

# Anytime Density-Adaptive Planning in $\rho$ POMDPs

Ron Benchetrit, Idan Lev-Yehudi, Tamir Shazman, and Vadim Indelman

**Abstract**—Partially Observable Markov Decision Processes (POMDPs) provide a principled framework for decision-making under uncertainty in domains such as autonomous driving and robotic exploration.  $\rho$ POMDPs replace the state-dependent reward with a belief-dependent one, enabling explicit uncertainty reasoning and active information gathering. Existing online  $\rho$ POMDP solvers mostly fall into two families with complementary limitations: their per-node beliefs are either refined over time but blind to belief density, or, at best, density-adaptive but frozen on instantiation. We propose ADAPT (Anytime Density-Adaptive Particle Tree), an online tree-search algorithm whose per-node particle belief is simultaneously anytime and density-adaptive, and we show that a single tolerance parameter bounds its per-node reward error. To make information-theoretic rewards affordable under continual belief refinement, we derive an incremental update scheme that lowers the per-particle cost of common entropy estimators by an order of magnitude. ADAPT outperforms state-of-the-art solvers on both POMDP and  $\rho$ POMDP benchmarks.

**Index Terms**—Planning under Uncertainty, Autonomous Agents

## I. INTRODUCTION

Autonomous agents must make informed decisions under uncertainty, stemming from both the environment’s dynamics and the agent’s perception. Sources of uncertainty include sensor noise, modeling approximations, and stochastic action outcomes. The Partially Observable Markov Decision Process (POMDP) is a general framework for modeling such problems, with applications spanning autonomous driving, robotic exploration, and active perception [1]. This generality comes at a cost: exact optimal solutions are PSPACE-complete [2], so practical solvers resort to approximation.

Online tree-search algorithms approximate POMDP solutions by building a search tree from the current belief at each time-step, concentrating computation on reachable beliefs and, via an exploration–exploitation tradeoff, expanding unevenly toward the most promising branches [3]. Anytime variants are useful under real-time constraints, computing a usable policy whenever the agent must act and improving it as further computation permits.

Non-parametric POMDP solvers represent the belief with a set of sampled states, called *particles*, valued for their flexibility in representing arbitrary beliefs over large and continuous state spaces. We call a per-node belief *anytime*

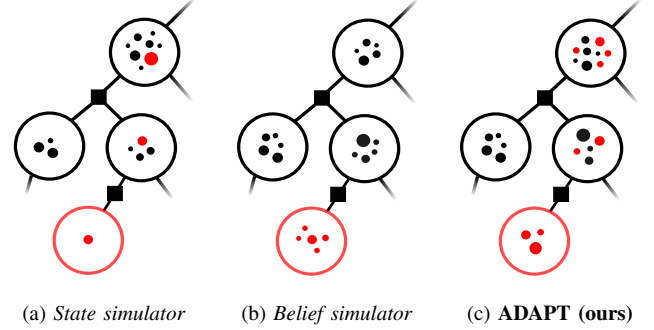


Fig. 1. Comparison of belief-tree construction. New particles and nodes are marked in red. A *state simulator* (a) adds particles along the visited trajectory, refining beliefs over time; a *belief simulator* (b) sizes each belief once on instantiation and keeps it fixed; **ADAPT** (c) refines over time like (a) yet sizes each belief to its local density like (b).

if its particle count grows with the computation it receives, and *density-adaptive* if the algorithm sets that count from the complexity of the belief itself, accounting for its effective support and dimensionality. Viewed through how the per-node belief evolves, particle-based solvers cluster into two families (Figure 1). *State simulators* [4]–[7] build each node’s belief implicitly, accumulating the particles of simulated trajectories that pass through it, so the most promising—hence most visited—nodes hold the richest beliefs: the count is a function of visitation, and the belief is *anytime* but *blind to belief density*. *Belief simulators* [6], [8] instead solve a particle-belief MDP, sizing each node’s belief once at instantiation without later refinement, so beliefs in promising and non-promising regions receive the same computational resources. Most algorithms in this family [6], [9]–[11] are not density-adaptive either: the count is a parameter, tuned offline and identical for every node in the tree. The exception is AdaOPS [8], which sets that one-time count by KLD-sampling [12], the standard density-adaptive rule, and is therefore density-adaptive but still not anytime. No existing solver is both *anytime* and *density-adaptive*.

Belief-dependent rewards explicitly model tasks that require reasoning over uncertainty. They have been applied to active localization [13], active SLAM [14], reward shaping in sparse-reward problems [9], [15], and as heuristics for guiding tree search [16]. A  $\rho$ POMDP [17] is a POMDP with a belief-dependent reward, also known as Belief Space Planning (BSP) [18]–[20].  $\rho$ POMDPs often use information-theoretic rewards such as entropy or information gain. Estimating these from a particle belief is costly: estimators such as that of [21] and kernel-density approaches scale quadratically with the number of particles. Under such costly rewards, the gap above is exacerbated, as every additional particle drives up the reward estimation cost.

