{k+1}|x^2_k, a_k)$
$\mathbb{P}_T(x^{n_x-1}_{k+1}|x^{n_x-1}_k, a_k)$
$\mathbb{P}_T(x^{n_x-1}_{k+1}|x^{n_x}_k, a_k)$
$\mathbb{P}_T(x^{n_x}_{k+1}|x^1_k, a_k)$
$\mathbb{P}_T(x^{n_x}_{k+1}|x^2_k, a_k)$
$\mathbb{P}_T(x^{n_x}_{k+1}|x^{n_x-1}_k, a_k)$
$\mathbb{P}_T(x^{n_x}_{k+1}|x^{n_x}_k, a_k)$

Figure 8. Schematic visualization of calculations reuse principle in bounds. We select **columns** using indexes from set A^s_k and rows by A^s_{k+1}. We marked by **olive** color resulting constituents of the bounds.

For the sake of clarity and to remove unnecessary clutter we apply an identical simplification described by (54) to both beliefs b_k and b_{k+1}. The simplification indices for both beliefs are defined by A^s_{k+1}. However this is not an inherent limitation. One can easily maintain two sets of indices so as the theory presented below is developed to this more general setting. Moreover, as mentioned in Section 3, we have the same belief b_{k+1} also participating in $\rho^I(b_{k+1}, a_{k+1}, z_{k+2}, b_{k+2})$. In this reward, the simplification indices for b_{k+1} will according to A^s_{k+2} (and not according to A^s_{k+1}).

Utilizing the chosen simplification (54), we now introduce the following upper and lower bounds on (55).

Theorem 4. Adaptive bounds on differential entropy estimator. *The estimator* (55) *can be bounded by*

$$\begin{aligned}&\ell\big(b_k, a_k, z_{k+1}, b_{k+1}; A^s_k, A^s_{k+1}\big)\\ &\leq -\widehat{\mathcal{H}}(b_k, a_k, z_{k+1}, b_{k+1})\\ &\leq u\big(b_k, a_k, z_{k+1}, b_{k+1}; A^s_k, A^s_{k+1}\big),\end{aligned} \tag{56}$$

where

$$u \triangleq -\log\left[\sum_{i=1}^{n_x}\mathbb{P}_O\big(z_{k+1}|x^i_{k+1}\big)w^i_k\right] \tag{57}$$

$$+\sum_{i\notin A^s_{k+1}} w^i_{k+1}\cdot\log\big[m\cdot\mathbb{P}_O\big(z_{k+1}|x^i_{k+1}\big)\big]$$

$$+\sum_{i\in A^s_{k+1}} w^i_{k+1}\cdot\log\left[\mathbb{P}_O\big(z_{k+1}|x^i_{k+1}\big)\sum_{j=1}^{n_x}\mathbb{P}_T\big(x^i_{k+1}|x^j_k, a_k\big)w^j_k\right]$$

$$\ell \triangleq -\log\left[\sum_{i=1}^{n_x}\mathbb{P}_O\big(z_{k+1}|x^i_{k+1}\big)w^i_k\right]$$

$$+\sum_{i=1}^{n_x} w^i_{k+1}\cdot\log\left[\mathbb{P}_O\big(z_{k+1}|x^i_{k+1}\big)\sum_{j\in A^s_k}\mathbb{P}_T\big(x^i_{k+1}|x^j_k, a_k\big)w^j_k\right] \tag{58}$$

and where superscript s is the discrete level of simplification $s \in \{1, 2, \ldots, n_{\max}\}$, $m \triangleq \max_{\substack{x'\\x,a}} \mathbb{P}_T(x'|x, a)$ *and* A^s_k, $A^s_{k+1} \subseteq \{1, 2, \ldots, n_x\}$.

See proof in Appendix 11.6. Theorem 4 accommodates different sets $A^s_k \neq A^s_{k+1}$. These sets denote sets of particle indices from b_k and b_{k+1} for simplification level s. In general, each of these sets can have its own simplification level. However, this is out of the scope of this paper. Here, both sets A^s_k, A^s_{k+1} have the same simplification level, as well as the number of levels. Yet, the number of particles at each level can vary between A^s_k and A^s_{k+1}. Each subsequent level (low to high) defines a larger set of indices such that higher levels of simplification (i.e. more samples) correspond to tighter and lower levels of simplification correspond to looser bounds. Note that the bounds (57) and (58) actually use the original and simplified beliefs so it settles with equations (21) and (22).

Importantly, by caching the shared calculations of both bounds in the same time instance, we never repeat the calculation of these values and obtain maximal speedup. Without compromising on the solution's quality we are accelerating the online decision making process.

6.2. Bounds properties and analysis

We now turn to analysis of the bounds and investigation of their properties. Allow us to start from computational complexity. We then examine monotonicity and convergence of the bounds and reuse of calculations.

6.2.1. Computational complexity. Equations (57) and (58) suggest that the bounds are cheaper to calculate than $\widehat{\mathcal{H}}$ from (55), with complexity of $O(n^s_x \cdot n_x)$ instead of $O(n^2_x)$, where $n^s_x \triangleq |A^s_k| \equiv |A^s_{k+1}|$. Altogether, time saved for all belief nodes in the tree will result in the total speedup of our approach.

6.2.2. Monotonicy and convergence. **Theorem 5.** Monotonicity and convergence. *The bounds from* (56) *are monotonic (Assumption* 1) *and convergent (Assumption* 2) *to* (55).

See proof in Appendix 11.7. Finally, bounding (55) using Theorem 4 corresponds, in our general framework from Section 3, to (21).

6.2.3. Re-use of calculations. The bounds can be tightened on demand incrementally without an overhead. Moving from simplification level s to level $s+1$, corresponds to adding some m additional particles to b^s to get b^{s+1}. For bounds calculation, we store the highlighted elements of the matrix in Figure 8. This allows us to reuse the calculations when promoting the simplification level and **between the lower and the upper bounds** in a particular time index. Namely, after a few bounds-contracting iterations they are just the reward itself and the entire calculation is roughly time-equivalent to calculating the original reward. This will happen in a worst-case scenario.

We provide the theoretical time complexity analysis using the specific bounds (from Section 6.1) in Appendix 11.8. Now we are keen to present our simulations.

7. Adaptation overhead

Whereas the bounds presented in Section 6 are incremental repeated resimplifications may lead to actually slower decision making. This overhead is caused by additional algorithmics introduced by the resimplification routine. We can anticipate such scenarios when the actions are symmetrical in terms of the reward. However, as we empirically observed and will shortly present in the next section, in the setting of given belief tree the cases where the simplification is beneficial prevail. Especially in the LAZY variant since there Alg. 5 engages resimplification routine only at the root of the belief tree.

In the setting of MCTS the situation is slightly more complicated. In UCB we cannot prune actions for eternity but only dismiss up until the next arrival to the belief node. This is because when MAB (defined in Section 2.3.3) converges it switches the current best action with arrivals to the belief node; such a behavior necessitates our simplification approach to tighten the bounds for many candidate actions. As a result in a MCTS setting we obtain less speedup than in the setting of a given belief tree considering LAZY variant (Alg. 5). Nevertheless in some problems the simplification approach is invaluable, as for example, in the problem described in Section 8.1.3 and investigated in Section 8.3.5. Importantly, we can further accelerate resimplification routines by parallelization. However, this is out of the scope of this paper. All our implementations are single threaded.

8. Simulations and results

We evaluate our proposed framework and approaches in simulation considering the setting of nonparametric fully continuous POMDP. Our implementation is built upon the JuliaPOMDP package collection (Egorov et al., 2017). For our simulations, we used a 16 cores 11th Gen Intel(R) Core(TM) i9-11900K with 64 GB of RAM working at 3.50 GHz.

First, we study empirically the specific simplification and bounds from Section 6 and show that they become tighter as the number of particles increases. We, then benchmark our algorithms for planning in the setting of a given belief tree (Section 4) and in an anytime MCTS setting (Section 5). In the former setting, we compare SITH-BSP and LAZY-BSP against Sparse Sampling (Kearns et al., 2002). In an anytime MCTS setting, we compare SITH-PFT with PFT-DPW (Sunberg and Kochenderfer, 2018) and IPFT (Fischer and Tas, 2020). This performance evaluation is conducted considering three problems, as discussed next.

8.1. Problems under consideration

We proceed to the description of the evaluated problems. In two first problems, the immediate reward for b' is

$$\rho(b,a,z',b') = -(1-\lambda)\underset{x'\sim b'}{\mathbb{E}}[r(x')] - \lambda\widehat{\mathcal{H}}(b,a,z',b'). \tag{59}$$

8.1.1. Continuous light dark. Our first problem is *2D continuous Light-Dark problem*. The robot starts at some unknown point $x_0 \in \mathbb{R}^2$. In this world, there are spatially scattered beacons with known locations. Near the beacons, the obtained observations are less "noisy." The robot's mission is to get to the goal located at the upper right corner of the world. The state dependent reward in this problem is $r(x) = -\|x - x^{\text{goal}}\|_2^2$. The initial belief is $b_0 = \mathcal{N}(\mu_0, I\cdot\sigma_0)$, where we select $x_0 = \mu_0$ for actual robot initial state. The motion and observation models are

$$\mathbb{P}_T(x'|x,a) = \mathcal{N}(x+a, I\cdot\sigma_T), \tag{60}$$

and

$$O = \mathbb{P}_O(z|x) = \mathcal{N}\left(x - x^b, I\cdot\sigma_O\cdot\max\{d(x), d_{\min}\}\right), \tag{61}$$

respectively, where $d(x)$ is the ℓ^2 distance from robot's state x to the nearest beacon with known location denoted by x^b, and $d_{\min}$ is a tuneable parameter.

8.1.2. Target tracking. Our second problem is *2D continuous Target Tracking*. In this problem, we have a moving target in addition to the agent. In this problem the belief is maintained over both positions, the agent and the target. The state dependent reward in this problem is $r(x) = -\|x^{\text{agent}} - x^{\text{target}}\|_2^2$. The motion model of the target and the agent follows

$$\mathbb{P}_T(\cdot|x,a) = \mathcal{N}(x^{\text{agent}} + a^{\text{agent}}, \Sigma_T)\cdot\mathcal{N}(x^{\text{target}} + a^{\text{target}}, \Sigma_T),$$

where by x we denote the concatenated $\{x^{\text{agent}}, x^{\text{target}}\}$. For target actions we use a circular buffer with $\{\uparrow, \uparrow, \leftarrow\}$ action sequence of unit length motion primitives. For simplicity we assume that in inference as well as in the planning session the agent knows the target action sequence. The observation model is also the multiplication of the observation model from the previous section with the additional observation model due to a moving target. Thus, the overall observation model is

$$\mathbb{P}_O\left(\cdot|x;\{x^{b,i}\}_{i=1}\right) = \mathcal{N}\left(x^{\text{agent}}, \Sigma_O\left(x^{\text{agent}};\{x^{b,i}\}_{i=1}\right)\right)$$
$$\cdot\mathcal{N}(x^{\text{agent}} - x^{\text{target}}, \Sigma_O(x^{\text{agent}}, x^{\text{target}})),$$

where $\Sigma_O(x^{\text{agent}};\{x^{b,i}\}_{i=1})$ conforms to the observation model covariance described in Section 8.1.1 and

$$\Sigma_O(x^{\text{agent}}, x^{\text{target}}) \tag{62}$$
$$= \begin{cases} \sigma_T^2 I\|x^{\text{agent}} - x^{\text{target}}\|_2, \text{if } \|x^{\text{agent}} - x^{\text{target}}\|_2 \geq d_{\min} \\ \sigma_O^2 I, \text{else} \end{cases}.$$

Before the planning experiments we study of the entropy estimators and the bounds presented in Theorem 4.

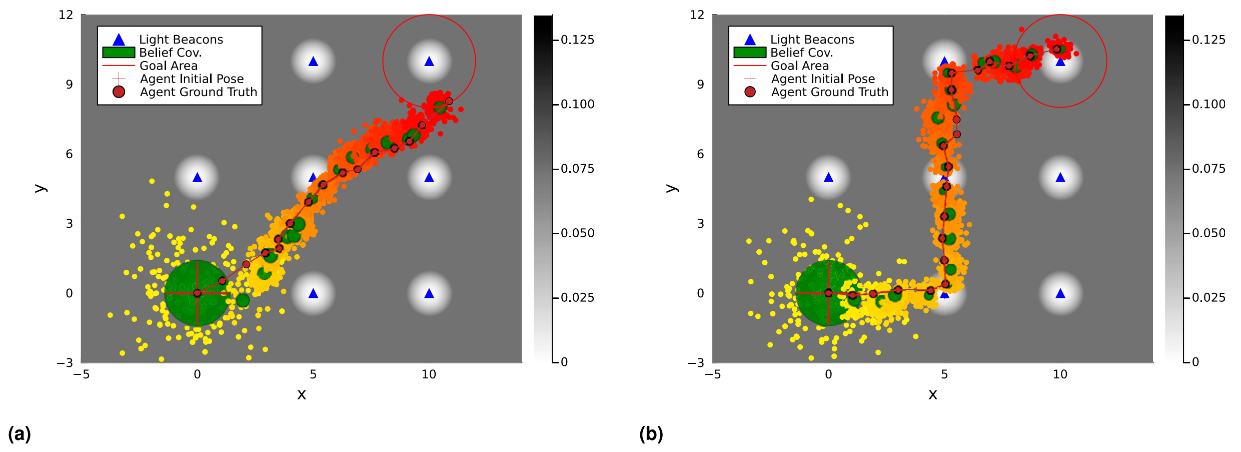


Figure 9. The plot shows the evolution of belief in terms of sets of particles along the actual trajectory of the robot. The color of the particles from yellow to red illustrates the evolution of the belief over time. The green ellipses represent the parametric Gaussian belief covariances obtained from update by the Kalman filter. The canvas color here is $\sigma_O = \sigma_T = 0.075$ as in equations (60) and (61), respectively. (a) Our first scenario. (b) Our second scenario.

