IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Previous Knowledge Utilization In Online Anytime Belief Space Planning

Michael Novitsky, Moran Barenboim and Vadim Indelman

***Abstract*— Online planning under uncertainty remains a critical challenge in robotics and autonomous systems. While tree search techniques are commonly employed to construct partial future trajectories within computational constraints, most existing methods discard information from previous planning sessions considering continuous spaces. This study presents a novel, computationally efficient approach that leverages historical planning data in current decision-making processes. We provide theoretical foundations for our information reuse strategy and introduce an algorithm based on Monte Carlo Tree Search (MCTS) that implements this approach. Experimental results demonstrate that our method significantly reduces computation time while maintaining high performance levels. Our findings suggest that integrating historical planning information can substantially improve the efficiency of online decision-making in uncertain environments, paving the way for more responsive and adaptive autonomous systems.**

I. INTRODUCTION

Autonomous agents often operate under uncertainty due to sensor noise and incomplete information, maintaining a belief (probability distribution) over possible states instead of direct access to the true environment state. Partially Observable Markov Decision Processes (POMDPs) provide a framework for such settings, but solving them optimally is computationally intractable (PSPACE-complete) [1], mainly due to the curse of history, curse of dimensionality, and continuous state, action and observation spaces common in real-world applications.

Recent advancements have introduced online algorithms [2] [3] [4] that find approximate solutions to POMDPs. These algorithms operate within limited budget constraints, such as restricted time, and employ a sampling-based approach to construct partial trees and search for the optimal action that maximizes the expected cumulative reward. By sampling a subset of the belief space, these algorithms effectively address both the curse of history and the curse of dimensionality, which are key obstacles in solving POMDPs.

In POMDPs, the reward function of a belief node is typically formulated as the expected reward over states. However, this formulation may be insufficient for certain problems, such as information gathering and active sensing. In such cases, the problem is commonly addressed as Belief Space Planning (BSP) or ρ-POMDP [5], where the reward is defined over the belief itself. Information-theoretic measures, such as information gain and differential entropy, are commonly used to quantify uncertainty in the decision-making process [6]. However, exact calculation of information-theoretic rewards becomes intractable for general distributions, as it requires integrating over all possible states. To address this challenge, approximation methods such as kernel density estimation (KDE) and particle filter estimation [7] have been proposed in the literature. Nonetheless, these methods still incur significant computational expenses, with computation complexity scaling quadratically with the number of samples. As reward calculation is performed for each node in the tree, it becomes the primary source of computational complexity in online planning algorithms.

The main objective of this paper is to improve planning efficiency within a non-parametric setting, continuous state, action and observation spaces, and general reward functions. To address these challenges, we contribute a novel approach that leverages the Multiple Importance Sampling framework [8] to tackle the problem of reusing information from previous planning sessions. We demonstrate how our method can be integrated with Monte Carlo Tree Search (MCTS) to create a novel online algorithm called Incremental Reuse Particle Filter Tree (IR-PFT). We evaluate our algorithm in an online planning setting, demonstrating reduced planning time without performance loss.

The code for this paper is available at https://github.com/miken1990/ir-pft.

II. RELATED WORK

Solving POMDPs is challenging, but methods like POMCP [2] and POMCPOW [10] have advanced the field. POMCP extends UCT [9] to partial observability by sampling a state from the current belief, propagating it through a search tree, and recording statistics with a Multi-Armed Bandit for action selection in discrete spaces. POMCPOW adapts this approach to continuous action and observation spaces using progressive widening and weighted particle sets with an observation likelihood model.

In PFT-DPW [10] the authors adopt particle filter formulation for belief update and each belief is represented with a constant number of samples. $\rho-$POMCP [11] propagates a set of particles in each simulation using a particle filter and adds it to existing particles in visited nodes. Frequently visited nodes achieve better representation, with convergence proven asymptotically for continuous, bounded ρ.

Manuscript received: June 3, 2025; Accepted: August 13, 2025. This paper was recommended for publication by Editor Markus Vincze upon evaluation of the Associate Editor and Reviewers' comments. Michael Novitsky and Moran Barenboim are with the Technion Autonomous Systems Program (TASP), Technion - Israel Institute of Technology, Haifa 32000, Israel, (miken1990; moranbar@campus.technion.ac.il). Vadim Indelman is with the Department of Aerospace Engineering and with the Department of Data and Decision Science, Technion - Israel Institute of Technology, Haifa 32000, Israel. vadim.indelman@technion.ac.il. This work was partially supported by US NSF/US-Israel BSF.

IPFT [6] extends PFT [10] by incorporating a reward function defined as a linear combination of differential entropy and expected state reward. In each simulation, a particle set is sampled from the root belief and propagated through the tree. Entropy estimates are averaged across particle sets at each belief node to estimate differential entropy.

LABECOP [4] is designed for continuous observation spaces. At each belief node b, a state s is sampled, an action a selected via modified UCB, and an observation o is sampled. The states for b, a are reweighted using o to improve $\hat{Q}(b, a)$.

SITH-BSP [12], [13] and AI-FSSS [14] make use of simplification of the reward function calculation and observation space sampling accordingly, while preserving action consistency. The extension of ρ-POMDP to $\mathbb{P}\rho-$POMDP in [15] incorporates stochastic bounds on the return, while also quantifying the effects of applying simplifications.

DESPOT [3] and subsequent works [16], [17], [18] propose algorithms that use determinized random sampling to build the search tree incrementally, with recent work addressing large observation spaces [17]. The use of α-vectors in [16]–[18] relies on piecewise-linear approximations and thus only applies to POMDPs with belief-dependent rewards that admit such representations.

Previous methods start each planning session from scratch, while iX-BSP [19], [20] proposes reuse but assumes an open loop setting and does not address non-parametric beliefs. Sample reuse improves data efficiency in RL by leveraging past experiences [29]. Our approach instead uses online planning from a given belief, assuming known transition and observation models. Adaptive Belief Tree (ABT) [21] is an online POMDP solver for dynamic continuous-state domains that revises its particle-based belief tree upon model changes. It uses a generative transition model and assumes state-based rewards; the original formulation did not address continuous acton and observation spaces, nor belief-dependent rewards. In this work, we address continuous state, action, and observation spaces with general belief-dependent rewards, a non-parametric framework, and a closed-loop setting.