This work was supported by the Israel Ministry of Innovation, Science and Technology. (Corresponding author: Ron Benchetrit.)

The authors are with the Technion – Israel Institute of Technology, Haifa 32000, Israel: R. Benchetrit with the Faculty of Computer Science, I. Lev-Yehudi with the Technion Autonomous Systems Program (TASP), T. Shazman with the Faculty of Data and Decision Sciences, and V. Indelman with the Stephen B. Klein Faculty of Aerospace Engineering and the Faculty of Data and Decision Sciences (e-mail: {ronbenc, idanlev, tmyr}@campus.technion.ac.il; vadim.indelman@technion.ac.il).

Existing  $\rho$ POMDP solvers [9]–[11] fall almost entirely in the belief-simulator family, evaluating the reward once per node.  $\rho$ POMCP [22], by contrast, is a state-simulator  $\rho$ POMDP solver: it refines its beliefs incrementally, yet recomputes the reward from scratch on every update, at prohibitive computational cost as particle counts grow.

Two gaps thus remain: no per-node belief is at once *anytime* and *density-adaptive* (a *representation gap*), and no solver refines belief-dependent rewards without recomputing them from scratch (a *reward-computation gap*). We introduce **ADAPT** (Anytime Density-Adaptive Particle Tree), an online tree-search algorithm that closes both: it progressively refines each per-node belief over time yet sizes it to its local density, and pairs this with an incremental reward update that keeps belief-dependent rewards affordable under continual refinement. More specifically, our contributions are:

- 1) **ADAPT**, an online tree-search algorithm for discrete and continuous domains whose per-node particle belief is simultaneously *anytime* and *density-adaptive*, combining the strengths of the state-simulator and belief-simulator families.
- 2) A theoretical analysis showing that a single tolerance parameter suffices to bound ADAPT’s per-node reward error, for both state- and entropy-based rewards at finite computation, and that this error vanishes throughout the tree as planning time grows.
- 3) An incremental reward-update scheme for belief-dependent rewards, demonstrated on common entropy estimators, that avoids full recomputation and lowers their per-particle cost by an order of magnitude, making them affordable under continual belief refinement.
- 4) An empirical evaluation on continuous-space POMDP and  $\rho$ POMDP benchmarks, where ADAPT outperforms state-of-the-art baselines from both families.

## II. PRELIMINARIES

### A. POMDPs and $\rho$ POMDPs

A POMDP is defined by the tuple  $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{Z}, \mathcal{R}, \gamma \rangle$ , where  $\mathcal{S}$ ,  $\mathcal{A}$ ,  $\mathcal{O}$  are the state, action, and observation spaces;  $\mathcal{T}(s' | s, a)$  and  $\mathcal{Z}(o | a, s')$  are the transition and observation models;  $\mathcal{R}(s, a, s')$  is the state-dependent reward; and  $\gamma \in (0, 1]$  is the discount factor. Since the agent cannot observe the state directly, it maintains a belief  $b_t(s) = \mathbb{P}(s | h_t)$ , where the history  $h_t = (b_0, a_0, o_1, \dots, a_{t-1}, o_t)$  summarizes past actions and observations. The belief is a sufficient statistic for optimal decision-making [23]; the agent thus seeks an optimal policy  $\pi^* : b \mapsto a$  maximizing the expected total discounted reward  $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, \pi(b_t), s_{t+1}) | b_0]$ .

A  $\rho$ POMDP [17] extends this formulation by replacing  $\mathcal{R}$  with a belief-dependent reward  $\rho(b, a, b')$ . The value and action-value functions of a policy  $\pi$  become

$$V^\pi(b_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \rho(b_t, \pi(b_t), b_{t+1})\right], \quad (1)$$

$$Q^\pi(b, a) = \mathbb{E}_{b'}[\rho(b, a, b') + \gamma V^\pi(b')]. \quad (2)$$

For notational convenience we may write  $\rho(hao)$  and  $V^\pi(hao)$  to denote quantities at the belief reached from history  $h$  via

action  $a$  and observation  $o$ . As in the online POMDP planning methods considered here, we assume that the transition and observation models and the reward function are given.

### B. Online Tree Search

Online tree-search solvers approximate  $\pi^*$  by constructing a search tree rooted at the current belief and expanding it incrementally. Monte Carlo Tree Search (MCTS) builds this tree asymmetrically through four routines applied recursively at each node [24], [25]. *Selection* traverses existing nodes using an action criterion that balances exploitation of high-value branches with exploration of under-sampled ones, typically UCB1 [24]; *expansion* adds a new node when an unvisited action or observation is reached; *simulation* estimates the value of a new leaf, often via a rollout from a default policy; and *backpropagation* updates the value estimates of all ancestors along the trajectory. The search is therefore anytime, concentrating effort on the most promising branches. To handle continuous action and observation spaces, *double progressive widening* (DPW) [26] gradually expands the set of actions and observations considered at each node, keeping the branching factor manageable.