8.1.3. Safe autonomous localization. Our third problem is a variation of the problem presented in Section 8.1.1. Here we change the reward to be the combination of localization reward and safety reward (Zhitnikov and Indelman, 2022a)

$$\begin{aligned}\rho(b,a,z',b') = {} & \overbrace{-\widehat{\mathcal{H}}(b,a,z',b')}^{\text{localization reward}} \\ & \underbrace{+s(2\cdot \mathbf{1}_{\{\mathbb{P}(\{x'\in\mathcal{X}^{\text{safe},'}\}|b')\geq\delta\}}(b')-1)}_{\text{safety reward}}.\end{aligned} \tag{63}$$

Such a safety reward divides the candidate actions into two sets, the safe set and the unsafe. If the safety parameter s is sufficiently large to assure that safe action is selected, these two sets are detached enough in terms of safety reward and the unsafe set is substantially inferior such that there is no point to calculate localization reward precisely over this set of actions. There, we can, without any harm for decision-making outcome, substitute differential entropy by the bounds at the low simplification levels. This aspect makes the simplification paradigm invaluable.

8.2. Entropy estimators and bounds study

In this section, we experiment with a passive case of the continuous 2D Light Dark problem from Section 8.1.1. Our goal is to study the various entropy estimators and our derived bounds from Section 6 over the estimator developed in Boers et al. (2010). In this study, we manually supply the robot with an action sequence to conduct. This results in a single lace of the beliefs corresponding to observations that the robot actually obtained by executing a given externally action sequence. We also provide some attempt in this section to compare estimated reward with the exact analytical counterpart.

Over this sequence of the beliefs, at each time instance of the sequence we calculate minus differential entropy estimator (information) in four ways. The first is the Boers estimator (Boers et al., 2010) and our bounds from Theorem 4. The second is KDE approximation as done by Fischer and Tas (2020). The third is the naive calculation of discrete entropy over the particles weights: $\widehat{\mathcal{H}}(b) = -\sum_i w^i \cdot \log w^i$. The fourth estimator is analytical and it requires additional explanation. If we make an unrealistic assumption that robot's ground truth state from which the observation has been taken is known, plug it into the covariance matrix of (61) and set prior belief to be Gaussian; the motion and observation models met all the requirement for the exact update by the Kalman filter (linear additive models). For the proof, see Thrun et al. (2005). In this case, the belief stays Gaussian and the differential entropy has closed form solution.

We have two scenarios. In the first scenario, the robot moves diagonally to the goal using a unit length action $\nearrow$ (Figure 9(a)) 15 times. Along the way, it passes close-by two beacons. Consequentially, the robot's information about its state peaks twice. In our second scenario, the robot moves five times to the right $\rightarrow$ followed by 10 times $\uparrow$ and again five times to the right $\rightarrow$ (Figure 9(b)).

The prior belief in this setting follows a Gaussian distribution $b_0 = \mathcal{N}\left(\begin{pmatrix}0.0\\0.0\end{pmatrix}, \begin{pmatrix}2.0 & 0.0\\0.0 & 2.0\end{pmatrix}\right)$, the motion and observation models parameters are $\sigma_O = \sigma_T = 0.075, d_{\min} = 0.0001$. The number of unsimplified belief weighted particles is $n_x = 300$. For creating initial weighted particles, we use the following proposal

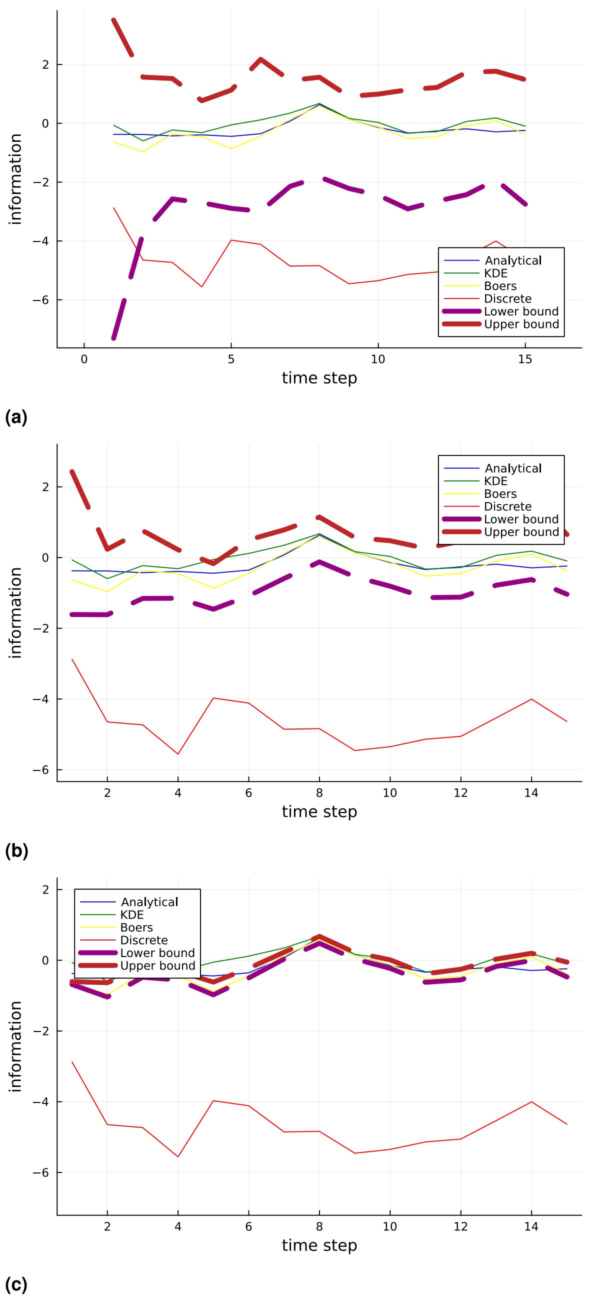


Figure 10. Bounds convergence for our first scenario $n_x = 300$: (a) $n^s_x = 30$ particles, (b) $n^s_x = 150$ particles, and (c) $n^s_x = 270$ particles.

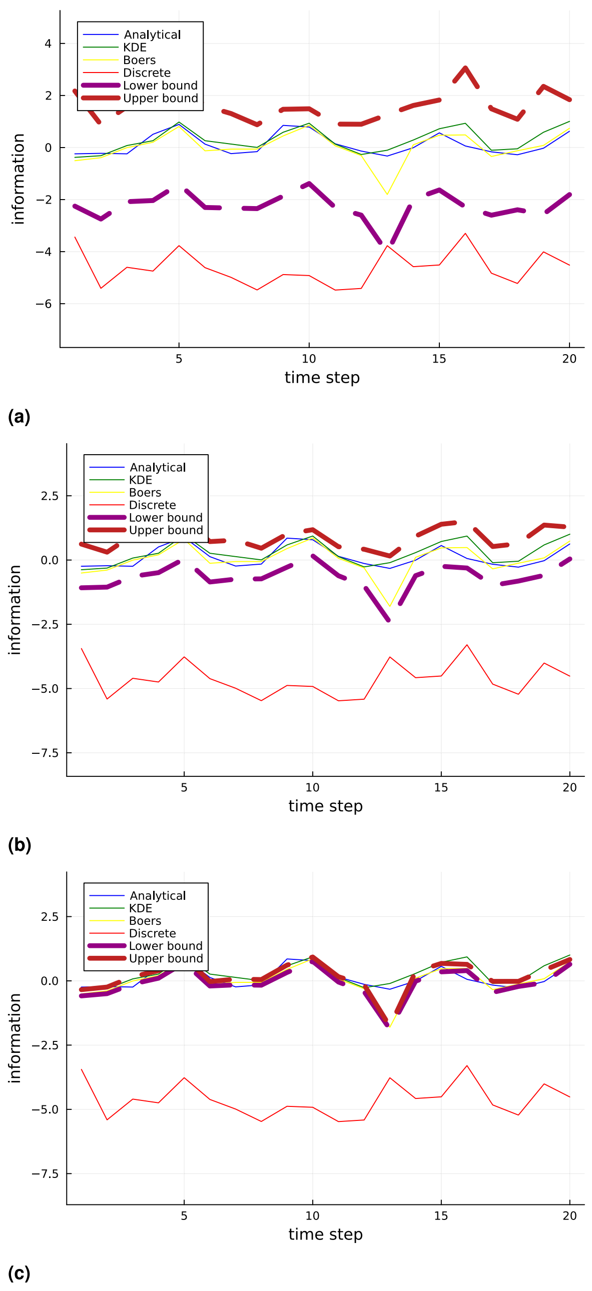


Figure 11. Bounds convergence for our second scenario $n_x = 300$: (a) $n^s_x = 30$, (b) $n^s_x = 150$ particles, and (c) $n^s_x = 270$ particles.

$$q = 0.25 \cdot \mathcal{N}\left(\begin{pmatrix} 0.0 \\ 1.0 \end{pmatrix}, \begin{pmatrix} 2.0 & 0.0 \\ 0.0 & 0.2 \end{pmatrix}\right) + \\ +0.25 \cdot \mathcal{N}\left(\begin{pmatrix} 1.0 \\ 0.0 \end{pmatrix}, \begin{pmatrix} 2.0 & 0.0 \\ 0.0 & 0.2 \end{pmatrix}\right) + \\ +0.25 \cdot \mathcal{N}\left(\begin{pmatrix} -1.0 \\ 0.0 \end{pmatrix}, \begin{pmatrix} 2.0 & 0.0 \\ 0.0 & 0.2 \end{pmatrix}\right) + \\ +0.25\mathcal{N}\left(\begin{pmatrix} 1.0 \\ -1.0 \end{pmatrix}, \begin{pmatrix} 2.0 & 0.0 \\ 0.0 & 0.2 \end{pmatrix}\right).$$

The initial weights are the ratio $w(x) = b_0(x)/q(x)$.

To examine the bounds monotonical convergence with a growing number of simplified belief particles we plot the bounds (57) and (58) for minus entropy estimator (55) alongside estimators described above for the entire robot trajectory of the beliefs.

The results for the first and second scenarios are provided in Figures 10 and 11, respectively. For both scenarios, we observe that the bounds become tighter as the number of particles of simplified belief n^s_x increases. We also witness that all estimators vary but the overall trend is similar, putting aside the discrete entropy over the weights.

The discrete entropy over the weights fails to adequately represent the uncertainty of the belief. This is an anticipated result. Let us proceed to the planning experiments.

8.3. Planning

In this section, we study and benchmark our efficient planning algorithms. In our algorithms 1 and 5, the tree is built by SS (Kearns et al., 2002) such that the given belief tree is obtained when the algorithm descends to the leaves. We first compare Alg. 1 and 5 versus SS. We then proceed to simulations in an anytime MCTS setting.

For all further experiments, the belief is approximated by a set of n_x weighted samples as in (53). The robot does replanning after each executed action.

8.3.1. Acceleration measures. Let us begin this section by describing our measures of acceleration. We report planning time speedup in terms of saved accesses to particles.

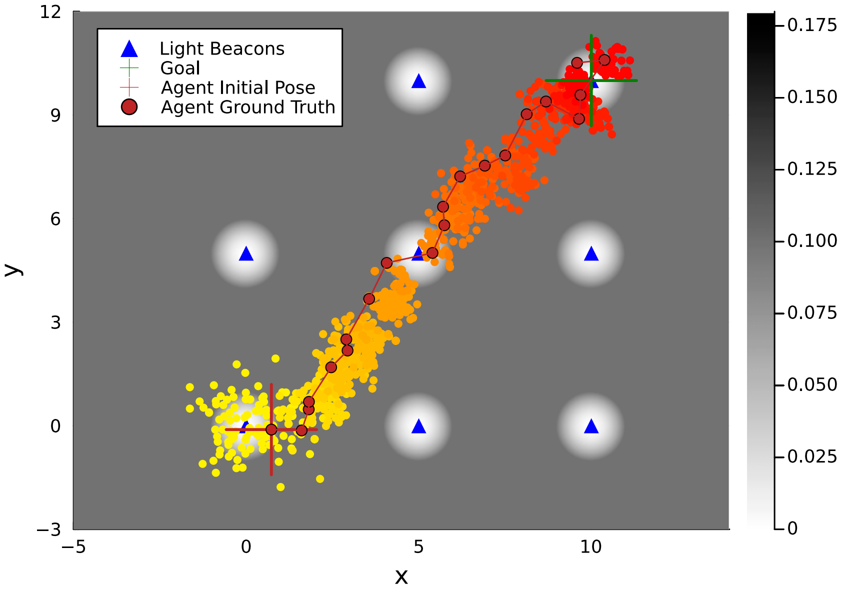


Figure 12. Exemplary 2D light dark problem planning scenario. Here, we present the first trial of configuration $\lambda = 0.5$ of Table 1.