III. Background

A. POMDP

POMDP is defined as a 7-tuple $(S, A, O, \mathbb{P}_T, \mathbb{P}_O, r, b_0)$, where S, A, and O denote the state, action, and observation spaces. Since the exact state of the world is not known and only observations are available, the agent maintains a probability density function (PDF) over the state space—also referred to as the belief. The belief at time k is given by $b_k(s_k) = \mathbb{P}(s_k|H_k)$, where $H_k = (b_0, a_0, o_1, .., o_k) = \{b_0, o_{1:k}, a_{1:k-1}\}$ denotes the history up to time k, consisting of the initial belief, the sequence of observations, and the sequence of actions taken. $\mathbb{P}_T$ is the state transition density function, $\mathbb{P}_O$ is the observation density function, $r(b, a, b')$ represents the reward function based on the current belief b, action a, and the resulting belief b', and b_0 denotes the initial belief over states. It is assumed that the belief is sufficient statistics for the decision making and a Bayesian update is used to update the belief recursively:

$$b_{k+1}(s_{k+1}) = \eta \mathbb{P}_O(o_{k+1}|s_{k+1}) \int_{s_k} \mathbb{P}_T(s_{k+1}|s_k, a_k) b_k(s_k) ds_k. \tag{1}$$

where η is a normalization term. A policy $\pi \in \Pi$ is a mapping from belief space to action space $\pi : b \to a$. We define the value function V^π for any policy π and horizon d as

$$V^\pi(b_k) = \mathop{\mathbb{E}}_{b_{k+1:k+d}}[G_k|b_k, \pi]. \tag{2}$$

where $\pi \triangleq \pi_{k:k+d-1}$ represents a sequence of policies for horizon d and $G_k = \sum_{i=k}^{k+d-1} r(b_i, \pi_i(b_i), b_{i+1})$ is the return. Similarly, we define the action value function Q^π as

$$Q^\pi(b_k, a) = \mathbb{E}_{b_{k+1}}[r(b_k, a, b_{k+1}) + V^\pi(b_{k+1})]. \tag{3}$$

B. Non-Parametric Setting

We adopt a non-parametric setting, representing belief distributions with state particles. Using a particle filter [22], we update posterior estimates with each new observation. The theoretical belief b_k is approximated by m particles $\{s_k^i\}_{i=1}^m$ assuming resampling, each with uniform weight $\frac{1}{m}$

$$\hat{b}_k(s) = \frac{1}{m} \sum_{i=1}^{m} \delta(s - s_k^i). \tag{4}$$

Given resampled belief $\hat{b}_k$, action a_k, and propagated belief $\hat{b}_{k+1}^-$, calculating $\mathbb{P}(\hat{b}_{k+1}^-|\hat{b}_k, a_k)$ involves determining all the matchings between the states in $\hat{b}_k$ and those in $\hat{b}_{k+1}^-$ which is $\sharp P$-complete [23]. We assume, similar to [24], that the beliefs are not permutation invariant, meaning particle beliefs with different particle orders are not considered identical. This assumption simplifies the derivation of the propagated belief likelihood. Consequently, we can express $\hat{b}_k$ as $\{s_k^i, \frac{1}{m}\}_{i=1}^m$ and $\hat{b}_{k+1}^-$ as $\{s_{k+1}^{-i}, \frac{1}{m}\}_{i=1}^m$

$$\mathbb{P}(\hat{b}_{k+1}^-|\hat{b}_k, a_k) = \frac{1}{m} \prod_{i=1}^{m} \mathbb{P}(s_{k+1}^{-i}|s_k^i, a_k). \tag{5}$$

In the rest of the paper we assume a non-parametric setting and for the ease of notation we remove the hat sign $\hat{}$ and the state argument (s) from all beliefs.

C. Importance Sampling

Importance sampling estimates properties of a target PDF $p(x)$ by sampling from a proposal PDF $q(x)$, assigning weights to adjust each sample's contribution according to $p(x)$

$$\hat{\mathbb{E}}_p^{IS}[f(x)] = \frac{1}{N} \sum_{i=1}^{N} w^i \cdot f(x^i), \;\; w^i = \frac{p(x^i)}{q(x^i)}, \;\; x^i \sim q. \tag{6}$$

The PDF q must satisfy $q(x^i) = 0 \Rightarrow p(x^i) = 0$. With M proposal PDFs $\{q_m\}_{m=1}^M$, Multiple Importance Sampling

formulation [8] can be used:

$$\hat{\mathbb{E}}_p^{MIS}[f(x)] = \sum_{m=1}^{M} \frac{1}{n_m} \sum_{i=1}^{n_m} w^m(x^{i,m}) \frac{p(x^{i,m})}{q_m(x^{i,m})} f(x^{i,m}). \tag{7}$$

Here, n_m denotes the number of samples that originate from PDF q_m, $x^{i,m}$ denotes the ith sample that originates from PDF q_m and the weights w^m must satisfy

$$\begin{aligned} q_m(x^{i,m}) = 0 \Rightarrow w^m(x) f(x) p(x) = 0. \\ f(x^{i,m}) \neq 0 \Rightarrow \sum_{m=1}^{M} w^m(x^{i,m}) = 1, \end{aligned} \tag{8}$$

ensuring that at least one PDF $q_m(x) \neq 0$ for every valid x. Under these conditions, the estimator in (7) remains unbiased [8]. We assume that the weights w^m are determined using the balance heuristic which bounds the variance of the estimator [8] and in this case the MIS estimator is

$$\hat{\mathbb{E}}_p^{MIS}[f(x)] = \sum_{m=1}^{M} \sum_{i=1}^{n_m} \frac{p(x^{i,m})}{\sum_{j=1}^{M} n_j \cdot q_j(x^{i,m})} f(x^{i,m}). \tag{9}$$

For more details on MIS, see [8].