### C. Particle Filtering

Representing general beliefs exactly is intractable in large and continuous state spaces. *Particle filters* approximate a belief by a finite weighted set  $\{(s_i, w_i)\}_{i=1}^N$  of state samples and weights [23], commonly self-normalized,  $\hat{w}_i = w_i / \sum_j w_j$ . Unlike parametric approaches restricted to a fixed family, they are non-parametric and can represent arbitrary distributions.

Beliefs are updated sequentially by importance sampling [27]: each particle is propagated through the transition model,  $s'_i \sim \mathcal{T}(\cdot | s_i, a)$ , and reweighted by the observation likelihood,  $w'_i \propto w_i \mathcal{Z}(o | a, s'_i)$ .

*KLD-sampling* [12] increases the efficiency of particle filtering by adapting the number of particles on-the-fly. Intuitively, it uses few particles when the belief is concentrated and many when it is spread out. Assuming the belief is piecewise-constant over a discretization of the state space into bins (e.g., a multidimensional histogram), let  $k$  denote the number of non-empty bins, i.e. the effective support of the belief. KLD-sampling then sets the required number of particles to

$$N = \frac{k-1}{2\varepsilon} \left( 1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}} z_{1-\delta} \right)^3, \quad (3)$$

where  $\varepsilon$  is a tolerance,  $\delta$  a failure probability, and  $z_{1-\delta}$  the upper  $1 - \delta$  quantile of the standard normal distribution. Drawing  $N$  particles then ensures that the empirical bin distribution  $p^N$  satisfies

$$D_{\text{KL}}(p^N \| p) \leq \varepsilon \quad \text{with probability } 1 - \delta, \quad (4)$$

where  $p$  is the true bin distribution. The count grows with the support  $k$ , so more complex or multimodal beliefs are allocated more particles.

#### D. Information-Theoretic Measures

A belief's uncertainty is commonly quantified by its *entropy* [28]: the *Shannon entropy* for a discrete belief and the *differential entropy* for a continuous one,

$$H(b) = - \sum_{s \in \mathcal{S}} b(s) \log b(s), \quad h(b) = - \int_{\mathcal{S}} b(s) \log b(s) ds. \quad (5)$$

For general beliefs these have no closed form and are approximated from the particles by sample-based estimators. For a discrete belief, the Shannon entropy estimator is

$$\hat{H}(b) = - \sum_{s_i \in \mathcal{S}} \hat{w}_i \log \hat{w}_i. \quad (6)$$

For continuous beliefs, a more suitable choice is the estimator of [21], hereafter the *Boers estimator*, as discussed therein. It converges to the true entropy under mild assumptions but its cost is *quadratic* in  $N$ . For a posterior belief  $b'$  obtained from  $b$  by action  $a$  and observation  $o$ , it is

$$\begin{aligned} \hat{h}(b') = \log & \left[ \sum_{i=1}^N \mathcal{Z}(o|a, s'_i) \hat{w}_i \right] - \sum_{i=1}^N \hat{w}'_i \log \mathcal{Z}(o|a, s'_i) \\ & - \sum_{i=1}^N \hat{w}'_i \log \underbrace{\left[ \sum_{j=1}^N \mathcal{T}(s'_i|s_j, a) \hat{w}_j \right]}_{T_i}, \end{aligned} \quad (7)$$

where the prime (') marks quantities of the posterior belief  $b'$ :  $s'_i$  and  $\hat{w}'_i$  are the state and normalized weight of the  $i$ th posterior particle. We define the *information gain* of a belief update from  $b$  to  $b'$  as

$$\text{IG}(b, b') = h(b) - h(b'), \quad (8)$$

the resulting reduction in uncertainty.

### III. ADAPT

#### A. Algorithm

ADAPT incrementally constructs a belief tree by simulating future actions and observations from the current belief; its main `Simulate` procedure is outlined in Algorithm 1. `SelectAction( $h$ )` chooses the action branch to explore. With small discrete action spaces, common implementations enumerate the actions and apply UCB1, as in POMCP [4]; with large discrete or continuous action spaces, action progressive widening introduces new actions gradually and applies the selection criterion among existing children, as in POMCPOW [6]. `SelectObservation( $h, a$ )` chooses the successor observation branch. In small discrete observation spaces, an observation sampled from the generative model maps to its corresponding child [4]. In large discrete or continuous spaces, where exact observations may rarely or never repeat, observation progressive widening either samples a new observation or revisits an existing child [6]. Within a finite planning budget, these choices determine which branches are visited and hence which node beliefs receive further refinement.

Each belief node in the tree is represented by a set of weighted particles. Upon visitation, the algorithm adaptively refines the belief using the `AddParticles` function (line 8).