Table 1. This table shows cumulative results of 20 consecutive alternating planning and execution sessions of the continuous light dark problem averaged over 15 trials. Each planning session creates a single belief tree to perform a search for optimal action. This given belief tree has 4,809 belief nodes. Overall, in 20 planning sessions, we have 96,180 belief nodes. The horizon in each planning session is $L = 3$. The number of observations sampled from each belief action node is $n_z^1 = 1$, $n_z^2 = 3$, and $n_z^3 = 3$ at the corresponding to superscripts depths 1, 2, and 3, respectively. This table examines the influence of various values of λ.

BSP Alg.	n_x	λ	Particles speedup (64)	Time speedup (66)	Resimpl. calls (recursive)	Motion model calls	Obs. model calls	Return $(\widehat{V})$
Alg 1 SITH	100	0.1	78.76 ± 0.20	64.44 ± 1.51	$2.05 \cdot 10^5 \pm 0.05 \cdot 10^5$	$3.13 \cdot 10^8 \pm 0.02 \cdot 10^8$	$9.62 \cdot 10^6 \pm 0.0$	−115.49 ± 16.58
Alg 5 LAZY			85.46 ± 1.22	71.59 ± 1.52	$10.71 \cdot 10^5 \pm 4.61 \cdot 10^5$	$2.38 \cdot 10^8 \pm 0.13 \cdot 10^8$	$9.62 \cdot 10^6 \pm 0.0$	−115.49 ± 16.58
SS						$9.62 \cdot 10^8 \pm 0.0$	$9.62 \cdot 10^6 \pm 0.0$	−115.49 ± 16.58
Alg 1 SITH	100	0.2	68.82 ± 0.32	53.59 ± 2.05	$3.36 \cdot 10^5 \pm 0.06 \cdot 10^5$	$4.22 \cdot 10^8 \pm 0.03 \cdot 10^8$	$9.62 \cdot 10^6 \pm 0.0$	−103.51 ± 14.91
Alg 5 LAZY			80.09 ± 1.52	65.01 ± 1.88	$25.65 \cdot 10^5 \pm 6.17 \cdot 10^5$	$3.01 \cdot 10^8 \pm 0.18 \cdot 10^8$	$9.62 \cdot 10^6 \pm 0.0$	−103.51 ± 14.91
SS						$9.62 \cdot 10^8 \pm 0.0$	$9.62 \cdot 10^6 \pm 0.0$	−103.51 ± 14.91
Alg 1 SITH	100	0.3	58.33 ± 0.52	42.76 ± 2.96	$4.13 \cdot 10^5 \pm 0.05 \cdot 10^5$	$5.40 \cdot 10^8 \pm 0.01 \cdot 10^8$	$9.62 \cdot 10^6 \pm 0.0$	−91.86 ± 13.88
Alg 5 LAZY			74.85 ± 2.63	58.94 ± 3.04	$42.66 \cdot 10^5 \pm 9.80 \cdot 10^5$	$3.59 \cdot 10^8 \pm 0.29 \cdot 10^8$	$9.62 \cdot 10^6 \pm 0.0$	−91.86 ± 13.88
SS						$9.62 \cdot 10^8 \pm 0.0$	$9.62 \cdot 10^6 \pm 0.0$	−91.86 ± 13.88
Alg 1 SITH	100	0.4	45.66 ± 0.83	29.33 ± 4.78	$4.70 \cdot 10^5 \pm 0.04 \cdot 10^5$	$6.84 \cdot 10^8 \pm 0.08 \cdot 10^8$	$9.62 \cdot 10^6 \pm 0.0$	−80.44 ± 11.77
Alg 5 LAZY			69.94 ± 1.89	53.85 ± 2.56	$59.05 \cdot 10^5 \pm 8.76 \cdot 10^5$	$4.16 \cdot 10^8 \pm 0.22 \cdot 10^8$	$9.62 \cdot 10^6 \pm 0.0$	−80.44 ± 11.77
SS						$9.62 \cdot 10^8 \pm 0.0$	$9.62 \cdot 10^6 \pm 0.0$	−80.44 ± 11.77
Alg 1 SITH	100	0.5	34.46 ± 0.79	18.98 ± 4.16	$5.27 \cdot 10^5 \pm 0.05 \cdot 10^5$	$7.92 \cdot 10^8 \pm 0.01 \cdot 10^8$	$9.62 \cdot 10^6 \pm 0.0$	−66.3 ± 8.0
Alg 5 LAZY			63.6 ± 2.23	46.67 ± 2.81	$81.48 \cdot 10^5 \pm 8.52 \cdot 10^5$	$4.87 \cdot 10^8 \pm 0.24 \cdot 10^8$	$9.62 \cdot 10^6 \pm 0.0$	−66.3 ± 8.0
SS						$9.62 \cdot 10^8 \pm 0.0$	$9.62 \cdot 10^6 \pm 0.0$	−66.3 ± 8.0
Alg 1 SITH	100	0.6	25.09 ± 0.89	12.05 ± 4.83	$5.85 \cdot 10^5 \pm 0.05 \cdot 10^5$	$8.64 \cdot 10^8 \pm 0.05 \cdot 10^8$	$9.62 \cdot 10^6 \pm 0.0$	−55.36 ± 6.93
Alg 5 LAZY			56.32 ± 2.72	38.45 ± 3.65	$113.26 \cdot 10^5 \pm 11.45 \cdot 10^5$	$5.71 \cdot 10^8 \pm 0.28 \cdot 10^8$	$9.62 \cdot 10^6 \pm 0.0$	−55.36 ± 6.93
SS						$9.62 \cdot 10^8 \pm 0.0$	$9.62 \cdot 10^6 \pm 0.0$	−55.36 ± 6.93

The following speedup is based on the final number of simplified beliefs particles required for planning

$$\frac{\sum_i \left(n_{i,x}^2 - n_{i,x}^s n_{i,x}\right)}{\sum_i n_{i,x}^2} \cdot 100, \tag{64}$$

where the summation is over the future posterior beliefs in all the belief trees in a number of a consecutive planning sessions in particular scenario. Equation (64) measures relative speedup without time spent on re-simplifications. It is calculated at the end of several consecutive planning sessions. To calculate speedup according to (64) one shall pick up the **final** number of particles of simplified belief $n_{i,x}^s$ used for the simplified reward for each belief node i, sum over all the nodes of the belief trees (given or constructed on the fly) from planning sessions, make a calculation portrayed by (64). Importantly, acceleration measure (64) assumes that time of evaluating the motion and observation models do not vary from one evaluation to another. If the number of belief particles is not depending on the belief ($n_{i,x} = n_x$), we can further simplify the (64) to

$$\frac{\sum_i \left(n_x - n_{i,x}^s\right)}{\sum_i n_x} \cdot 100. \tag{65}$$

To calculate planning time speedup, we use the following metric

$$\frac{t_{\text{baseline}} - t_{\text{our}}}{t_{\text{baseline}}} \cdot 100. \tag{66}$$

If the quantities (64) and (66) are identical we can conclude that there will be no overhead from re-simplifications and adapting the bounds. Note also that in the first place it is not clear how many particles n_x for belief representation to take. The number of particles n_x shall be as large as possible due to fact that we do not know when the belief represented by weighted particles will converge to the corresponding theoretical belief.

Table 2. This table shows cumulative results of 20 consecutive alternating planning and execution sessions averaged over 15 trials of continuous light dark problem. The given belief tree in a single planning session has 4,809 belief nodes. Overall, in 20 planning sessions, we have 96,180 belief nodes. The horizon in each planning session is $L = 3$. The number of observations sampled from each belief action node is $n_z^1 = 1, n_z^2 = 3$, and $n_z^3 = 3$ at the corresponding to superscripts depths 1, 2, and 3, respectively. In this table, we examine influence of various number of belief particles.

BSP Alg.	n_x	λ	Particles speedup (64)	Time speedup (66)	Resimpl. calls (recursive)	Motion model calls	Obs. model calls	Return ($\widehat{V}$)
Alg 1 SITH	200	0.5	34.1 ± 0.8	25.01 ± 5.11	$5.30 \cdot 10^5 \pm 0.04 \cdot 10^5$	$31.80 \cdot 10^8 \pm 0.25 \cdot 10^8$	$19.24 \cdot 10^6 \pm 0.0$	-69.36 ± 7.95
Alg 5 LAZY			64.0 ± 2.98	51.71 ± 4.9	$83.95 \cdot 10^5 \pm 10.24 \cdot 10^5$	$19.35 \cdot 10^8 \pm 1.29 \cdot 10^8$	$19.24 \cdot 10^6 \pm 0.0$	-69.36 ± 7.95
SS						$38.47 \cdot 10^8 \pm 0.0$	$19.24 \cdot 10^6 \pm 0.0$	-69.36 ± 7.95
Alg 1 SITH	300	0.5	33.84 ± 0.83	18.67 ± 2.02	$5.30 \cdot 10^5 \pm 0.04 \cdot 10^5$	$71.74 \cdot 10^8 \pm 0.58 \cdot 10^8$	$28.85 \cdot 10^6 \pm 0.0$	-68.29 ± 8.42
Alg 5 LAZY			63.39 ± 3.44	47.34 ± 3.72	$84.09 \cdot 10^5 \pm 10.66 \cdot 10^5$	$43.91 \cdot 10^8 \pm 3.09 \cdot 10^8$	$28.85 \cdot 10^6 \pm 0.0$	-68.29 ± 8.42
SS						$86.56 \cdot 10^8 \pm 0.0$	$28.85 \cdot 10^6 \pm 0.0$	-68.29 ± 8.42
Alg 1 SITH	400	0.5	33.97 ± 0.85	25.34 ± 3.44	$6.65 \cdot 10^5 \pm 0.06 \cdot 10^5$	$181.50 \cdot 10^8 \pm 1.41 \cdot 10^8$	$54.51 \cdot 10^6 \pm 0.0$	-67.92 ± 11.52
Alg 5 LAZY			66.06 ± 2.3	53.74 ± 3.4	$106.90 \cdot 10^5 \pm 15.73 \cdot 10^5$	$105.35 \cdot 10^8 \pm 5.75 \cdot 10^8$	$54.51 \cdot 10^6 \pm 0.0$	-67.92 ± 11.52
SS						$218.05 \cdot 10^8 \pm 0.0$	$54.51 \cdot 10^6 \pm 0.0$	-67.92 ± 11.52

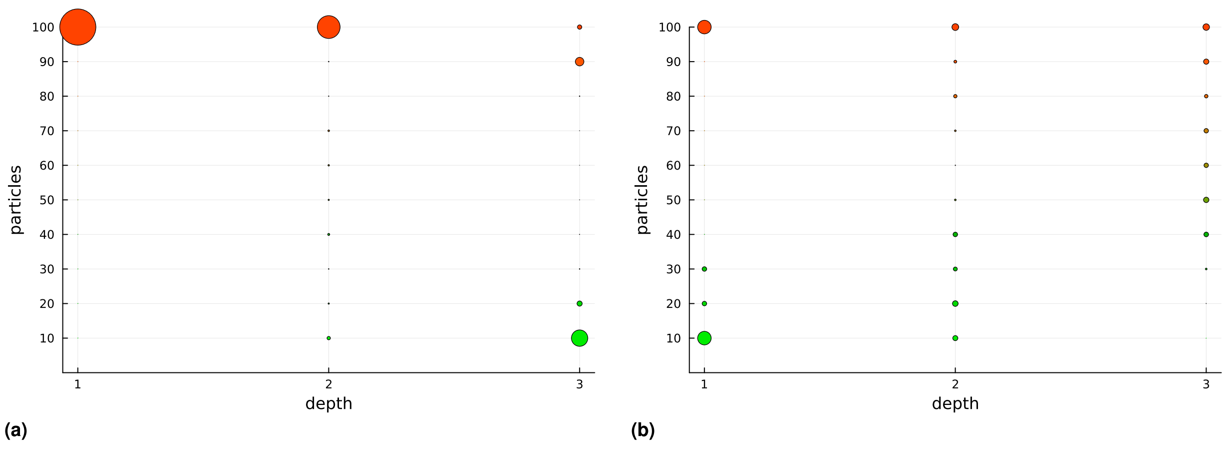


Figure 13. Simplification levels at each depth of the given belief tree of light dark problem (Section 8.1.1) after determining best action for one of the planning sessions. Here, we present planning session 6 of the first trial of configuration $\lambda = 0.5$ of Table 1. The radius of circles represents the fraction of all nodes at a particular depth that have a particular simplification level. This figure is associated with Table 3. (a) LAZY-SITH-BSP Alg 5 and (b) SITH-BSP Alg 1.