D. PFT-DPW

The PFT-DPW algorithm [10] is based on the UCT algorithm [9] and expands its application to a continuous state, action and observation setting. It utilizes Monte-Carlo simulations to progressively construct a policy tree for the belief MDP [10]. At every belief node b_k and action a_k it sets up visitation counts $N(b_k, a_k)$ and $N(b_k)$, where $N(b_k) = \sum_{a_k} N(b_k, a_k)$ and action-value function is calculated incrementally

$$Q(b_k, a_k) \triangleq \frac{1}{N} \sum_{i=1}^{N} G_k^i, \tag{10}$$

by averaging accumulated reward upon initiating from node b_k and taking action a_k within the tree. Notably, $Q(b_k, a_k)$ (10) is not equal to $Q^\pi(b_k, a_k)$ (3) as the policy varies across different simulations within the tree, causing the distribution of the trajectories to be non-stationary, hence the absence of the π superscript. The particle filter generates a propagate belief b_{k+1}^- and posterior belief b_{k+1} from b_k and a_k, sampling observation o_{k+1} and computing reward r

$$b_{k+1}, b_{k+1}^-, o_{k+1}, r \leftarrow G_{PF(m)}(b_k, a_k). \tag{11}$$

To handle continuous spaces, Double Progressive Widening limits a node's children to kN^α, where N is the node visit count, and k and α are hyperparameters [10].

IV. APPROACH

Our contributions are threefold: (1) an efficient incremental update method for the Multiple Importance Sampling (MIS) estimator, enabling action-value estimation from prior and newly arriving data; (2) the application of MIS for experience-based value estimation using expert-provided demonstrations without planning; and (3) an MCTS-inspired online algorithm that speeds up computations by reusing data from previous planning sessions.

A. Incremental Multiple Importance Sampling Update

In our setting, samples arrive incrementally in batches. A straightforward computation of (9) would necessitate a complexity of $O(M^2 \cdot n_{avg})$, where M denotes the number of different distributions and n_{avg} denotes the average sample count across all distributions. We develop an efficient way to update the estimator (9) incrementally in the theorem below.

Theorem 1: Consider an MIS estimator (9) with M different distributions and n_m samples for each distribution $q_m \in \{q_1, ..., q_M\}$. Given a batch of L I.I.D samples from distribution $q_{m'}$, where $q_{m'}$ could be one of the existing distributions or a new, previously unseen distribution, $\hat{\mathbb{E}}_p^{MIS}[f(x)]$ (9) can be efficiently updated with a computational complexity of $O(M \cdot n_{avg} + M \cdot L)$ and memory complexity $O(M \cdot n_{avg})$. Proof. see Appendix A.

B. Experience-Based Value Function Estimation

We assume that we have access to a dataset

$$D \triangleq \{\tau^i, G_{k_i}^i\}_{i=1}^{|D|} \tag{12}$$

of trajectories executed by an agent that followed an uknown policy π. Each trajectory is defined as the sequence

$$\begin{aligned} \tau^i \triangleq (b_{k_i}^i, a_{k_i}^i) \rightarrow (b_{k_i+1}^{-i}, o_{k_i+1}^i, b_{k_i+1}^i, a_{k_i+1}^i) \rightarrow ... \\ \rightarrow (b_{k_i+d}^{-i}, o_{k_i+d}^i, b_{k_i+d}^i), \end{aligned} \tag{13}$$

where k_i represents the starting time index and is used to differentiate between different steps in trajectory τ^i and d is the horizon length. We assume that the agent applied a particle filter with resampling at each step of the trajectory. The return G^i associated with trajectory τ^i is defined as the accumulated reward,

$$G_{k_i}^i \triangleq \sum_{j=0}^{d-1} r(b_{k_i+j}^i, a_{k_i+j}^i, b_{k_i+j+1}^i). \tag{14}$$

In this section, we evaluate $V^\pi(b_k)$ for the current belief b_k using only the dataset D (12), without planning. Such estimation is important in data-expensive domains like autonomous vehicles [25] and robotic manipulation tasks [26]. In the next section, we will expand our methodology to include planning.

Reusing trajectories where the initial belief is set to b_k presents no challenge - we can aggregate all trajectories that begin with belief b_k and action a_k and assuming we have N such trajectories, we define a sample-based estimator

$$\hat{Q}^\pi(b_k, a_k) \triangleq \frac{1}{N} \sum_{i=1}^{N} G_k^i. \tag{15}$$

In continuous spaces, the chance of sampling the same belief twice is zero, so each trajectory in D (12) begins from a belief different from b_k. To be able to reuse trajectories from (12), we discard the initial belief and action of the trajectory, instead linking the current belief and action to the remainder of the trajectory. Formally, given a trajectory $\tau^i \in D$, $\tau^i = (b_{k_i}^i, a_{k_i}^i) \rightarrow \tau_{suffix}^i$ where

$$\begin{aligned} \tau_{suffix}^i \triangleq (b_{k_i+1}^{-i}, o_{k_i+1}^i, b_{k_i+1}^i, a_{k_i+1}^i) \rightarrow ... \\ \rightarrow (b_{k_i+d}^{-i}, o_{k_i+d}^i, b_{k_i+d}^i). \end{aligned} \tag{16}$$

and the current belief b_k and action a_k, we construct a new trajectory τ_i' (see Figure 1),

$$\tau_i' \triangleq (b_k, a_k) \rightarrow \tau^i_{suffix}. \tag{17}$$

To estimate $Q^\pi(b_k, a_k)$ using the information within tra-

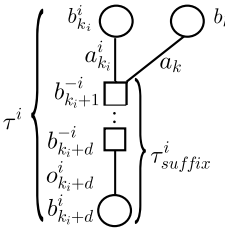


Fig. 1: τ^i is a trajectory that was executed by an agent that followed policy π, τ^i_{suffix} is the part that we reuse from τ^i for the current belief b_k and action a_k.

jectory τ^i, two adjustments are required. Firstly, we need to modify the initial term in the return G^i to be equal to $r(b_k, a_k, b^i_{k_i+1})$, recognizing that $b_k \neq b^i_{k_i}$ and $a_k \neq a^i_{k_i}$. Consequently, we define the return of trajectory

$$\tilde{G}^i_k \triangleq G^i_{k_i} - r(b^i_{k_i}, a^i_{k_i}, b^i_{k_i+1}) + r(b_k, a_k, b^i_{k_i+1}). \tag{18}$$