---

#### Algorithm 1 ADAPT

---

```

1: procedure SIMULATE( $h, d$ )
2:   if  $d = 0$  then
3:     return 0
4:   end if
5:    $a \leftarrow \text{SelectAction}(h)$ 
6:    $o \leftarrow \text{SelectObservation}(h, a)$ 
7:    $\varepsilon \leftarrow E(N(hao))$ 
8:   AddParticles( $hao, \varepsilon$ )  $\triangleright$  update belief with new
    particles
9:   if  $o \notin C(ha)$  then  $\triangleright$  new node
10:     $C(ha) \leftarrow C(ha) \cup \{o\}$ 
11:     $total \leftarrow \rho(hao) + \gamma \text{ROLLOUT}(hao, d - 1)$ 
12:  else
13:     $total \leftarrow \rho(hao) + \gamma \text{SIMULATE}(hao, d - 1)$ 
14:  end if
15:   $N(h) \leftarrow N(h) + 1$ 
16:   $N(ha) \leftarrow N(ha) + 1$ 
17:   $Q(ha) \leftarrow Q(ha) + \frac{total - Q(ha)}{N(ha)}$ 
18:  return  $total$ 
19: end procedure
    
```

---

The number of particles to add is determined by the current belief's density and a tolerance parameter  $\varepsilon$ . We choose KLD-sampling for `AddParticles`.

A newly encountered node—one whose observation is not yet in  $C(ha)$ , the set of observations already expanded under  $ha$ —is evaluated by a rollout, a previously visited one by a recursive `Simulate` call.

#### B. Anytime Tolerance Schedule

The schedule  $E(N(hao))$  (line 7 of Algorithm 1) sets the per-visit tolerance  $\varepsilon$  from a node's visitation count, tightening it—and thus adding particles—as the node is revisited. A faster-decaying schedule focuses more resources on each node, while a slower one performs faster simulations. We adopt the simple polynomial schedule

$$E(N) = \frac{\varepsilon_0}{N^m}, \quad m > 0, \quad \varepsilon_0 > 0,$$

whose exponent  $m$  sets the decay rate and whose constant  $\varepsilon_0$  is the tolerance at a node's first visit. We choose this family for its interpretable parameters and because every  $m > 0$  satisfies the vanishing-tolerance condition of Proposition 3.

The exponent also controls the cost of continual refinement. Once a node's occupied support stabilizes, Eq. (3) gives a particle count  $n(v) = \Theta(v^m)$  after  $v$  visits. Even with the incremental update introduced in Section IV-B, the Boers estimator requires  $O(n)$  work per particle insertion. Let  $W(v)$  denote the cumulative reward-update cost up to the  $v$ th visit. Summing over successive insertions gives  $W(v) = O(\sum_{j=1}^{n(v)} j) = O(v^{2m})$ . Thus, the amortized cost per visit,  $W(v)/v = O(v^{2m-1})$ , remains bounded for  $0 < m \leq 1/2$ . For  $O(1)$  insertions, as with state-dependent rewards or incremental Shannon entropy (Section IV-A), the same argument gives  $W(v) = O(v^m)$  and the condition  $0 < m \leq 1$ . The cost analysis constrains the range of  $m$ ; its choice within that range

depends on the trade-off between belief refinement and tree expansion. Section V-C2 evaluates the effect of scheduling and sensitivity to  $m$ .

### C. Adaptive Sampling via KLD-Sampling

Algorithm 2 summarizes our KLD-sampling `AddParticles`, which adaptively determines the number of particles required for accurate belief approximation [12]. At a visited node  $hao$ , we iteratively draw samples from the parent belief  $b(h)$ , propagate and weight them, and update the effective support  $k$  whenever a sample falls into a previously empty bin. Sampling continues until the number of accumulated particles reaches the target count  $N$  determined by  $k$  and the tolerance  $\varepsilon$  via (3), subject to a minimum count  $N_{\min}$ . As a result, each node's belief is refined to a precision matched to its complexity.

---

#### Algorithm 2 KLD-Sampling `ADDPARTICLES`

---

```

1: procedure ADDPARTICLES( $hao, \varepsilon$ )
2:    $n \leftarrow |b(hao)|$   $\triangleright$  number of particles
3:    $k \leftarrow$  number of occupied bins in  $b(hao)$ 
4:   while  $n < \max(N_{k,\varepsilon}, N_{\min})$  do  $\triangleright$  Eq. (3)
5:     sample  $s \sim b(h)$  according to particle weights
6:     sample  $s' \sim \mathcal{T}(\cdot | s, a)$   $\triangleright$  propagate via transition
       model
7:      $w \leftarrow \mathcal{Z}(o | a, s')$   $\triangleright$  observation likelihood
8:     insert  $(s', w)$  into  $b(hao)$ 
9:      $n \leftarrow n + 1$ 
10:    if  $s'$  occupies a new bin,  $k \leftarrow k + 1$ 
11:  end while
12: end procedure

```

---

### D. Relation to Existing Solvers

ADAPT's two properties (anytime belief refinement and density adaptivity) are governed by the tolerance schedule and the occupied-bin count. The schedule sets the anytime behavior: a large  $m$  tightens the tolerance quickly, while  $m \rightarrow 0$  approaches a fixed tolerance. Allowing the occupied-bin count  $k$  to vary makes particle allocation density-adaptive; holding it fixed makes allocation depend only on visitation through the schedule. With fixed  $k$  and  $m = 1$ , the target particle count grows linearly with visits, as in POMCPOW. In the limit  $m \rightarrow 0$ , each belief has a fixed particle count—a common count across nodes when  $k$  is held fixed, as in PFT-DPW [6], or a density-dependent count when  $k$  varies, as in AdaOPS [8].