To thoroughly study the acceleration yielded by our simplification paradigm, we calculate total speedup over a number of the consecutive planning sessions in terms of particles in accordance to (64) and in terms of time in accordance to (66).

8.3.2. Results for 2D continuous light-dark in the setting of a given belief tree. We start from the problem described in Section 8.1.1. Our action space is constituted by motion primitives of unit length $\mathcal{A} = \{\rightarrow, \nearrow, \uparrow, \nwarrow, \leftarrow, \swarrow, \downarrow, \searrow\}$. In this problem the selected parameters are $\sigma_T = \sigma_O = 0.1$, $d_{\min} = 0.0001$, $\gamma = 0.95$. We simulate 15 trials of 20 consecutive alternating planning and action execution sessions. Figure 12 shows an exemplary trial of 20 executions of the best action identified by the robot.

We investigate the influence of the parameter λ on speedup in Table 1 and the impact of changing the number of particles in Table 2. In both tables, we see the particles speedup (column 4) and the time speedup (column 5). As expected with increasing values of λ (column 3) the speedup diminishes. LAZY-BSP (Alg. 5) produces larger speedup in terms of particles (column 4) and time (column 5) than SITH-BSP (Alg. 1). All three algorithms always selected the same optimal action. We observe that the return is always identical (column 9). Significant time speedup is obtained in the range of 35%−70% for LAZY-BSP depending on the values of λ. For the SITH-BSP we see less time speedup ranging from 65% to 10% with increasing λ.

In all tables, the number of motion and observation model calls does not include belief update calls but only the calls for reward or bounds calculation. The number of accesses to the observation model is always the same for all three algorithms (column 8). This agrees with the structure of the bounds (57) and (58). For the baseline SS, up to rounding errors, the number of motion model accesses, as we anticipated, is the squared number of unsimplified belief particles multiplied by number belief nodes in the tree minus one for root belief, multiplied by number of planning sessions (column 7 in the tables). This is in agreement with (55). Also, for all three algorithms the number of accesses to the observation model was the number of particles of unsimplified belief minus one for root belief, multiplied by the number of belief nodes in the tree, multiplied by the number of planning sessions.

We see that, while having larger particle speedup (column 3), LAZY-BSP makes more resimplification calls (column 6) than SITH-BSP. Observing the histograms of simplification levels in Figure 13, we understand that LAZY variant of resimplification strategy leads to lower simplification levels of the rewards at the deepest level of a given belief tree. This was expected since the rewards at the upper levels of the belief tree participate in more laces and therefore their simplification level is promoted more times (See Alg. 5). In addition, at the lowest levels reside more beliefs and corresponding rewards. This fact is corroborated by Table 3 where we witness that LAZY-BSP yields more

Table 3. This table displays the numbers of the beliefs at each simplification level in a given tree after the identification of optimal action at the root b_k. Here, we investigate light dark problem and belief tree as in Figure 13. The given belief tree has 4,809 belief nodes.

							Simpl. level, particles									
							$s = 1$	$s = 2$	$s = 3$	$s = 4$	$s = 5$	$s = 6$	$s = 7$	$s = 8$	$s = 9$	$s = 10$
BSP Alg.	n_x	n_z^1	n_z^2	n_z^3	λ	L	$n_x^s = 10$	$n_x^s = 20$	$n_x^s = 30$	$n_x^s = 40$	$n_x^s = 50$	$n_x^s = 60$	$n_x^s = 70$	$n_x^s = 80$	$n_x^s = 90$	$n_x^s = 100$
Alg 5 LAZY	100	1	3	3	0.5	3	2103	666	91	47	14	10	15	88	1094	680
Alg 1 SITH							30	61	241	618	696	567	576	465	684	870

beliefs, in the given in belief tree, with lower simplification levels than SITH-BSP.

8.3.3. Results for 2D continuous target tracking in the setting of a given belief tree. Our action space is $\mathcal{A} = \{\rightarrow, \nearrow, \uparrow, \nwarrow, \leftarrow, \swarrow, \downarrow, \searrow, \text{Null}\}$, where action Null means that agent doesn't take any action. In this problem we selected the parameters to be $d_{\min} = 0.0001$, $\Sigma_T = I \cdot \sigma_T$, where $\sigma_T = 0.1$ and $\sigma_O = 0.1$, $\gamma = 0.95$.

We simulate 15 trials of 15 consecutive alternating planning sessions and the executions by the robot of the selected optimal action. Figure 14 shows an exemplary trial. We show the agent particles in Figure 14(a) and the target particles in Figure 14(b). Similar to the previous section, we study speedup with growing λ in Table 4 and as function of various amounts of belief particles in Table 5. Again we observe that speedup diminishes with growing λ; LAZY-BSP (Alg. 5) produces a larger speedup in terms of particles (column 4) and time (column 5) than SITH-BSP (Alg. 1); accesses to motion and observation models are as expected; the return is identical for three algorithms.

In Figure 15, which is associated with Table 6, we observe that to select an optimal action LAZY-BSP

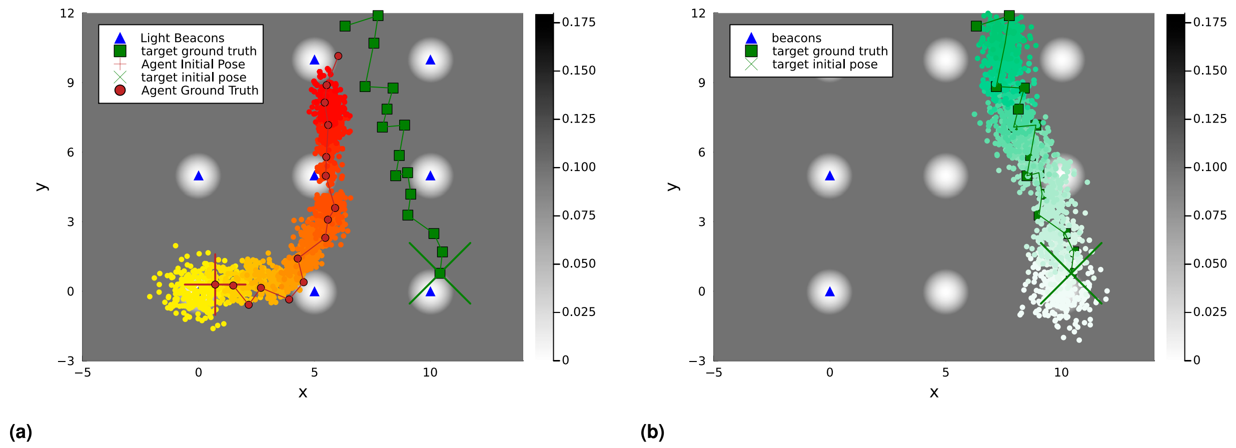


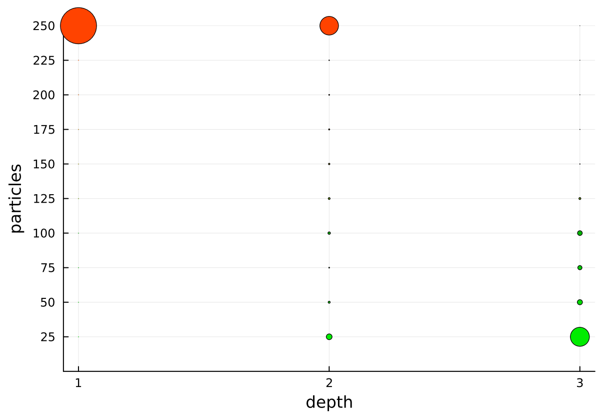
Figure 14. In this illustration we show second trial of Table 5, configuration $n_x = 250$. The canvas color here is $\sigma_O = \sigma_T = 0.1$. (a) Agent particles. (b) Target particles.

Table 4. This table shows cumulative results of 15 consecutive alternating planning and action execution sessions averaged over 15 trials of continuous target tracking problem. The given in a single planning session belief tree has 6,814 belief nodes. Overall, in 15 planning sessions, we have 102,210 belief nodes. The horizon in each planning session is $L = 3$. The number of observations sampled from each belief action node is $n_z^1 = 1, n_z^2 = 3$, and $n_z^3 = 3$ at corresponding to superscripts depths 1, 2, and 3, respectively. In this table, we examine influence of various values of λ.

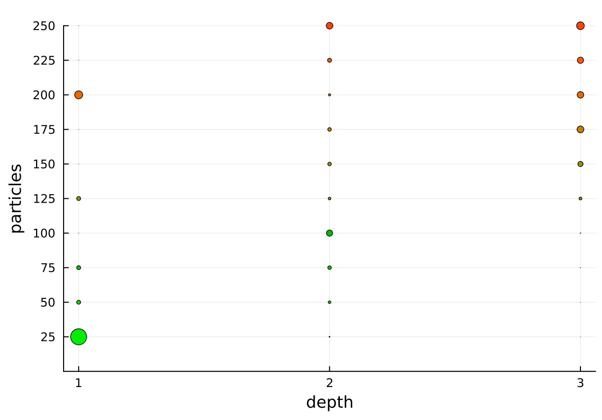
BSP Alg.	n_x	λ	Particles speedup (64)	Time speedup (66)	Resimpl. calls (recursive)	Motion model calls	Obs. model calls	Return $(\widehat{V})$
Alg 1 SITH	100	0.1	77.43 ± 0.26	60.3 ± 2.21	$1.69 \cdot 10^5 \pm 0.04 \cdot 10^5$	$3.48 \cdot 10^8 \pm 0.03 \cdot 10^8$	$10.22 \cdot 10^6 \pm 0.0$	-79.87 ± 9.69
Alg 5 LAZY			86.97 ± 1.28	71.18 ± 2.42	$7.44 \cdot 10^5 \pm 3.09 \cdot 10^5$	$2.32 \cdot 10^8 \pm 0.16 \cdot 10^8$	$10.22 \cdot 10^6 \pm 0.0$	-79.87 ± 9.69
SS						$10.22 \cdot 10^8 \pm 0.0$	$10.22 \cdot 10^6 \pm 0.0$	-79.87 ± 9.69
Alg 1 SITH	100	0.2	64.64 ± 0.57	46.39 ± 2.27	$2.60 \cdot 10^5 \pm 0.04 \cdot 10^5$	$5.03 \cdot 10^8 \pm 0.07 \cdot 10^8$	$10.22 \cdot 10^6 \pm 0.0$	-73.38 ± 9.8
Alg 5 LAZY			83.52 ± 1.7	67.24 ± 2.62	$16.52 \cdot 10^5 \pm 5.56 \cdot 10^5$	$2.75 \cdot 10^8 \pm 0.22 \cdot 10^8$	$10.22 \cdot 10^6 \pm 0.0$	-73.38 ± 9.8
SS						$10.22 \cdot 10^8 \pm 0.0$	$10.22 \cdot 10^6 \pm 0.0$	-73.38 ± 9.8
Alg 1 SITH	100	0.3	49.57 ± 0.93	29.25 ± 2.39	$3.14 \cdot 10^5 \pm 0.05 \cdot 10^5$	$6.86 \cdot 10^8 \pm 0.10 \cdot 10^8$	$10.44 \cdot 10^6 \pm 0.0$	-66.29 ± 9.3
Alg 5 LAZY			79.83 ± 2.55	63.34 ± 3.45	$26.61 \cdot 10^5 \pm 8.41 \cdot 10^5$	$3.21 \cdot 10^8 \pm 0.30 \cdot 10^8$	$10.44 \cdot 10^6 \pm 0.0$	-66.29 ± 9.3
SS						$10.22 \cdot 10^8 \pm 0.0$	$10.44 \cdot 10^6 \pm 0.0$	-66.29 ± 9.3
Alg 1 SITH	100	0.4	35.75 ± 1.09	14.45 ± 2.85	$3.61 \cdot 10^5 \pm 0.06 \cdot 10^5$	$8.33 \cdot 10^8 \pm 0.09 \cdot 10^8$	$10.44 \cdot 10^6 \pm 0.0$	-59.99 ± 8.05
Alg 5 LAZY			74.38 ± 3.5	55.69 ± 4.38	$42.74 \cdot 10^5 \pm 12.16 \cdot 10^5$	$3.90 \cdot 10^8 \pm 0.38 \cdot 10^8$	$10.44 \cdot 10^6 \pm 0.0$	-59.99 ± 8.05
SS						$10.22 \cdot 10^8 \pm 0.0$	$10.44 \cdot 10^6 \pm 0.0$	-59.99 ± 8.05
Alg 1 SITH	100	0.5	25.51 ± 1.04	6.44 ± 2.49	$4.05 \cdot 10^5 \pm 0.06 \cdot 10^5$	$9.18 \cdot 10^8 \pm 0.08 \cdot 10^8$	$10.44 \cdot 10^6 \pm 0.0$	-53.15 ± 7.03
Alg 5 LAZY			67.76 ± 3.88	47.94 ± 5.08	$63.18 \cdot 10^5 \pm 15.71 \cdot 10^5$	$4.75 \cdot 10^8 \pm 0.44 \cdot 10^8$	$10.44 \cdot 10^6 \pm 0.0$	-53.15 ± 7.03
SS						$10.22 \cdot 10^8 \pm 0.0$	$10.44 \cdot 10^6 \pm 0.0$	-53.15 ± 7.03
Alg 1 SITH	100	0.6	18.06 ± 1.0	2.63 ± 2.32	$4.43 \cdot 10^5 \pm 0.06 \cdot 10^5$	$9.65 \cdot 10^8 \pm 0.06 \cdot 10^8$	$10.44 \cdot 10^6 \pm 0.0$	-46.97 ± 7.14
Alg 5 LAZY			59.53 ± 3.78	38.14 ± 4.69	$89.27 \cdot 10^5 \pm 15.03 \cdot 10^5$	$5.77 \cdot 10^8 \pm 0.43 \cdot 10^8$	$10.44 \cdot 10^6 \pm 0.0$	-46.97 ± 7.14
SS						$10.22 \cdot 10^8 \pm 0.0$	$10.44 \cdot 10^6 \pm 0.0$	-46.97 ± 7.14

Table 5. This table shows cumulative results of 15 consecutive alternating planning and execution sessions averaged over 15 trials of continuous target tracking problem. The given belief tree has 6,814 belief nodes. Overall, in 15 planning sessions, we have 102,210 belief nodes. The horizon in each planning session is $L = 3$. The number of observations sampled from each belief action node is $n_z^1 = 1, n_z^2 = 3$, and $n_z^3 = 3$ at corresponding to superscripts depths 1, 2, and 3, respectively. In this table, we examine various numbers of belief particles.