Secondly, we need to adjust the weight of $\tilde{G}^i$ due to the disparity between $\mathbb{P}(\tau^i_{suffix}|b^i_{k_i}, a^i_{k_i}, \pi)$ and $\mathbb{P}(\tau^i_{suffix}|b_k, a_k, \pi)$, which is acheived through importance sampling. The distribution $\mathbb{P}(\cdot|b^i_{k_i}, a^i_{k_i}, \pi)$ of τ^i_{suffix} (16) is determined by the initial belief $b^i_{k_i}$ and action $a^i_{k_i}$. Given N_{IS} partial trajectories sampled from the same distribution $\mathbb{P}(\cdot|b^i_{k_i}, a^i_{k_i}, \pi)$, we define an Importance Sampling estimator

$$\hat{Q}^\pi_{IS}(b_k, a_k) \triangleq \frac{1}{N_{IS}} \sum_{i=1}^{N_{IS}} w^i \cdot \tilde{G}^i_k. \tag{19}$$

where $w^i \triangleq \frac{\mathbb{P}(\tau^i_{suffix}|b_k, a_k, \pi)}{\mathbb{P}(\tau^i_{suffix}|b^i_{k_i}, a^i_{k_i}, \pi)}$. As a result of our approach to constructing reusable trajectories as described in (17), we can efficiently calculate the weights w^i utilizing the theorem presented below.

Theorem 2: Given belief node b_k, action a_k and trajectory $\tau^i = (b^i_{k_i}, a^i_{k_i}) \rightarrow \tau^i_{suffix}$ where τ^i_{suffix} is defined in (16), the following equality holds:

$$\frac{\mathbb{P}(\tau^i_{suffix}|b_k, a_k, \pi)}{\mathbb{P}(\tau^i_{suffix}|b^i_{k_i}, a^i_{k_i}, \pi)} = \frac{\mathbb{P}(b^{-i}_{k_i+1}|b_k, a_k)}{\mathbb{P}(b^{-i}_{k_i+1}|b^i_{k_i}, a^i_{k_i})}. \tag{20}$$

Proof. see appendix B.

Let M be the number of unique partial trajectory PDFs $\{\mathbb{P}(\cdot|b^m_{k_m}, a^m_{k_m}, \pi)\}_{m=1}^M$ each defined by its initial belief $b^m_{k_m}$ and action $a^m_{k_m}$, and let n_m denote the sample count for each. Consequently, the dataset D (12) can be reformulated as follows:

$$D \triangleq \{\tau^{l,m}, G^{l,m}_k\}^{M,n_m}_{m=1,l=1}. \tag{21}$$

Using this formulation, we define a multiple importance sampling estimator assuming the balance heuristic (9),

$$\hat{Q}^\pi_{MIS}(b_k, a_k) \triangleq \sum_{m=1}^{M} \sum_{l=1}^{n_m} \frac{\mathbb{P}(\tau^{l,m}_{suffix}|b_k, a_k, \pi)\tilde{G}^{l,m}_k}{\sum_{j=1}^M n_j \cdot \mathbb{P}(\tau^{l,m}_{suffix}|b^j_{k_j}, a^j_{k_j}, \pi)}. \tag{22}$$

where $\tau^{l,m}_{suffix}$ represents the lth partial trajectory that was sampled from the distribution $\mathbb{P}(\cdot|b^m_{k_m}, a^m_{k_m}, \pi)$ and $\tilde{G}^{l,m}_k$ is the adjusted accumulated reward (18).

Using Theorem 2, we can re-write the MIS estimator (22)

$$\hat{Q}^\pi_{MIS}(b_k, a_k) \triangleq \sum_{m=1}^{M} \sum_{l=1}^{n_m} \frac{\mathbb{P}(b^{-l,m}_{k_m+1}|b_k, a_k)}{\sum_{j=1}^M n_j \cdot \mathbb{P}(b^{-l,m}_{k_m+1}|b^j_{k_j}, a^j_{k_j})} \cdot \tilde{G}^{l,m}_k. \tag{23}$$

Since each element in the second sum of (23) corresponds to a propagated belief, which might appear more than once, we can rewrite the sum in a more compact form. Specifically, we group the terms based on unique propagated beliefs and account for their multiplicity:

$$\hat{Q}^\pi_{MIS}(b_k, a_k) \triangleq \sum_{m=1}^{M} \sum_{l=1}^{|C(b_{k_m}, a_{k_m})|} W(b^{-l,m}_{k_m+1}) \cdot \sum_{y=1}^{N(b^{-l,m}_{k_m+1})} \tilde{G}^{m,l,y}_k. \tag{24}$$

The weights $W(b^{-l,m}_{k_m+1})$ are defined by:

$$W(b^{-l,m}_{k_m+1}) = \frac{\mathbb{P}(b^{-l,m}_{k_m+1}|b_k, a_k)}{\sum_{j=1}^M n_j \cdot \mathbb{P}(b^{-l,m}_{k_m+1}|b^j_{k_j}, a^j_{k_j})} \tag{25}$$

$C(b_{k_m}, a_{k_m})$ denotes the set of reused propagated belief children associated with b_{k_m} and a_{k_m}. The term $N(b^{-l,m}_{k_m+1})$ represents the visitation count of $b^{-l,m}_{k_m+1}$, indicating the number of trajectories that pass through the propagated belief $b^{-l,m}_{k_m+1}$ and $\tilde{G}^{m,l,y}_k$ is the return of the y-th trajectory passing through $b^{-l,m}_{k_m+1}$. Note that $b^{-l,m}_{k_m+1}$ in (24) represents unique propagated beliefs, which differs from (23), where it denotes the propagated belief associated with a single trajectory. Figure 2 illustrates the estimator from (24). For the current belief b_k and action a_k, three prior trajectories are incorporated: two from $(b^i_{k_i}, a^i_{k_i})$ and one from $(b^j_{k_j}, a^j_{k_j})$. Light green edges show the connections between (b_k, a_k) and the reused nodes for estimating $\hat{Q}^\pi_{MIS}(b_k, a_k)$. Further

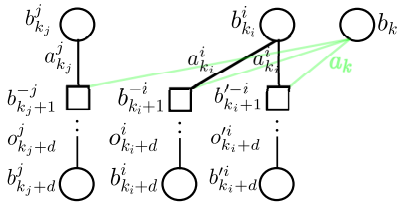


Fig. 2: Illustration of reuse of three trajectories.

in this work, we consider a framework where the dataset D (21) expands over time with trajectory samples from an agent following policy π. Theorem 1 is used to efficiently update the estimator (23) with new samples.