### E. Theoretical Properties

For the analysis we assume, as in [12], a partition of the state space into bins, with the belief  $b$  assigning true mass  $p_k$  to bin  $k$ . KLD-sampling then guarantees that the empirical bin masses  $p_k^N$  satisfy  $D_{\text{KL}}(p^N \| p) \leq \varepsilon$  with probability  $1 - \delta$  (4). We further assume a bounded reward, taking  $R \in [0, R_{\max}]$  without loss of generality, and write  $R_k$  for the mean reward

over bin  $k$ , leaving the bin masses as the only approximated quantity.

**Proposition 1** (Bounded state-reward deviation). *Let  $\bar{R} = \sum_k p_k R_k$  be the node's expected state reward, and let  $\bar{R}^N = \sum_k p_k^N R_k$  be its KLD-sampling estimate at tolerance  $\varepsilon$ . Then*

$$|\bar{R}^N - \bar{R}| \leq R_{\max} \sqrt{2\varepsilon}$$

*with probability  $1 - \delta$ .*

*Proof.* Since  $R_k \in [0, R_{\max}]$ ,  $|\bar{R}^N - \bar{R}| = |\sum_k (p_k^N - p_k) R_k| \leq R_{\max} \|p^N - p\|_1$ ; Pinsker's inequality bounds  $\|p^N - p\|_1 \leq \sqrt{2 D_{\text{KL}}(p^N \| p)}$ , and the KLD sampling guarantee (4) yields the claim.  $\square$

The entropy estimate is biased, and  $\varepsilon$  bounds this bias as well. Under KLD-sampling's piecewise-constant model, both the differential entropy of a continuous belief and the discrete entropy differ from the bin Shannon entropy only by a sample-independent offset, hence share its plug-in bias.

**Proposition 2** (Bounded entropy bias). *Let  $\hat{H}^N = H(p^N)$  be the plug-in Shannon entropy of the empirical bin distribution over  $N$  particles, estimating the true bin entropy  $H(p)$ . Under the KLD-sampling model, its bias satisfies  $-\varepsilon \leq \text{Bias}(\hat{H}^N) \leq 0$ .*

*Proof.* Using the local entropy expansion in [29, Eq. (4.2)], the plug-in estimator satisfies  $\text{Bias}(\hat{H}^N) = \mathbb{E}[H(p^N)] - H(p) = -\mathbb{E}[D_{\text{KL}}(p^N \| p)]$ , so non-negativity of the divergence gives the upper bound  $\text{Bias}(\hat{H}^N) \leq 0$ . For the lower bound, the KLD-sampling derivation [12] models  $2N D_{\text{KL}}(p^N \| p)$  as  $\chi_{k-1}^2$ , whose mean gives  $\mathbb{E}[D_{\text{KL}}(p^N \| p)] = \frac{k-1}{2N}$ , and Eq. (3) ensures  $N \geq \frac{k-1}{2\varepsilon}$ ; hence  $\mathbb{E}[D_{\text{KL}}(p^N \| p)] \leq \varepsilon$  and  $\text{Bias}(\hat{H}^N) \geq -\varepsilon$ .  $\square$

The bounds of Propositions 1 and 2 shrink as  $\varepsilon \rightarrow 0$ ; it remains to verify that the schedule drives  $\varepsilon \rightarrow 0$  throughout the tree. This holds under a single mild assumption.

**Assumption 1** (Consistent selection). The action- and observation-selection strategies (lines 5 and 6 of Algorithm 1) are *consistent*: as a node's visitation count grows, every one of its children is selected infinitely often. This is a property of the selection widening rather than of the domain—polynomial progressive widening [30] is consistent in this sense. Definition 1 in the Appendix states this formally via the widening functions.

**Proposition 3** (Asymptotic refinement). *Under Assumption 1 and any schedule with  $E(N) \rightarrow 0$  as  $N \rightarrow \infty$ , the per-node reward-estimation bounds of Propositions 1 and 2 tighten to zero at every node of the belief tree as planning time grows.*

*Proof.* By Assumption 1, each child of a node is selected infinitely often as the node is visited, imposing a deterministic, divergent lower bound on visitation counts: composing from the root down,  $N(hao) \rightarrow \infty$  at every node as planning time grows. Any schedule with  $E(N) \rightarrow 0$  then drives  $\varepsilon = E(N(hao)) \rightarrow 0$ , and the bounds of Propositions 1 and 2 vanish accordingly.  $\square$

#### IV. INCREMENTAL REWARD COMPUTATION

Because ADAPT refines its belief representation incrementally, recomputing belief-dependent rewards from scratch on every visit is prohibitively inefficient. We derive incremental updates for the two entropy estimators of Section II-D, the Shannon (6) and Boers (7) estimators. The approach is not specific to entropy: it applies whenever the reward decomposes additively into contributions of individual particles, or of particle pairs as in (7), so that inserting a particle affects only the terms it participates in. The empirical mean and covariance of the belief, and rewards built from them, are one such family, and incremental entropy estimation also applies to kernel-density approximations [9], [31].