BSP Alg.	n_x	λ	Particles speedup (64)	Time speedup (66)	Resimpl. calls (recursive)	Motion model calls	Obs. model calls	Return $(\widehat{V})$
Alg 1 SITH	150	0.5	25.19 ± 0.94	8.72 ± 2.4	$4.03 \cdot 10^5 \pm 0.03 \cdot 10^5$	$20.71 \cdot 10^8 \pm 0.15 \cdot 10^8$	$15.33 \cdot 10^6 \pm 0.0$	-54.0 ± 8.16
Alg 5 LAZY			68.36 ± 2.66	50.23 ± 3.2	$63.14 \cdot 10^5 \pm 9.15 \cdot 10^5$	$10.55 \cdot 10^8 \pm 0.65 \cdot 10^8$	$15.33 \cdot 10^6 \pm 0.0$	-54.0 ± 8.16
SS						$22.10 \cdot 10^8 \pm 0.0$	$15.33 \cdot 10^6 \pm 0.0$	-54.0 ± 8.16
Alg 1 SITH	250	0.5	23.87 ± 0.98	11.01 ± 3.93	$4.11 \cdot 10^5 \pm 0.05 \cdot 10^5$	$58.10 \cdot 10^8 \pm 0.40 \cdot 10^8$	$25.55 \cdot 10^6 \pm 0.0$	-55.57 ± 9.59
Alg 5 LAZY			66.18 ± 3.35	51.51 ± 3.83	$70.02 \cdot 10^5 \pm 12.74 \cdot 10^5$	$30.79 \cdot 10^8 \pm 2.37 \cdot 10^8$	$25.55 \cdot 10^6 \pm 0.0$	-55.57 ± 9.59
SS						$63.88 \cdot 10^8 \pm 0.0$	$25.55 \cdot 10^6 \pm 0.0$	-55.57 ± 9.59
Alg 1 SITH	350	0.5	23.95 ± 1.07	40.18 ± 10.29	$4.11 \cdot 10^5 \pm 0.03 \cdot 10^5$	$113.81 \cdot 10^8 \pm 0.89 \cdot 10^8$	$35.77 \cdot 10^6 \pm 0.0$	-55.62 ± 8.73
Alg 5 LAZY			66.36 ± 2.58	67.17 ± 4.86	$69.40 \cdot 10^5 \pm 10.08 \cdot 10^5$	$60.19 \cdot 10^8 \pm 3.62 \cdot 10^8$	$35.77 \cdot 10^6 \pm 0.0$	-55.62 ± 8.73
SS						$125.21 \cdot 10^8 \pm 0.0$	$35.77 \cdot 10^6 \pm 0.0$	-55.62 ± 8.73



(a)



(b)

Figure 15. Simplification levels at each depth of the given belief tree of target tracking problem (Section 8.1.2) after determining best action for one of the planning sessions. Here, we present planning session 6 of the first trial of configuration $n_x = 250$ of Table 5. The radius of circles represents the fraction of all nodes at particular depth that have a particular simplification level. This figure is associated with Table 6. (a) LAZY-SITH-BSP Alg 5 and (b) SITH-BSP Alg 1.

leaves more beliefs with lower simplification levels at the bottom of the given belief tree and produces more beliefs with lower simplification levels than SITH-BSP. A significant time speedup is obtained in the range of 30%–70% for LAZY-BSP depending on the values of λ. For the SITH-BSP, we see less time speedup ranging from 60% to 2% with increasing λ. The same best action was identified by SITH-BSP, LAZY-BSP and SS in all cases. Interestingly, in configuration $n_x = 350$ of Table 5, for the first time, we obtained that time speedup (66) is larger than particle speedup (64). This points to the fact that this run was so long due to large number of un-simplified belief particles $n_x = 350$ so that the time of access to motion and observation models varied.

8.3.4. Experiments with MCTS. In an anytime setting of MCTS we focus on the 2*D*-continuous light dark problem from Section 8.1.1. We place a single "light beacon" in the continuous world. Here we changed the reward. The agent's goal is to get to location (0, 0) and execute the terminal action - Null. Executing it within a radius of 0.5 from (0, 0) will give the agent a reward of 200, and executing it outside the radius will yield a negative reward of −200. For all other actions, the multi-objective reward function is $\rho(b, a, z, b') = -\underset{x \sim b'}{\mathbb{E}}[\|x\|_2] - \widehat{\mathcal{H}}(b, a, z, b')$. The agent can move in eight evenly spread directions $\mathcal{A} = \{\rightarrow, \nearrow, \uparrow, \nwarrow, \leftarrow, \swarrow, \downarrow, \searrow, \text{Null}\}$. Motion and observation models, and the initial belief are $\mathbb{P}_T(\cdot|x, a) = \mathcal{N}(x + a, \Sigma_T)$, $\mathbb{P}_O(z|x) = \mathcal{N}\left(x, \min\left\{1, \|x - x^b\|_2^2\right\} \cdot \Sigma_O\right)$, and $b_0 = \mathcal{N}(x_0, \Sigma_0)$, respectively. x^b is the 2D location of the beacon and all covariance matrices are diagonal (i.e., $\Sigma = I \cdot \sigma^2$).

We selected the following parameters $x_0 = \begin{pmatrix} -5.5 \\ 0.0 \end{pmatrix}, \Sigma_0 = \begin{pmatrix} 0.2 & 0.0 \\ 0.0 & 0.2 \end{pmatrix}, \sigma_T = \sigma_O = 0.075$. We experiment with 10 different configurations (rows of Table 7) that differ in n_x (number of particles), L (MCTS simulation depth), and #iter (number of MCTS simulation iterations per planning session). Each scenario comprises 10 planning sessions, that is, the agent performs up to 10 planning action-executing iterations. The scenario stops if the best action determined in planning is Null. We repeat each experiment 25 times. In each such repetition we run

Table 6. This table displays the numbers of the beliefs at each simplification level in given tree after the identification of optimal action at the root b_k. Here, we investigate target tracking problem and belief tree as in Figure 15. The size of given belief tree is 6,814 belief nodes.

							Simpl. level, particles									
							$s=1$	$s=2$	$s=3$	$s=4$	$s=5$	$s=6$	$s=7$	$s=8$	$s=9$	$s=10$
BSP Alg.	n_x	n_z^1	n_z^2	n_z^3	λ	L	$n_x^s=25$	$n_x^s=50$	$n_x^s=75$	$n_x^s=100$	$n_x^s=125$	$n_x^s=150$	$n_x^s=175$	$n_x^s=200$	$n_x^s=225$	$n_x^s=250$
Alg 5 LAZY	250	1	3	3	0.5	3	3487	949	776	884	379	106	48	34	11	139
Alg 1 SITH							10	19	46	144	538	966	1266	1208	1164	1452

PFT-DPW and SITH-PFT with the same seed and calculate the relative time speedup in percentage according to (66), where $t_{\text{PFT}-\text{DPW}}$ and $t_{\text{SITH}-\text{PFT}}$ are running times of a baseline and our methods, respectively.

In all different configurations, we obtained significant time speedup of approximately 20% while achieving the exact same solution compared to PFT. In Table 7, we report the mean and standard error of (66) as well as maximum and minimum value. Remarkably, we observe that we never slowdown the PFT-DPW with SITH-PFT. We also present total running times of 25 repetitions of at most 10 (the simulation stops if best identified action is Null) planning sessions in Table 8. Note that we divided the total planning time by the number of planning sessions in each repetition.

An illustration of evaluated scenario can be found in Figure 16. Note that SITH-PFT (Figure 16(a)) yields an identical to PFT solution (Figure 16(b)) while IPFT demonstrates a severely degraded behavior. We remind the purpose of our work is to speedup the PFT approach when coupled with information-theoretic reward. Since the two algorithms produce identical belief trees and action at the end of each planning session, there is no point reporting the algorithms *identical* performances (apart from planning time).

8.3.5. Localization with collision avoidance solved by MCTS. In this section, we investigate the application of three algorithms, IPFT (Fischer and Tas, 2020), PFT-DPW (Sunberg and Kochenderfer, 2018) and our SITH-PFT encapsulated by Alg. 7. The algorithmic implementation of IPFT boils down to making more simulations inside IPFT with substantially less number of belief particles subsampled from root belief.

Further, we discuss the quality and speed of IPFT. Representation of the belief with a tiny amount of particles induces larger error in differential entropy estimation and other parts of the reward function such as, for example, soft safety reward component in (63). The authors of (Fischer and Tas, 2020) claim that IPFT averages differential entropies calculated from tiny subsets subsampled from the particle belief. However, observing the SIMULATE routine (similar to our in Alg. 7) in (Fischer and Tas, 2020), we see that in practice this average is obtained through more simulations, starting from a new subsample from the root belief, with less number of particles, thereby averaging entropies calculated from different beliefs with less number of particles, but same history of actions and observations. The parameter K in (Fischer and Tas, 2020) in practice is the visitation count $N(b)$ of each belief in the belief tree. There is no direct control of this parameter. In other words, to make a proper comparison we shall increase the number of SIMULATE calls inside IPFT by a factor $K = n_x/m$, where m is the size of the subsample from a belief represented by n_x particles. In such a way in both belief trees, built by IPFT and PFT-DPW, there are the same number of total particles. This is in contrast to using the same number of calls to SIMULATE in both trees. If the number of calls to SIMULATE is the same the number

Table 7. Time speedup (66) obtained SITH-PFT versus PFT-DPW. The rows are different configurations of the number of belief particles n_x, maximal tree depth L, and the number of iterations per planning session. In all simulations SITH-PFT and PFT-DPW declared *identical* actions as optimal and exhibited *identical* belief trees in terms of connectivity and visitation counts.

(n_x, L, #iter.)	mean ± std	max.	min.
(50, 30, 200)	19.35 ± 6.34	30.17	7.99
(50, 50, 500)	17.43 ± 5.4	33.49	10.72
(100, 30, 200)	21.97 ± 8.74	49.24	7.36
(100, 50, 500)	22.54 ± 6.33	36.09	13.65
(200, 30, 200)	26.27 ± 9.36	42.43	11.17
(200, 50, 500)	26.17 ± 7.64	44.31	14.43
(400, 30, 200)	21.88 ± 8.47	37.04	10.34
(400, 50, 500)	21.71 ± 6.01	32.69	9.67
(600, 30, 200)	20.27 ± 7.38	32.95	8.77
(600, 50, 500)	19.93 ± 6.48	31.26	6.49

Table 8. Total runtime of 25 repetitions of two algorithms.

(n_x, L, #iter.)	Algorithm	Tot. plan. time (sec)
(50, 30, 200)	PFT-DPW	49.7
	SITH-PFT	40.25
(50, 50, 500)	PFT-DPW	125.05
	SITH-PFT	103.71
(100, 30, 200)	PFT-DPW	185.47
	SITH-PFT	145.08
(100, 50, 500)	PFT-DPW	460.29
	SITH-PFT	357.57
(200, 30, 200)	PFT-DPW	709.66
	SITH-PFT	526.18
(200, 50, 500)	PFT-DPW	1755.08
	SITH-PFT	1298.86
(400, 30, 200)	PFT-DPW	2672.56
	SITH-PFT	2099.0
(400, 50, 500)	PFT-DPW	6877.24
	SITH-PFT	5403.91
(600, 30, 200)	PFT-DPW	6335.09
	SITH-PFT	5056.96
(600, 50, 500)	PFT-DPW	15,682.47
	SITH-PFT	12,602.09

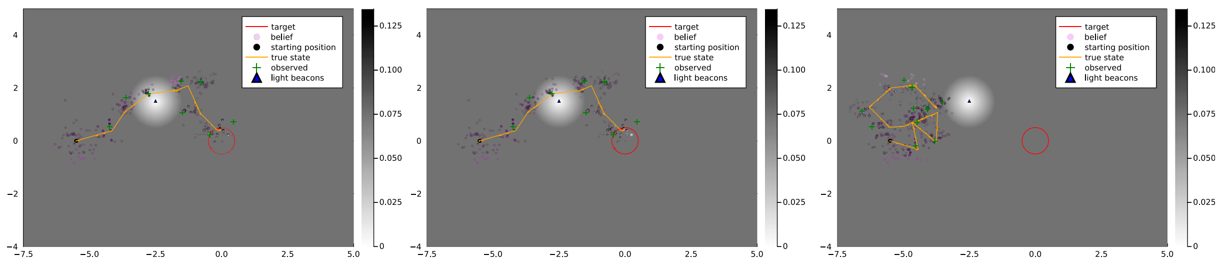
of particles in the tree built by IPFT will be much smaller than in the tree built by PFT-DPW. Do note that we cannot assure that the same K will be for each future history due to the exploratory nature of MCTS.