For the estimator in (24) to be unbiased, the conditions in (8) must hold. Specifically, for every propagated belief b^-_{k+1} that can be generated from b_k, a_k, there must exist at least one pair $b^j_{k_j}, a^j_{k_j}$ from which b^-_{k+1} can also be generated.

Our framework differs from standard off-policy evaluation. Traditional importance sampling estimates $V^{\pi_t}(b_k)$ using

trajectories from b_k under a behavior policy π_b [27]. In contrast, we estimate $V^{\pi}(b_k)$ using trajectories from *different* beliefs in the dataset D (12). To oue knowledge, such a setting has not been previously explored for action-value estimation in POMDPs.

C. Our POMDP Planning Algorithm: IR-PFT

Thus far, we used trajectories from a single policy π to estimate $Q^{\pi}(b_k, a_k)$. In this section, we introduce an anytime POMDP planning algorithm that accelerates planning by reusing trajectories from dataset D, which contains data from prior planning sessions.

We name our algorithm Incremental Reuse Particle Filter Tree (IR-PFT). Instead of calculating $Q(b_k, a_k)$ from scratch in each planning session, we use previous experience to speed up the calculations.

We adopt the same approach as in Section IV-B to reuse trajectories, with three key modifications: first, the propagated belief nodes from the previous planning session in dataset D have a shorter planning horizon. We extend the horizon of these nodes before reusing them; second, the policy varies across different simulations (as in standard MCTS), resulting in a non-stationary distribution of reused trajectories in D; and third, we integrate the planning and generation of new trajectories with the reuse of previous trajectories within an anytime MCTS setting.

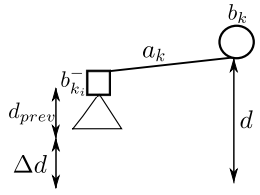


Fig. 3: Illustration of horizon gap.

Figure 3 visually illustrates the horizon alignment process, where a propagated belief node $b_{k_i}^-$ with horizon d_{prev} must be extended by Δd to match the current horizon d. We analyze the complexity of the correction of belief nodes from previous planning sessions in case of using the MCTS [28] algorithm in Corollary 1.

Corollary 1: Given an MCTS tree T with horizon d_{prev}, number of simulations m and N nodes, extending its horizon by Δd will require adding at most $m \cdot \Delta d$ nodes and reward calculations.

The proof is straightforward: after m simulations, the MCTS tree contains at most m leaves and we need to extend each leaf by Δd and for each new node we calculate a reward.

We now define the estimator

$$\hat{Q}_{MIS}(b_k, a_k) \triangleq \sum_{m=1}^{M} \sum_{l=1}^{|C(b_{k_m}, a_{k_m})|} W(b_{k_m+1}^{-l,m}) \cdot \sum_{y=1}^{N(b_{k_m+1}^{-l,m})} \bar{G}_k^{m,l,y}, \tag{26}$$

where the weights $W(b_{k_m+1}^{-l,m})$ are defined in (25). $\bar{G}_k^{m,l,y}$ is the extended return defined as

$$\bar{G}_k^{m,l,y} = \tilde{G}_k^{m,l,y} + \sum_{i=k+d_{prev}^{l,m}}^{k+d_{prev}^{l,m}+\Delta d-1} r(b_i, \pi_{rollout}(b_i), b_{i+1}). \tag{27}$$

$d_{prev}^{l,m}$ is the horizon of propagated belief $b_{k_m+1}^{-l,m}$, $\bar{G}_k^{m,l,y}$ shares the same values in the summation as the return $\tilde{G}_k^{m,l,y}$ (18), but it also includes additional terms from the extended trajectory due to the horizon extension using the rollout policy $\pi_{rollout}$. Therefore, only the rewards for these additional terms need to be computed when extending the horizon, while all shared terms ($\tilde{G}_k^{m,l,y}$) can be reused.

Since the tree policy in MCTS differs across simulations, the resulting trajectory distribution becomes non-stationary. As a consequence, the update defined in (26) functions in a heuristic manner, and its convergence properties remain unproven. We leave a rigorous investigation of this aspect for future work.

After extending the horizon of a reused propagated belief node b^- and reusing its action-value function, the counter $N(b_k, a_k)$ is incremented by the visitation count $N(b^-)$ using the relation $N(b_k, a_k) = \sum_{b_k^- \in C(b_k, a_k)} N(b_k^-)$. This approach accelerates our algorithm for a given number of simulations, offering a speedup over PFT-DPW [10].

To summarize, here is the high-level logical flow of our algorithm: At each iteration, we either reuse a propagated belief node by extending its horizon by Δd, as illustrated in Figure 3, and compute the extended return for the subtree rooted at the reused node b^-, or create a new node. Subsequently, the Multiple Importance Sampling (MIS) estimator (26) is employed to evaluate the action-value function. To avoid the computational expense of a naive calculation, we leverage Theorem 1 to perform efficient incremental updates of the estimator in (26).

D. Algorithm Description

The complete algorithm is outlined across multiple methods - Algs. 1, 2, 3 and 4. Alg. 1 illustrates a general planning loop wherein the agent iteratively plans and executes actions until the problem is solved. After each planning session, reuse candidates are updated based on the preceding planning tree. The main algorithm is detailed in Alg. 2 with key

Algorithm 1: General Planning Loop

```
1: Procedure: SOLVE(b, D)
2: while ProblemNotSolved() do
3:     a ← Plan(b, D)
4:     o ← ReceiveObservation(b, a)
5:     b', b'^-, r ← G_{PF(m)}(b, a, o)
6:     UpdateReuseCandidates(a, D, b, b')
7:     b ← b'
8: end while
```

modifications compared to the PFT-DPW algorithm [10] highlighted in red. The ActionProgWiden method (line 5)

is implemented following the same approach as described in [10]. ShouldReuse method (line 7) evaluates three conditions: current node b is the root, the balance between reused and new nodes, and the availability of reuse candidates. The second criterion is important because, while acquiring estimates from prior partial trajectories is runtime-efficient, generating new trajectory samples from the correct distribution is essential. Currently, reuse is applied only at the root node, where it yields the greatest computational savings by avoiding many reward computations due to its shallow depth. While extending reuse to nodes at other depth levels is feasible, it falls outside the scope of this work.