##### A. Incremental Shannon Entropy

Recomputing the Shannon estimator (6) from scratch after each new particle costs  $O(N)$ . Using the factorization

$$\hat{H}(b) = -\frac{1}{\sum_{s_j \in \mathcal{S}} w_j} \sum_{s_i \in \mathcal{S}} w_i \log w_i + \log \left( \sum_{s_j \in \mathcal{S}} w_j \right), \quad (9)$$

the introduction of a new particle  $s_k$  changes only  $w_k$  while the remaining weights are unaffected.<sup>1</sup> Caching the previous  $\hat{H}(b)$  and  $\sum_{s_j \in \mathcal{S}} w_j$  allows the entropy to be updated in  $O(1)$  time, avoiding full recomputation.

##### B. Incremental Boers Entropy Estimator

From-scratch evaluation of the Boers estimator (7) is impractical under incremental refinement; we reduce its quadratic cost to linear time per insertion. When  $b$  and  $b'$  are augmented with a new particle, the affected quantities are denoted with a tilde, e.g.,  $\tilde{w}_i$ ,  $\tilde{w}'_i$ , and  $\tilde{T}_i$ .

While all terms in (7) update incrementally, the  $T_i$  terms dominate the cost. The updated  $\tilde{T}_i$  for  $i = 1, \dots, N$  satisfies

$$\begin{aligned} \tilde{T}_i &= \sum_{j=1}^N \mathcal{T}(s'_i | s_j, a) \tilde{w}_j + \mathcal{T}(s'_i | s_{N+1}, a) \tilde{w}_{N+1} \\ &= \frac{\sum_{j=1}^N w_j}{\sum_{j=1}^{N+1} w_j} T_i + \mathcal{T}(s'_i | s_{N+1}, a) \tilde{w}_{N+1}. \end{aligned} \quad (10)$$

Caching  $T_i$  and the weight sum lets (10) be updated in  $O(1)$  for each  $i = 1, \dots, N$ . The new term  $\tilde{T}_{N+1}$  must be computed from scratch at  $O(N)$  cost, but only once per inserted particle. Overall, a single incremental update costs  $O(N)$  rather than  $O(N^2)$ , substantially reducing the per-insertion overhead; we quantify this runtime advantage empirically in Section V-C1.

#### V. EXPERIMENTS

We evaluate our approach on two continuous-space benchmarks, *Continuous Light-Dark* and *Slippery-Roomba*, against  $\rho$ POMCPOW and  $\rho$ PFT-DPW,  $\rho$ POMDP adaptations of state-of-the-art solvers [6] representative of the state-simulator and belief-simulator families. Unlike classical POMCPOW,  $\rho$ POMCPOW initializes each node belief with  $n_{\text{init}}$  particles and carries them through rollout so that belief-dependent

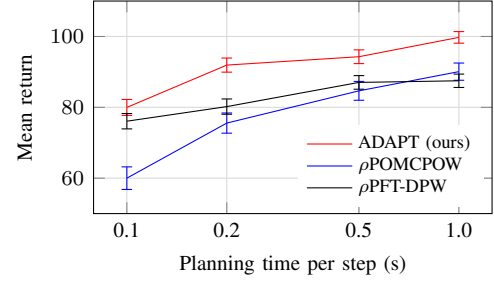


Fig. 2. Continuous Light-Dark: mean return vs. planning-time budget over 500 episodes. Error bars are  $\pm 1$  standard error. Higher is better.

rewards can be computed. Our ablations examine two variants of ADAPT: Batch-ADAPT and Density-Neutral ADAPT (DN-ADAPT). Each result reports mean return  $\pm$  standard error over 500 episodes. All solvers are implemented in Julia within the POMDPs.jl framework [32]. For each solver, hyperparameters were selected by grid search using random seeds disjoint from those used for evaluation. Our implementation, experiment code, benchmark and solver parameters, tuning details, and hardware specifications are available at <https://github.com/ronbenc/adapt-rhopomdp>, together with the supplementary video.

##### A. Continuous Light-Dark

The Light-Dark problem places the agent at an unknown position on a continuous line, with a discrete action set: a signed step along the line, or a *declare* action ending the episode. Motion is affected by constant Gaussian noise, while observations are noisy position measurements whose noise increases linearly with distance from the light source. Declaring inside the goal region yields +100 and −100 otherwise; other steps each incur a −1 penalty. We use the belief-dependent reward

$$\rho(b, a, b') = \mathbb{E}_{s, s'}[\mathcal{R}(s, a, s')] + 30 \text{IG}(b, b'),$$

with IG from Eq. (8), evaluated using the Boers estimator in Eq. (7). The optimal policy must trade direct goal progress against detours that localize the agent near the light. Figure 2 reports mean return at four planning-time budgets  $T \in \{0.1, 0.2, 0.5, 1.0\}$  seconds.