The speed of IPFT is linked with the rollout policy of MCTS. As we mentioned above, when the belief is represented by particles we know that asymptotically when the number of particles tends to infinity this representation converges to the theoretical belief for any given belief (Crisan and Doucet, 2002). Therefore, we shall take as many particles as possible for the belief representation. Given that the size of subsample m in IPFT does not change, this will increase the parameter K and therefore slowdown IPFT. Because when the new belief node is expanded in the belief tree there is always a rollout initiated, a more complex rollout policy will slowdown IPFT more, yet, this is ultimately the question of how big the parameter K is.

As we observe in Figure 17, IPFT is less accurate compared to PFT-DPW and SITH-PFT in spite of a much larger number of calls to SIMULATE routine compared to PFT-DPW and SITH-PFT. Clearly, better localization is closer to the beacons. In Figure 17(a) we see that more trajectories went to completely different from beacons directions as opposed to Figure 17(b) and (c) displaying identical results. From Figure 18(a) we conclude that in 10 from 15 trials the information reward obtained in execution of the optimal action returned by IPFT was inferior to the corresponding reward obtained by SITH-PFT and PFT-DPW. From Figure 18(b) we see that IPFT is slowest from the three algorithms while SITH-PFT (Alg. 7) is the fastest *in all trials*.

8.4. Discussion

Although the speedup was significant and steady for all simulations, we did not observe growth in speed-up with growth of number of belief particles in any simulation. This can be explained by the fact that increasing number of particles of the belief (n_x) changes also the bounds because the parameter n_x is present in the bounds as well. The



(a) SITH-PFT **(b)** PFT-DPW **(c)** IPFT

Figure 16. 2D continuous light dark. The agent starts from an initial unknown location and is given an initial belief. The goal is to get to location (0, 0) (circled in red) and execute the terminal action. Near the beacon (white light) the observations are less noisy. We consider multi-objective function, accounting for the distance to the goal and the differential entropy approximation (with the minus sign for reward notation). Executing the terminal action inside the red circle gives the agent a large positive reward but executing it outside it, will yield a large negative reward. (a) SITH-PFT. (b) PFT-DPW. (c) IPFT.

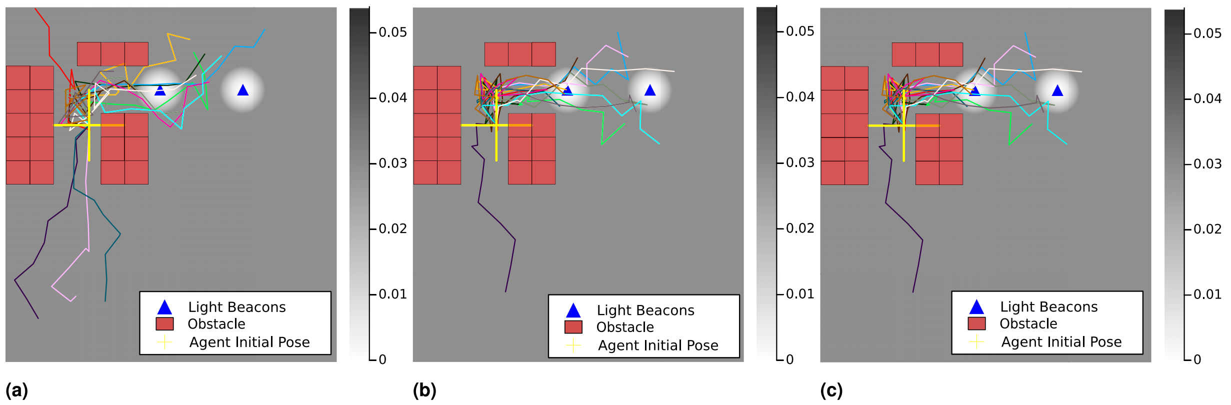


Figure 17. The plot shows 15 differently colored robot trajectories. Each such trajectory comprises 10 time steps. In each such step the robot performs re-planning and executes the best action selected by an appropriate BSP algorithm. The color of each trajectory matches planning with the same seed in each plot. The canvas color here is $\sigma_O = \sigma_T = 0.03$ as in equations (60) and (61), respectively. The parameters are $n_x = 300$, $m = 20$, number of calls to SIMULATE of IPFT is 4,500, the number of calls to SIMULATE of PFT-DPW and SITH-PFT is 300. In such a setting the constructed belief trees by these methods have the same number of total samples (see Section 8.3.5 for details). (a) Safe IPFT. (b) Safe SITH-PFT (Alg 7). (c) Safe PFT-DPW.

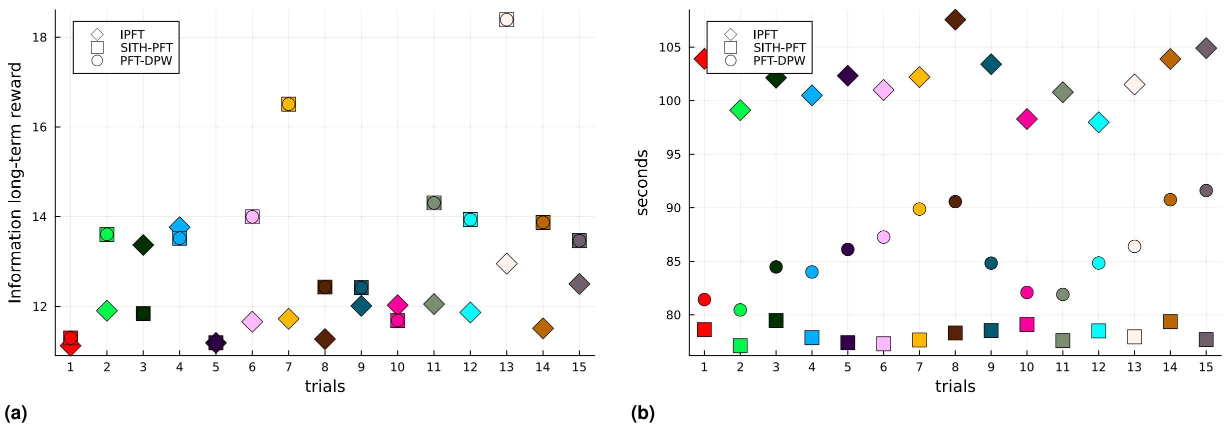


Figure 18. This plot is associated with Figure 17. Each color matches the corresponding trajectory in Figure 17. The parameters are $n_x = 300$, $m = 20$,and $K = 300/20 = 15$ number of calls to SIMULATE of IPFT is 4,500, the number of calls to SIMULATE of PFT-DPW and SITH-PFT is 300. In such a setting the constructed belief trees by these methods have the same number of total samples (see Section 8.3.5 for details). (a) Cumulative information reward as in (63) in the execution of the trajectory. Here, the SITH-PFT curve and the PFT-DPW curve overlap. This is because the rewards are identical since the same best action is calculated by SITH-PFT and PFT-DPW; (b) average planning times of 10 planning sessions in each trial.

limitation of our approach is that it leans on converging bounds, which are not trivial to derive and specific for a particular reward function. In addition, it requires slightly more caching than the baseline. Our simplification approach may still be ill-timed, since the resimplifications take an additional toll in terms of running time.

9. Conclusions

We contributed a rigorous provable theory of adaptive multilevel simplification that accelerates the solution of belief-dependent fully continuous POMDP. Our theory always identifies the same optimal action or policy as the unsimplified analog. Our theoretical approach receives as input adaptive bounds over the belief-dependent reward. Using the suggested theory and any bounds satisfying stated conditions we formulated three algorithms, considering a given belief tree and an anytime MCTS setting. We also contributed a specific simplification for non-parametric beliefs represented by weighted particles and derived novel bounds over a differential entropy estimator. These bounds are computationally cheaper than the latter. Our experiments demonstrate that our algorithms are paramount in terms of computation time while guaranteed to have the same performance as the baselines. In the setting of the given belief tree, we achieved a speedup up to 70%. In an anytime MCTS setting, our algorithm enjoyed the speedup of 20%.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research was supported by the Israel Science Foundation (ISF) and by a donation from the Zuckerman Fund to the Technion Artificial Intelligence Hub (Tech.AI).

ORCID iD

Andrey Zhitnikov https://orcid.org/0000-0002-6316-0998

Supplemental Material

Supplemental material for this article is available online.

References

Araya M, Buffet O, Thomas V, et al. (2010) A pomdp extension with belief-dependent rewards. In: *Advances in Neural Information Processing Systems (NIPS)*. Glasgow, Scotland: Curran Associates, Inc, 64–72.

Auer P, Cesa-Bianchi N and Fischer P (2002) Finite-time analysis of the multiarmed bandit problem. *Machine Learning* 47(2): 235–256.

Auger D, Couetoux A and Teytaud O (2013) Continuous upper confidence trees with polynomial exploration–consistency. In: Machine learning and knowledge discovery in databases: European conference, ECML PKDD 2013, proceedings, part I 13, Prague, Czech Republic, 23–27 September 2013, 194–209. Springer.

Barenboim M and Indelman V (2022) Adaptive information belief space planning. In: The 31st international joint conference on artificial intelligence and the 25th European conference on artificial intelligence (IJCAI-ECAI), Vienna, Austria, 23–29 July 2022.

Barenboim M and Indelman V (2023) Online pomdp planning with anytime deterministic guarantees. In: *Advances in Neural Information Processing Systems (NIPS)*. Glasgow, Scotland: Curran Associates, Inc.

Boers Y, Driessen H, Bagchi A, et al. (2010) Particle filter based entropy. In: 2010 13th international conference on information fusion, Edinburgh, UK, 26–29 July 2010, pp. 1–8. DOI: 10.1109/ICIF.2010.5712013.

Burgard W, Fox D and Thrun S (1997) Active mobile robot localization. In: 15th international joint conference on artificial intelligence (IJCAI 97), Nagoya, Japan, 23–29 August 1997, 1346–1352. Citeseer.

Crisan D and Doucet A (2002) A survey of convergence results on particle filtering for practitioners. *IEEE Transactions on Signal Processing* 50(3): 736–746.

Dressel L, Kochenderfer MJ, Barbulescu L, et al. (2017) Efficient decision-theoretic target localization. In: SF Smith (ed) Mausam and proceedings of the twenty-seventh international conference on automated planning and scheduling, ICAPS 2017, Pittsburgh, PA, 18–23 June 2017, 70–78. AAAI Press.

Egorov M, Sunberg ZN, Balaban E, et al. (2017) Pomdps. jl: a framework for sequential decision making under uncertainty. *Journal of Machine Learning Research* 18(1): 831–835.

Elimelech K and Indelman V (2022) Simplified decision making in the belief space using belief sparsification. *The International Journal of Robotics Research* 41(5): 470–496.

Farhi EI and Indelman V (2019) iX-BSP: belief space planning through incremental expectation. In: IEEE Intl. Conf. on robotics and automation (ICRA), Montreal, QC, 20–24 May 2019.

Farhi E and Indelman V (2021) ix-bsp: incremental belief space planning.ArXiv Preprint arXiv:2102.09539.

Fehr M, Buffet O, Thomas V, et al. (2018) rho-pomdps have lipschitz-continuous epsilon-optimal value functions. In: S Bengio, H Wallach, H Larochelle, et al. (eds) *Advances in Neural Information Processing Systems*. Glasgow, Scotland: Curran Associates, Inc, 6933–6943.

Fischer J and Tas OS (2020) Information particle filter tree: an online algorithm for pomdps with belief-based rewards on continuous domains. In: International conference on machine learning (ICML), Vienna, Austria, 12–18 July 2020.

Garg NP, Hsu D and Lee WS (2019) Despot-α: online pomdp planning with large state and observation spaces. In: Robotics: science and systems (RSS), Freiburg im Breisgau, Germany, 22–26 June 2019.

Hoerger M and Kurniawati H (2021) An on-line pomdp solver for continuous observation spaces. In: IEEE Intl. Conf. on robotics and automation (ICRA), Xi'an, China, 30 May–5 June 2021, 7643–7649. IEEE.