The GetReuseCandidate method (line 8) selects a candidate propagated belief b'^{-} from dataset D using a distance function f_D (line 2). For example, f_D may be defined as $||b^{-} - b^{-}_{MLE}||_2^2$, where b^{-}_{MLE}—the maximum likelihood propagated belief for given b and a—is computed via (5) in $O(m)$ time. Since f_D scans the entire dataset, it must be efficient, and reusing nodes with high visitation counts further reduces runtime.

The FillHorizonPropagated method (line 9), addresses discrepancies in horizon lengths when reusing nodes from the previous planning sessions. Algorithm 4 performs recursive traversal of the subtree defined by propagated belief b'^{-} and extends its depth by d using the rollout policy.

At lines 10 and 11, we increment counters, where $N(b'^{-})$ represents the count of trajectories passing through reuse candidate propagated belief node b'^{-}. At line 12, we increment the PLAN procedure counter by $N(b'^{-})$.

At lines 13 and 29 we utilize (26) to update $Q(b, a)$, leveraging efficiency through the application of Theorem (1). At line 14 we store the propagated belief b'^{-}.

Lines 17–19 execute when reuse is skipped: they initialize a new propagated belief via the particle filter [22], save the propagated and posterior beliefs, and perform a rollout.

At lines 23 and 24 we sample uniformly both propagated and posterior beliefs.

In Algorithm 3, UpdateReuseCandidates selects nodes with more than n_{min} visits to maximize speedup.

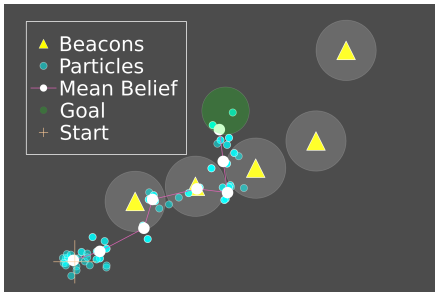
Algorithm 2: IR-PFT

1: **Procedure**: PLAN(b, D)
2: $i = 0$
3: **while** $i < n$ **do**
4: $Simulate(b, d_{max}, D)$
5: **end while**
6: $a = argmax_a\{Q(b, a)\}$
7: return a

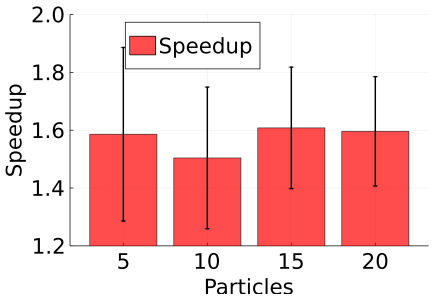
1: **Procedure**: SIMULATE(b, d, D)
2: **if** $d = 0$ **then**
3: return 0
4: **end if**
5: $a \leftarrow$ ActionProgWiden(b)
6: **if** $|C(ba)| \leq k_o N(ba)^{\alpha_o}$ **then**
7: **if** $ShouldReuse(b, a, D)$ **then**
8: $b'^{-} \leftarrow GetReuseCandidate(b, a, D)$
9: $FillHorizonPropagated(b'^{-}, d - d_{b^{-}})$
10: $N(b) \leftarrow N(b) + N(b'^{-})$
11: $N(ba) \leftarrow N(ba) + N(b'^{-})$
12: $i \leftarrow i + N(b'^{-})$ {update simulation counter}
13: $Q(ba) \leftarrow MISUpdate$ defined by eq. (26)
14: $C(b, a) \leftarrow C(b, a) \cup \{(b'^{-})\}$
15: return $Q(ba)$
16: **else**
17: $b', b'^{-}, r \leftarrow G_{PF(m)}(b, a)$
18: $C(b, a) \leftarrow C(b, a) \cup \{(b'^{-})\}$
19: $C(b'^{-}) \leftarrow C(b'^{-}) \cup \{(b', r)\}$
20: $total \leftarrow r + \gamma ROLLOUT(b', d - 1)$
21: **end if**
22: **else**
23: $b'^{-} \leftarrow$ sample uniformly from $C(ba)$
24: $b', r \leftarrow$ sample uniformly from $C(b'^{-})$
25: $total \leftarrow r + \gamma Simulate(b', d - 1, T)$
26: **end if**
27: $N(b) \leftarrow N(b) + 1$
28: $N(ba) \leftarrow N(ba) + 1$
29: $Q(ba) \leftarrow MISUpdate$ defined by eq. (26)
30: return $total$

V. RESULTS

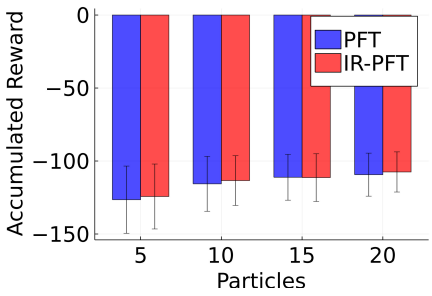
We assess the performance of the IR-PFT algorithm by comparing it to the PFT-DPW algorithm [10]. Our evaluation focuses on two main aspects: runtime and accumulated reward, with statistics measured for each. In all experiments, the solvers were limited to 1000 iterations for each planning phase. All experiments were conducted on the standard 2D Light Dark benchmark, where the agent must reach the goal while minimizing localization uncertainty. Beacons define illuminated regions, where observation noise decreases linearly outside them, reaching a minimum at the specified white radius, while areas beyond are considered dark. Refer to the illustration in Figure 4a. The reward function is defined as a weighted sum of the average distance to goal and differential entropy estimator which is calculated using [7]. Each algorithm was evaluated using different quantities of particles—specifically, $5, 10, 15$, and 20, while maintaining a constant horizon length of $d = 10$. In the following results reuse was done according to ShouldReuse method (line 7) as detailed in the previous section. We compared the performance of IR-PFT (with and without reuse) to PFT-DPW and present the findings as a speedup chart in Figure 4b, plotted against the number of particles. IR-PFT consistently outperformed PFT-DPW, with the speedup saturating at approximately 1.5—a factor primarily attributed to the savings in reward computation, which is the most computationally intensive part of the algorithm. We compare the accumulated rewards of IR-PFT with and without reuse (Figure 4c). The results show negligible differences, indicating that our method improves runtime without compromising planning performance.