ADAPT achieves the highest mean return at every budget.  $\rho$ POMCPOW improves steadily with more planning time, while  $\rho$ PFT-DPW changes little from 0.5 to 1.0 seconds.

##### B. Slippery-Roomba

We adapt the Roomba domain [33] by introducing stochastic transitions and a family of highly ambiguous map layouts. The state is a robot pose  $(x, y, \theta) \in \mathbb{R}^3$ , and at each step the agent selects a discrete combination of forward velocity  $v$  and angular velocity  $\omega$ . Motion follows a stochastic unicycle model in which both the heading and forward-step noise scale with the commanded  $\omega$  and  $v$ , respectively. Observations are noisy ray-cast distances to the nearest wall in the heading direction, with measurement noise proportional to the true distance.

<sup>1</sup>Assuming identical state particles are merged and their weights summed.

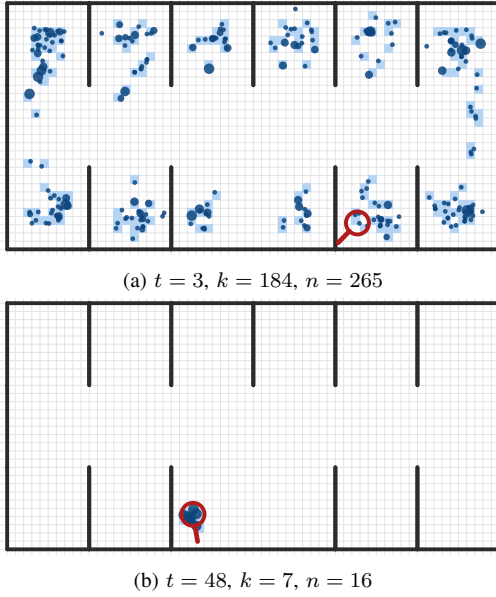


Fig. 3. ADAPT particle allocation in Slippery-Roomba ( $N = 6$ ),  $\varepsilon(v) = v^{-0.5}$ . Both have  $v = 6$  visits and  $\varepsilon \approx 0.41$ . The grid shows the KLD-sampling histogram bins of Section III-C; blue shading shows the  $k$  occupied bins; dot size indicates weight among the  $n$  particles; red marks the true pose.

Perceptual aliasing among rooms gives rise to multimodal posterior beliefs, and together with the stochastic dynamics this makes the domain especially challenging for online POMDP solvers. We use two map families parameterized by the number of rooms  $N$ . In the *one-sided corridor* all rooms open onto a single corridor; the *two-sided corridor* (Fig. 3) mirrors the rooms across the corridor, doubling the perceptual aliasing and yielding the multimodal beliefs that motivate ADAPT’s adaptive belief representation. On this domain we evaluate two complementary tasks: *Goal Navigation*, a pure POMDP with only state-dependent rewards that tests ADAPT as a generic POMDP solver, and *Active Localization*, a  $\rho$ POMDP whose reward is pure information gain. All solvers use a planning budget of 1 s per step for both tasks.

1) *Goal Navigation*: Each map contains a goal region and a stairs region. Reaching the goal yields a large positive reward, reaching the stairs incurs a large negative penalty, and every step incurs a small movement penalty and an additional penalty on wall contact. The maps are designed to be highly ambiguous: a solver that rushes toward the goal without first reducing positional uncertainty is likely to misidentify its room and step into the stairs. Figure 4 reports mean discounted return on the one-sided and two-sided corridors at  $N \in \{3, 4, 5, 6, 7, 8\}$ .

ADAPT outperforms the baselines at every  $N$  and on both layouts, demonstrating that its anytime adaptive belief representation is effective for generic POMDP planning. The advantage is more pronounced on the two-sided corridor, where the stronger perceptual aliasing makes accurate belief representation more critical.

2) *Active Localization*: In this  $\rho$ POMDP variant the agent must actively localize itself: there is no goal to reach, and the episode runs for a finite horizon of  $T = 50$  steps.