Hoerger M, Kurniawati H and Elfes A (2019) Multilevel monte-carlo for solving pomdps online. In: Proc. international symposium on robotics research (ISRR), Hanoi, Vietnam, 6–10 October 2019.

Hoerger M, Kurniawati H, Bandyopadhyay T, et al. (2020) Linearization in motion planning under uncertainty. In: *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*. Berlin, Germany: Springer, 272–287.

Hollinger GA and Sukhatme GS (2014) Sampling-based robotic information gathering algorithms. *The International Journal of Robotics Research* 33: 1271–1287.

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

Kearns M, Mansour Y and Ng AY (2002) A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine Learning* 49(2): 193–208.

Kitanov A and Indelman V (2024) Topological belief space planning for active slam with pairwise Gaussian potentials and performance guarantees. *The International Journal of Robotics Research* 43(1): 69–97. DOI: 10.1177/02783649231204898.

Kochenderfer M, Wheeler T and Wray K (2022) *Algorithms for Decision Making*. Cambridge, MA: MIT Press.

Kocsis L and Szepesvári C (2006) Bandit based Monte-Carlo planning. In: *Machine Learning: ECML*. Berlin, Germany: Springer, 282–293.

Kopitkov D and Indelman V (2017) No belief propagation required: belief space planning in high-dimensional state spaces via factor graphs, the matrix determinant lemma, and re-use of calculation. *The International Journal of Robotics Research* 36(10): 1088–1130.

Kopitkov D and Indelman V (2019) General-purpose incremental covariance update and efficient belief space planning via a factor-graph propagation action tree. *The International Journal of Robotics Research* 38(14): 1644–1673.

Kurniawati H, Hsu D and Lee WS (2008) SARSOP: efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: *Robotics: Science and Systems (RSS)*. Cambridge, MA: MIT Press.

Lev-Yehudi I, Barenboim M and Indelman V (2024) Simplifying complex observation models in continuous pomdp planning with probabilistic guarantees and practice. *Proceedings of the AAAI Conference on Artificial Intelligence* 38(18): 20176–20184.

Munos R (2014) *From Bandits to Monte-Carlo Tree Search: The Optimistic Principle Applied to Optimization and Planning*. Delft, the Netherlands: Now Publishers.

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

Pathak S, Thomas A and Indelman V (2018) A unified framework for data association aware robust belief space planning and perception. *Intl. J. of Robotics Research* 32(2–3): 287–315.

Platt R, Tedrake R, Kaelbling L, et al. (2010) Belief space planning assuming maximum likelihood observations. In: Robotics: science and systems (RSS), Zaragoza, Spain, 27–30 June 2010, pp. 587–593.

Shienman M and Indelman V (2022) Nonmyopic distilled data association belief space planning under budget constraints. In: *Robotics Research. ISRR 2022. Springer Proceedings in Advanced Robotics*. Cham, Switzerland: Springer.

Silver D and Veness J (2010) Monte-carlo planning in large pomdps. In: *Advances in Neural Information Processing Systems)*. Glasgow, Scotland: Curran Associates, Inc, pp. 2164–2172.

Smith T and Simmons R (2004) Heuristic search value iteration for pomdps. In: Conf. on uncertainty in artificial intelligence (UAI), Arlington, VA, 7 July 2004, pp. 520–527.

Spaan MT, Veiga TS and Lima PU (2015) Decision-theoretic planning under uncertainty with information rewards for active cooperative perception. *Autonomous Agents and Multi-Agent Systems* 29(6): 1157–1185.

Stachniss C, Grisetti G and Burgard W (2005) Information gain-based exploration using Rao-Blackwellized particle filters. In: Robotics: science and systems (RSS), Cambridge, MA, 8–11 June 2005, pp. 65–72.

Sunberg Z and Kochenderfer M (2018) Online algorithms for pomdps with continuous state, action, and observation spaces. In: Proceedings of the international conference on automated planning and scheduling, Delft, the Netherlands, 24–29 June 2018, 259–263.

Sutton RS and Barto AG (2018) *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Sztyglic O and Indelman V (2022) Speeding up online pomdp planning via simplification. In: IEEE/RSJ intl. conf. on intelligent robots and systems (IROS), Kyoto, Japan, 23–27 October 2022.

Thrun S, Burgard W and Fox D (2005) *Probabilistic Robotics*. Cambridge, MA: The MIT Press.

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

Walsh T, Goschin S and Littman M (2010) Integrating sample-based planning and model-based reinforcement learning. In: AAAI conf. on artificial intelligence, Atlanta, GA, 11–15 July 2010, 612–617.

Ye N, Somani A, Hsu D, et al. (2017) Despot: online pomdp planning with regularization. *Journal of Artificial Intelligence Research* 58: 231–266.

Zhitnikov A and Indelman V (2022a) Risk aware adaptive belief-dependent probabilistically constrained continuous pomdp planning. ArXiv Preprint arXiv:2209.02679.

Zhitnikov A and Indelman V (2022b) Simplified risk aware decision making with belief dependent rewards in partially observable domains. *Artificial Intelligence* 312: 103775.

Zhitnikov A and Indelman V (2024) Simplified continuous high dimensional belief space planning with adaptive probabilistic belief-dependent constraints. *IEEE Transactions on Robotics* 40: 1684–1705.

Appendix

Proof for Theorem 1

To shorten the notations we prove the theorem for value function under arbitrary policy. Note that by substituting the policy $\pi_{(\ell)+}$ by $\{\pi_\ell(b_\ell), \pi^*_{(\ell+1)+}\}$, where $a_\ell = \pi_\ell(b_\ell)$ we always can obtain action-value function. Without loosing generality assume the resimplification hits an arbitrary belief action node. The new upper bound will be

$$\overline{\widehat{V}}(b_\ell,\pi_{\ell+}) + \frac{1}{M}\left(\underbrace{\overline{\Delta}^{s+1}(b,a,b') - \overline{\Delta}^{s}(b,a,b')}_{\leq 0}\right) \leq \overline{\widehat{V}}(b_\ell,\pi_{\ell+}). \tag{67}$$

The new lower bound will be

$$\underline{\widehat{V}}\,(b_\ell,\pi_{\ell+}) - \frac{1}{M}\left(\underbrace{\underline{\Delta}^{s+1}(b,a,b') - \underline{\Delta}^{s}(b,a,b')}_{\leq 0}\right) \geq \underline{\widehat{V}}\,(b_\ell,\pi_{\ell+}), \tag{68}$$

where $M = n_z^d$ depending on the depth d of resimplified reward bound. Moreover if the inequalities involving increments are strict $\overline{\Delta}^{s}(b,a,b') > \overline{\Delta}^{s+1}(b,a,b')$ and $\underline{\Delta}^{s}(b,a) > \underline{\Delta}^{s+1}(b,a,b')$ also the retracting the bounds over Value function inequalities are strict. In case of MCTS, we have that $M = N(ha)/N(h')$, where history ha corresponds to b_ℓ and action a, and h' corresponds to b'.

Proof of Lemma 1

Recall that the bounds $\overline{\rho}, \underline{\rho}$ of belief nodes and "weakest link" rollout nodes are refined when the inequality (51) is encountered.

Assume in contradiction that the resimplification strategy does not promote any reward level and $G(ha) > 0$. This means that $G(ha)/d > 0$ and for all reward bounds the inequality $\gamma^{d-d'} \cdot (\overline{\rho} - \underline{\rho}) < 1/dG(ha)$. This is not possible since $G(ha)/d$ is the mean gap with respect to simulations of MCT and the depth of the belief tree, multiplied by the appropriate discount factor, over all the nodes that are the descendants to ha. See equation (33).

Proof of Lemma 2

Observe that

$$\overline{\text{UCB}}(ha) - \underline{\text{UCB}}\,(ha) = \overline{\widehat{Q}}(ha) - \underline{\widehat{Q}}\,(ha). \tag{69}$$

We already proved the desired for $\overline{\widehat{Q}}(ha), \underline{\widehat{Q}}\,(ha)$ in Theorem 1. Using the convergence $\underline{\widehat{Q}}\,(\cdot) = \widehat{Q}(\cdot) = \overline{\widehat{Q}}(\cdot)$ we obtain

$$\begin{aligned}&\underline{\widehat{Q}}\,(\cdot) + c \cdot \sqrt{\log(N(h))/N(ha)}\\ &\quad = \widehat{Q}(\cdot) + c \cdot \sqrt{\log(N(h))/N(ha)}\\ &\quad = \overline{\widehat{Q}}(\cdot) + c \cdot \sqrt{\log(N(h))/N(ha)}.\end{aligned} \tag{70}$$

The proof is completed.

Proof of Theorem 2

We provide proof by induction on the belief tree structure.

Base. Consider an initial given belief node b_0. No actions have been taken and no observations have been made. Thus, both the PFT tree and the SITH-PFT tree contain a single identical belief node, and the claim holds.

Induction hypothesis. Assume we are given two identical trees with n nodes, generated by PFT and SITH-PFT. The trees uphold the terms of **Definition 2**.

Induction step. Assume in contradiction that different nodes were added to the trees in the next simulation (expanding the belief tree by one belief node by definition). Thus, we got different trees.

Two scenarios are possible:

Case 1: The same action-observation sequence a_0, z_1, a_1, $z_2 \ldots a_m$ was chosen in both trees, but different nodes were added.

Case 2: Different action-observation sequences were chosen for both trees, and thus, we got different trees structure.

Since the Induction hypothesis holds, the last action a_m was taken from the same node denoted h' shared and identical to both trees. Next, the same observation model is sampled for a new observation, and a new belief node is added with a rollout emanating from it. The new belief nodes and the rollout are identical for both trees since both algorithms use the same randomization seed and observation and motion models. Case 2 must be true since we showed Case 1 is false. There are two possible scenarios such that different action-observation sequences were chosen:

Case 2.1. At some point in the actions-observations sequence, different observations z_i, z_i' were chosen.

Case 2.2. At some point in the actions-observations sequence, PFT chose action $a^\dagger$ while SITH-PFT chose a different action, $\tilde{a}$, or got stuck without picking any action.

Case 2.1 is not possible since if new observations were made, they are the same one by reasons contradicting Case 1. If we draw existing observations (choose some observation branch down the tree) the same observations are drawn since they are drawn with the same random seed and from the same observations "pool." It is the same "pool" since the Induction hypothesis holds. Case 2.2 must be true since we showed Case 2.1 is false, that is, when both algorithms are at the identical node denoted as h PFT chooses action $a^\dagger$, while SITH-PFT chooses a different action, $\tilde{a}$, or even got stuck without picking any action. Specifically, PFT chooses action $a^\dagger = \arg\max\limits_{a} \text{UCB}$ and SITH-PFT's candidate action is $\tilde{a} = \arg\max\limits_{a \in \mathcal{A}} \underline{\text{UCB}}\,(ha)$. Three different scenarios are possible:

Case 2.2.1. The $\overline{\text{UCB}}, \underline{\text{UCB}}$ bounds over $h\tilde{a}$ were tight enough and $\tilde{a}$ was chosen such that $a^{\dagger} \neq \tilde{a}$.

Case 2.2.2. SITH-PFT is stuck in an infinite loop. It can happen if the $\overline{\text{UCB}}, \underline{\text{UCB}}$ bounds over $h\tilde{a}$, and at least one of its sibling nodes ha, are not tight enough. However, all tree nodes are at the maximal simplification level. Hence, resimplification is triggered over and over without it changing anything.

Case 2.2.1 is not possible as the bounds are analytical (always true) and converge to the actual reward $(\underline{\text{UCB}} = \text{UCB} = \overline{\text{UCB}})$ for the maximal simplification level. Case 2.2.2 is not possible. If the bounds are not close enough to make a decision, resimplification is triggered. Each time some ha node - sibling to $h\tilde{a}$ and maybe even $h\tilde{a}$ itself is chosen in *SelectBest* to over-go resimplification. According to Lemmas 1 and 2, after some finite number of iterations for all of the sibling ha nodes (including $h\tilde{a}$) it holds $\underline{\text{UCB}}(ha) = \text{UCB}(ha) = \overline{\text{UCB}}(ha)$ and some action can be picked. If different actions have identical values we choose one by the same rule UCB picks actions with identical values (e.g. lower index/random). Since Case 2.2.2 is false, after some finite number of resimplification iterations, SITH-PFT will stop with bounds sufficient enough to make a decision; as Case 2.2.1 is false it holds that $a^{\dagger} = \tilde{a}$. Thus we get a contradiction and the proof is complete.

Proof of Theorem 3

Since same tree is built according to Theorem 2, the only modification is the final criteria at the end of the planning session at the root of the tree: $a^{*} = \arg\max Q(ha)$. Note we can set the exploration constant of UCB to $c = 0$ and we get that UCB is just the Q function. Thus if the bounds are not tight enough at the root to decide on an action, resimplification will be repeatedly called until SITH-PFT can make a decision. The action will be identical to the one chosen by UCB at PFT from similar arguments in the proof of Theorem 2. Note that additional final criteria for action selection could be introduced, but it would not matter as tree consistency is kept according to Theorem 2 and the bounds converge to the immediate rewards and Q estimations.