(a) Illustration of the continuous light dark 2d environment



(b) Speedup



(c) Accumulated reward comparison

Fig. 4: Light dark experiments comparing PFT and IR-PFT.

Algorithm 3: Reuse Functions

Procedure: UPDATEREUSECANDIDATES(a,D,b_k,b^{real}_{k+1})
ReuseDict $dict \leftarrow \{\}$
for $b^-_{k+1} \in C(b_k, a)$ **do**
  for $b_{k+1} \in C(b^-_{k+1})$ **do**
    for $a' \in Actions(b_{k+1})$ **do**
      for $b^-_{k+2} \in C(b_{k+1}, a')$ **do**
        if $N(b^-_{k+2}) > n_{min}$ **then**
          $D.append(b^-_{k+2})$
        end if
      end for
    end for
  end for
end for

Procedure: SHOULDREUSE(b,a,D)
if not b.IsRoot() **then**
  return $false$
end if
if $NumReused(b, a) > \frac{|C(b,a)|}{2}$ **then**
  return $false$
end if
$candidates \leftarrow D.GetReuseCandidatesDict()$
return not($candidates.empty()$)

Procedure: GETREUSECANDIDATE(b,a,D)
$b^- \leftarrow argmin_{b^-}\{f_D(b^-, b, a)\}$
return b^-

Algorithm 4: Fill Horizon Gap

Procedure: FILLHORIZONPROPAGATED(b^-, d)
$Q_{new}(b^-) \leftarrow 0$
for $b' \in C(b^-)$ **do**
  $Q_{new}(b^-) \leftarrow$
  $Q_{new}(b^-) + FillHorizonPosterior(b')$
end for
return $Q_{new}(b^-)$

Procedure: FILLHORIZONPOSTERIOR(b, d)
if $IsLeaf(b)$ **then**
  $a \leftarrow DefaultPolicy(b)$
  $b', b'^-, r \leftarrow G_{PF(m)}(b, a)$
  $N(b, a) \leftarrow 1$
  $N(b'^-) \leftarrow 1$
  $Q(b'^-) \leftarrow r$
  $Q(b, a) \leftarrow r$
  return r
end if
$Q(b) \leftarrow 0$
for $a \in Actions(b)$ **do**
  for $b'^- \in C(b, a)$ **do**
    $Q(b) \leftarrow$
    $Q(b) + FillHorizonPropagated(b'^-, d-1)$
  end for
end for
$Q(b) \leftarrow \frac{Q(b)}{N(b)}$
return $Q(b)$

VI. CONCLUSIONS

In this paper, we have proposed a general framework which allows to reuse prior information during the current planning session. We derived theoretical justification for reuse via Multiple Importance Sampling and introduced a new MCTS-like algorithm, IR-PFT which reuses information from previous planning session and allows to speed up calculations in current planning session. In order to evaluate IR-PFT algorithm, we conducted an empirical performance study. Specifically, we compared the performance of our approach with and without the reuse of prior information. We measured various performance metrics, including computation time and the accumulated reward. Our results clearly indicate a speed-up in the planning process when prior information is leveraged. Importantly, despite the accelerated computations, our approach maintains the same level of performance as the traditional planning approach without reuse. Incorporating prior information significantly boosts planning efficiency, delivering time savings while maintaining high-quality results. These findings underscore the effectiveness and potential of the proposed approach. Scaling to higher dimensions is challenging, as similar belief states become exponentially rarer. However, imposing structural assumptions

may still allow reuse, as shown in [20].

APPENDIX

A. Proof of Theorem 1

Consider an MIS estimator (9) with M different distributions and n_m samples for each distribution $q_m \in \{q_1, ..., q_M\}$. Given a batch of L I.I.D. samples from distribution $q_{m'}$ which may be an existing or new distribution,

$$\hat{\mathbb{E}}_p^{MIS}[f(x)] = \sum_{m=1}^{M}\sum_{i=1}^{n_m} \frac{p(x_{i,m})}{\sum_{j=1}^{M} n_j \cdot q_j(x_{i,m})} f(x_{i,m}),$$

(9) can be efficiently updated with a computational complexity of $O(M \cdot n_{avg} + M \cdot L)$ and memory complexity $O(M \cdot n_{avg})$.

For every distribution q_m we have the term $\sum_{i=1}^{n_m} \frac{p(x_{i,m})}{\sum_{j=1}^{M} n_j \cdot q_j(x_{i,m})} f(x_{i,m})$.

In case $m \neq m'$:

$\sum_{j=1}^{M} n_j \cdot q_j(x_{i,m}) \leftarrow \sum_{j=1}^{M} n_j \cdot q_j(x_{i,m}) + L \cdot q_{m'}(x_{i,m})$ - $O(1)$ complexity. We have n_m samples and M distributions so the complexity of this update is $O(M \cdot n_m)$.

In case $m = m'$:

For existing samples $\sum_{j=1}^{M} n_j \cdot q_j(x_{i,m}) \leftarrow \sum_{j=1}^{M} n_j \cdot q_j(x_{i,m}) + L \cdot q_{m'}(x_{i,m})$ - $O(1)$ complexity. We have n_m samples existing samples so in total $O(n_m)$ complexity.

For each new sample we need to calculate $\frac{p(x_{i,m})}{\sum_{j=1}^{M} n_j \cdot q_j(x_{i,m})} f(x_{i,m})$ - $O(M)$ complexity. We have L new samples so in total $O(L \cdot M)$ complexity. The total complexity of the update is $O(M \cdot n_{avg} + M \cdot L)$.