Proof for Theorem 4

Let us first prove that $u + \widehat{\mathcal{H}} \geq 0$. It holds

$$u + \widehat{\mathcal{H}} = \sum_{i \notin A^s_{k+1}} w^i_{k+1} \cdot \log\left[m \cdot \mathbb{P}_O\left(z_{k+1} | x^i_{k+1}\right)\right] + \sum_{i \in A^s_{k+1}} w^i_{k+1} \cdot \log\left[\mathbb{P}_O\left(z_{k+1} | x^i_{k+1}\right) \sum_{j=1}^{n_x} \mathbb{P}_T\left(x^i_{k+1} | x^j_k, a_k\right) w^j_k\right] - \sum_{i=1}^{n_x} w^i_{k+1} \cdot \log\left[\mathbb{P}_O\left(z_{k+1} | x^i_{k+1}\right) \sum_{j=1}^{n_x} \mathbb{P}_T\left(x^i_{k+1} | x^j_k, a_k\right) w^j_k\right] = \tag{71}$$

The equation (71) equals to

$$\sum_{i \notin A^s_{k+1}} w^i_{k+1} \cdot \log\left[m \cdot \mathbb{P}_O\left(z_{k+1} | x^i_{k+1}\right)\right] - \sum_{i \notin A^s_{k+1}} w^i_{k+1} \cdot \log\left[\mathbb{P}_O\left(z_{k+1} | x^i_{k+1}\right) \sum_{j=1}^{n_x} \mathbb{P}_T\left(x^i_{k+1} | x^j_k, a_k\right) w^j_k\right].$$

Fix arbitrary index $i \notin A^s_{k+1}$. The log is monotonically increasing function so it is left to prove that

$$m\mathbb{P}_O\left(z_{k+1} | x^i_{k+1}\right) \geq \mathbb{P}_O\left(z_{k+1} | x^i_{k+1}\right) \sum_{j=1}^{n_x} \mathbb{P}_T\left(x^i_{k+1} | x^j_k, a_k\right) w^j_k.$$

If $\mathbb{P}_O\left(z_{k+1} | x^i_{k+1}\right) = 0$, we finished. Assume $\mathbb{P}_O\left(z_{k+1} | x^i_{k+1}\right) \neq 0$. Recalling the definition $\max\limits_{\substack{x' \\ x,a}} \mathbb{P}_T(x'|x,a)$, it holds that

$$\mathbb{P}_O\left(z_{k+1} | x^i_{k+1}\right) \sum_{j=1}^{n_x} \max_{\substack{x_{k+1} \\ x_k, a_k}} \mathbb{P}_T(x_{k+1} | x_k, a_k) w^j_k \geq \mathbb{P}_O\left(z_{k+1} | x^i_{k+1}\right) \sum_{j=1}^{n_x} \mathbb{P}_T\left(x^i_{k+1} | x^j_k, a_k\right) w^j_k. \tag{72}$$

We reached the desired result. Now let us show the second part $\ell + \widehat{\mathcal{H}} \leq 0$. Observe, that

$$0 \geq \ell + \widehat{\mathcal{H}} = \sum_{i=1}^{n_x} w^i_{k+1} \log\left[\mathbb{P}_O\left(z_{k+1} | x^i_{k+1}\right) \sum_{j \in A^s_k} \mathbb{P}_T\left(x^i_{k+1} | x^j_k, a_k\right) w^j_k\right] - \sum_{i=1}^{n_x} w^i_{k+1} \log\left[\mathbb{P}_O\left(z_{k+1} | x^i_{k+1}\right) \sum_{j=1}^{n_x} \mathbb{P}_T\left(x^i_{k+1} | x^j_k, a_k\right) w^j_k\right] \tag{73}$$

Select arbitrary index i. We shall prove that

$$\log\left[\mathbb{P}_O\left(z_{k+1} | x^i_{k+1}\right) \sum_{j \in A^s_k} \mathbb{P}_T\left(x^i_{k+1} | x^j_k, a_k\right) w^j_k\right] - \log\left[\mathbb{P}_O\left(z_{k+1} | x^i_{k+1}\right) \sum_{j=1}^{n_x} \mathbb{P}_T\left(x^i_{k+1} | x^j_k, a_k\right) w^j_k\right] \leq 0$$

Again use that log is monotonically increasing and assume that $\mathbb{P}_O(z_{k+1}|x^i_{k+1}) \neq 0$. We have that

$$\begin{aligned}&\sum_{j\in A^s_k}\mathbb{P}_T(x^i_{k+1}|x^j_k,a_k)w^j_k - \sum_{j=1}^{n_x}\mathbb{P}_T(x^i_{k+1}|x^j_k,a_k)w^j_k \\ &= -\sum_{j\notin A^s_k}\mathbb{P}_T(x^i_{k+1}|x^j_k,a_k)w^j_k \leq 0\end{aligned}. \tag{74}$$

Proof for Theorem 5

We first prove that

$$\overline{\Delta}^s(b,a,b') \geq \overline{\Delta}^{s+1}(b,a,b') \geq 0. \tag{75}$$

Recall that from the previous proof equation (71)

$$\overline{\Delta}^s(b,a,b') = \sum_{i\notin A^s_{k+1}} w^i_{k+1}\log\left[m\cdot\mathbb{P}_O(z_{k+1}|x^i_{k+1})\right]$$
$$-\sum_{i\notin A^s_{k+1}} w^i_{k+1}\log\left[\mathbb{P}_O(z_{k+1}|x^i_{k+1})\sum_{j=1}^{n_x}\mathbb{P}_T(x^i_{k+1}|x^j_k,a_k)w^j_k\right]. \tag{76}$$

Suppose we promote the simplification level. Without loss of generality assume that $A^{s+1}_{k+1} = A^s_{k+1}\cup\{q\}$. From the above we conclude that $q \notin A^s_{k+1}$

$$\begin{aligned}&\overline{\Delta}^{s+1}(b,a,b') = \overline{\Delta}^s(b,a,b') \\ &-w^q_{k+1}(\log\left[m\cdot\mathbb{P}_O(z_{k+1}|x^q_{k+1})\right] \\ &-\log\left[\mathbb{P}_O(z_{k+1}|x^q_{k+1})\sum_{j=1}^{n_x}\mathbb{P}_T(x^q_{k+1}|x^j_k,a_k)w^j_k\right]\Bigg).\end{aligned} \tag{77}$$

It is left to prove that

$$\begin{aligned}&m\cdot\mathbb{P}_O(z_{k+1}|x^q_{k+1}) \\ &\geq \mathbb{P}_O(z_{k+1}|x^q_{k+1})\sum_{j=1}^{n_x}\mathbb{P}_T(x^q_{k+1}|x^j_k,a_k)w^j_k\end{aligned}. \tag{78}$$

We already done that in previous theorem. Now we prove the second part

$$\underline{\Delta}^s(b,a,b') \geq \underline{\Delta}^{s+1}(b,a,b') \geq 0. \tag{79}$$

The next equation is the minus equation (74)

$$\begin{aligned}&\underline{\Delta}^s(b,a,b') \\ &= \sum_{i=1}^{n_x} w^i_{k+1}\log\left[\mathbb{P}_O(z_{k+1}|x^i_{k+1})\sum_{j=1}^{n_x}\mathbb{P}_T(x^i_{k+1}|x^j_k,a_k)w^j_k\right] \\ &-\sum_{i=1}^{n_x} w^i_{k+1}\log\left[\mathbb{P}_O(z_{k+1}|x^i_{k+1})\sum_{j\in A^s_k}\mathbb{P}_T(x^i_{k+1}|x^j_k,a_k)w^j_k\right].\end{aligned} \tag{80}$$

Assume again without loosing generality that $A^{s+1}_k = A^s_k\cup\{q\}$. In that case

$$\underline{\Delta}^s(b,a,b') - \underline{\Delta}^{s+1}(b,a,b') = \tag{81}$$

$$-\sum_{i=1}^{n_x} w^i_{k+1}\log\left[\mathbb{P}_O(z_{k+1}|x^i_{k+1})\sum_{j\in A^s_k}\mathbb{P}_T(x^i_{k+1}|x^j_k,a_k)w^j_k\right] \tag{82}$$

$$+\sum_{i=1}^{n_x} w^i_{k+1}\log\left[\mathbb{P}_O(z_{k+1}|x^i_{k+1})\sum_{j\in A^{s+1}_k}\mathbb{P}_T(x^i_{k+1}|x^j_k,a_k)w^j_k\right]. \tag{83}$$

Select arbitrary index i. We got back to end to previous theorem. Note that by definition the bounds are convergent since we are using all the particles. To see it explicitly suppose that $\{i\notin A^s_{k+1}\} = \emptyset$ and $\{i\notin A^s_k\} = \emptyset$. We have that

$$\overline{\Delta}^s(b,a,b') = \underline{\Delta}^s(b,a,b') = 0. \tag{84}$$

This concludes the proof.

Bounds time complexity analysis

We turn to analyze the time complexity of our method using the chosen bounds (57) and (58). We assume the significant bottleneck is querying the motion $\mathbb{P}_T(x'|x,a)$ and observation $\mathbb{P}_O(z|x)$ models. Assume the belief is approximated by a set of n_x weighted particles,

$$b = \{x^i, w^i\}_{i=1}^{n_x}. \tag{85}$$

Consider the Boers et al. (2010) differential entropy approximation for belief at time $k+1$,

$$\hat{\mathcal{H}}(b_k,a_k,z_{k+1},b_{k+1}) \triangleq \underbrace{\log\left[\sum_{i=1}\mathbb{P}_Z(z_{k+1}|x^i_{k+1})w^i_k\right]}_{(a)} + \tag{86}$$

$$\underbrace{\sum_{i=1}^{n_x} w^i_{k+1}\cdot\log\left[\mathbb{P}_O(z_{k+1}|x^i_{k+1})\sum_{j=1}^{n_x}\mathbb{P}_T(x^i_{k+1}|x^j_k,a_k)w^j_k\right]}_{(b)}. \tag{87}$$

Denote the time to query the observation and motion models a single time as t_{obs} and t_{mot}, respectively. It is clear from (85), (86) (term a) and, (87) (term b) that:

$$\forall b \text{ as in } (86)\Theta\left(\hat{\mathcal{H}}(b)\right) = \Theta\left(n_x\cdot t_{obs} + n_x^2\cdot t_{mot}\right). \tag{88}$$

Since we share calculation between the bounds, the bounds' time complexity, for some level of simplification s, is:

$$\Theta(\ell^s + u^s) = \Theta\left(n_x\cdot t_{obs} + n^s_x\cdot n_x\cdot t_{mot}\right), \tag{89}$$

where n_x^s is the size of the particles subset that is currently used for the bounds calculations, for example $n_x^s = |A^s|$ (A^s is as in (57) and (58)) and ℓ^s, u^s denotes the immediate upper and lower bound using simplification level s. Further, we remind the simplification levels are discrete, finite, and satisfy

$$s \in \{1, 2, \ldots, n_{\max}\}, \ell^{s=n_{\max}} = -\widehat{\mathcal{H}} = u^{s=n_{\max}}. \tag{90}$$

Now, assume we wish to tighten ℓ^s, u^s and move from simplification level s to $s+1$. Since the bounds are updated incrementally (as introduced by Sztyglic and Indelman (2022)), when moving from simplification level s to $s+1$ the only additional data we are missing are the new values of the observation and motion models for the newly added particles. Thus, we get that the time complexity of moving from one simplification level to another is

$$\Theta\left(\ell^s + u^s \to \ell^{s+1} + u^{s+1}\right) = \Theta\left(\left(n_x^{s+1} - n_x^s\right) \cdot n_x \cdot t_{mot}\right), \tag{91}$$

where Θ ($\ell^s + u^s \to \ell^{s+1} + u^{s+1}$) denotes the time complexity of updating the bounds from one simplification level to the following one. Note the first term from (89), $n_x{\cdot}t_{obs}$, is not present in (91). This term has nothing to do with simplification level s and it is calculated linearly over all particles n_x. Thus, it is calculated once at the beginning (initial/lowest simplification level).

We can now deduce using (89) and (91)

$$\begin{aligned}&\Theta\left(\ell^{s+1} + u^{s+1}\right) = \\ &\Theta\left(\ell^s + u^s\right) + \Theta\left(\ell^s + u^s \to \ell^{s+1} + u^{s+1}\right).\end{aligned} \tag{92}$$

Finally, using (88)–(92), we come to the conclusion that if at the end of a planning session, a node's b simplification level was $1 \le s \le n_{\max}$ than the time complexity saved for that node is

$$\Theta\left(\left(n_x - n_x^s\right) \cdot n_x \cdot t_{mot}\right). \tag{93}$$

This makes perfect sense since if we had to resimplify all the way to the maximal level we get $s = n_{\max} \Rightarrow n_x^{s=n_{\max}} = n_x$ and by substituting $n_x^s = n_x$ in (93) we saved no time at all.

To conclude, the total speedup of the algorithm is dependent on how many belief nodes' bounds were not resimplified to the maximal level. The more nodes we had at the end of a planning session with lower simplification levels, the more speedup we get according to (93).