B. Proof of Theorem 2

$$\frac{\mathbb{P}(\tau^i_{suffix}|b_k, a_k, \pi)}{\mathbb{P}(\tau^i_{suffix}|b^i_{k_i}, a^i_{k_i}, \pi)} = \frac{\mathbb{P}(b^{-i}_{k_i+1}, ..., b^i_{k_i+L}|b_k, a_k, \pi)}{\mathbb{P}(b^{-i}_{k_i+1}, ..., b^i_{k_i+L}|b^i_{k_i}, a^i_{k_i}, \pi)}. \tag{28}$$

Applying chain rule yields,

$$\frac{\mathbb{P}(b^{-i}_{k_i+1}|b_k, a_k)}{\mathbb{P}(b^{-i}_{k_i+1}|b^i_{k_i}, a^i_{k_i})} \cdot \frac{\mathbb{P}(o^i_{k_i+1}, \ldots, b^i_{k_i+L}|b^{-i}_{k_i+1}, \pi)}{\cancel{\mathbb{P}(o^i_{k_i+1}, \ldots, b^i_{k_i+L}|b^{-i}_{k_i+1}, \pi)}} = \frac{\mathbb{P}(b^{-i}_{k_i+1}|b_k, a_k)}{\mathbb{P}(b^{-i}_{k_i+1}|b^i_{k_i}, a^i_{k_i})}. \tag{29}$$

REFERENCES

[1] C. Papadimitriou and J. Tsitsiklis, "The complexity of Markov decision processes," *Mathematics of operations research*, vol. 12, no. 3, pp. 441–450, 1987.

[2] D. Silver and J. Veness, "Monte-carlo planning in large pomdps," in *Advances in Neural Information Processing Systems (NIPS)*, 2010, pp. 2164–2172.

[3] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "Despot: Online pomdp planning with regularization." in *NIPS*, vol. 13, 2013, pp. 1772–1780.

[4] M. Hoerger and H. Kurniawati, "An on-line pomdp solver for continuous observation spaces," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7643–7649.

[5] M. Araya-López, O. Buffet, V. Thomas, and F. Charpillet, "A pomdp extension with belief-dependent rewards." in *NIPS*, 2010, pp. 64–72.

[6] J. Fischer and O. S. Tas, "Information particle filter tree: An online algorithm for pomdps with belief-based rewards on continuous domains," in *Intl. Conf. on Machine Learning (ICML)*, Vienna, Austria, 2020.

[7] Y. Boers, H. Driessen, A. Bagchi, and P. Mandal, "Particle filter based entropy," in *2010 13th International Conference on Information Fusion*, 2010, pp. 1–8.

[8] E. Veach and L. J. Guibas, "Optimally combining sampling techniques for monte carlo rendering," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM, 1995, pp. 419–428.

[9] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*. Springer, 2006, pp. 282–293.

[10] Z. Sunberg and M. Kochenderfer, "Online algorithms for pomdps with continuous state, action, and observation spaces," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 28, no. 1, 2018.

[11] V. Thomas, G. Hutin, and O. Buffet, "Monte carlo information-oriented planning," *arXiv preprint arXiv:2103.11345*, 2021.

[12] O. Sztyglic and V. Indelman, "Speeding up online pomdp planning via simplification," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2022.

[13] A. Zhitnikov, O. Sztyglic, and V. Indelman, "No compromise in solution quality: Speeding up belief-dependent continuous pomdps via adaptive multilevel simplification," *Intl. J. of Robotics Research*, 2024.

[14] M. Barenboim and V. Indelman, "Adaptive information belief space planning," in *the 31st International Joint Conference on Artificial Intelligence and the 25th European Conference on Artificial Intelligence (IJCAI-ECAI)*, July 2022.

[15] A. Zhitnikov and V. Indelman, "Simplified risk aware decision making with belief dependent rewards in partially observable domains," *Artificial Intelligence, Special Issue on "Risk-Aware Autonomous Systems: Theory and Practice"*, 2022.

[16] N. Ye, A. Somani, D. Hsu, and W. S. Lee, "Despot: Online pomdp planning with regularization," *JAIR*, vol. 58, pp. 231–266, 2017.

[17] N. P. Garg, D. Hsu, and W. S. Lee, "Despot-α: Online pomdp planning with large state and observation spaces," in *Robotics: Science and Systems (RSS)*, 2019.

[18] P. Cai, Y. Luo, D. Hsu, and W. S. Lee, "Hyp-despot: A hybrid parallel algorithm for online planning under uncertainty," *Intl. J. of Robotics Research*, vol. 40, no. 2-3, pp. 558–573, 2021.

[19] E. I. Farhi and V. Indelman, "ix-bsp: Belief space planning through incremental expectation," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2019.

[20] E. Farhi and V. Indelman, "ix-bsp: Incremental belief space planning," *https://arxiv.org/abs/2102.09539*, 2021.

[21] H. Kurniawati and V. Yadav, "An online POMDP solver for uncertainty planning in dynamic environment," in *Robotics Research*, Springer, 2016, pp. 611–629.

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

[23] L. G. Valiant, "The complexity of computing the permanent," *Theoretical Computer Science*, vol. 8, no. 2, pp. 189–201, Apr. 1979.

[24] M. H. Lim, T. J. Becker, M. J. Kochenderfer, C. J. Tomlin, and Z. N. Sunberg, "Optimality guarantees for particle belief approximation of pomdps," *Journal of Artificial Intelligence Research*, vol. 77, pp. 1591–1636, 2023.

[25] K. T. e. a. H. Caesar, J. Kabzan, "Nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles," in *CVPR ADP3 workshop*, 2021.

[26] A. Mandlekar, J. Booher, M. Spero, A. Tung, A. Gupta, Y. Zhu, A. Garg, S. Savarese, and L. Fei-Fei, "Scaling robot supervision to hundreds of hours with roboturk: Robotic manipulation dataset through human reasoning and dexterity," 2019.

[27] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[28] A. Couetoux and Teytaud, "Continuous upper confidence trees with polynomial exploration - consistency," in *European conference on machine learning*. Springer, 2013.

[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, in *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.