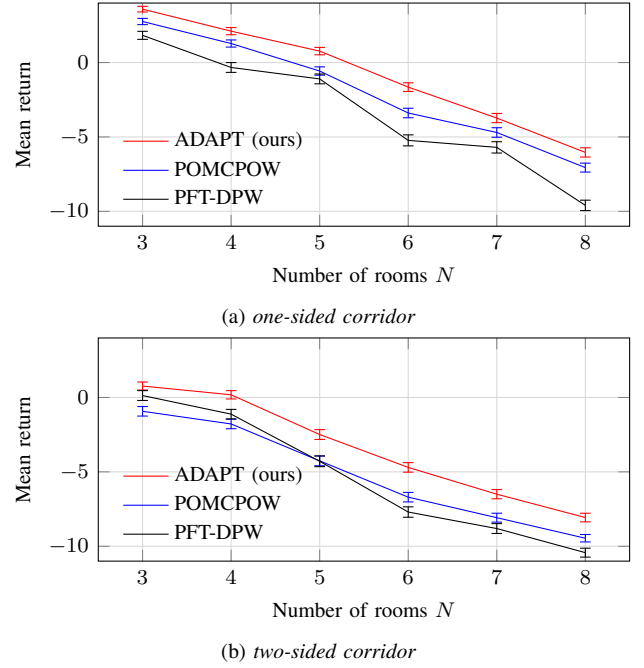


Fig. 4. Goal Navigation: mean discounted return  $\pm$  SE over 500 episodes versus the number of rooms  $N$ , on the (a) one-sided and (b) two-sided corridors. All solvers use a planning budget of 1 s per step.

TABLE I  
ACTIVE LOCALIZATION ON THE  $N = 6$  TWO-SIDED CORRIDOR: MEAN REWARD  $\pm$  STANDARD ERROR OVER 500 EPISODES. ALL SOLVERS USE A PLANNING BUDGET OF 1 s PER STEP.

| Active Localization on <i>two-sided corridor</i> , $N=6$ |                                    |
|----------------------------------------------------------|------------------------------------|
| Algorithm                                                | Mean reward $\pm$ SE               |
| <b>ADAPT (ours)</b>                                      | <b><math>-0.10 \pm 0.05</math></b> |
| $\rho$ POMCPOW                                           | $-0.91 \pm 0.11$                   |
| $\rho$ PFT-DPW                                           | $-0.77 \pm 0.10$                   |

The state-dependent reward  $R(s, a, s')$  carries only the per-step movement and wall-contact penalties. We use the belief-dependent reward

$$\rho(b, a, b') = \mathbb{E}_{s, s'}[R(s, a, s')] + 0.1 \text{IG}(b, b'),$$

with IG from Eq. (8), evaluated using the Boers estimator in Eq. (7). We evaluate on the two-sided corridor with  $N = 6$  rooms, whose stronger perceptual aliasing makes localization more challenging and highlights the value of accurate belief representation. Table I reports the mean reward.

ADAPT achieves the highest mean reward, outperforming both  $\rho$ POMCPOW and  $\rho$ PFT-DPW under the stronger aliasing of this domain.

### C. Ablation Studies

We conduct both ablations on Continuous Light-Dark.

1) *Effect of Incremental Reward Computation*: Batch-ADAPT recomputes the Boers estimator after each belief refinement, whereas DN-ADAPT retains anytime belief refinement and incremental reward computation but replaces KLD-based allocation with  $\rho$ POMCPOW’s visitation-driven particle growth. We compare ADAPT with Batch-ADAPT and

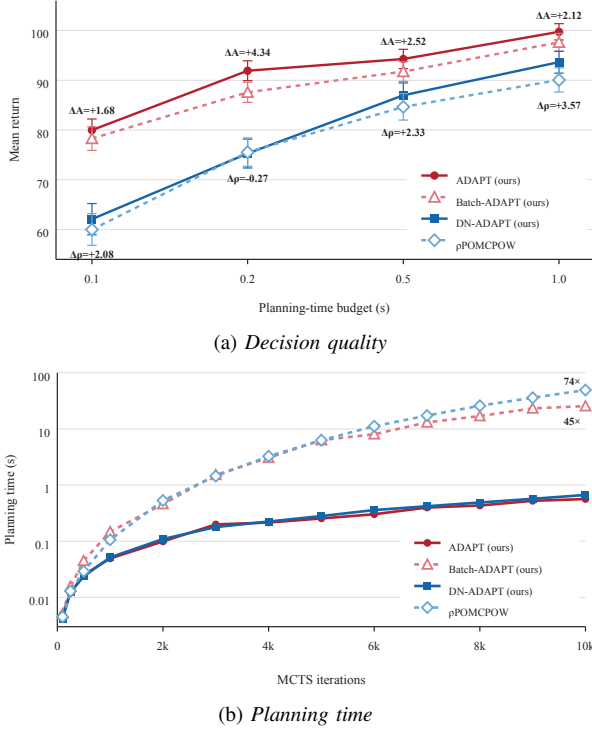


Fig. 5. Incremental reward computation on Continuous Light-Dark. (a) Mean return  $\pm$  SE over 500 episodes under matched planning-time budgets;  $\Delta$  denotes incremental minus batch computation. (b) Planning time at fixed MCTS iteration counts, with a logarithmic vertical axis. At 10k iterations, ADAPT achieves a 45 $\times$  speedup over Batch-ADAPT, and DN-ADAPT a 74 $\times$  speedup over  $\rho$ POMCPOW.

DN-ADAPT with  $\rho$ POMCPOW, its counterpart with batch reward computation. Figure 5 reports their decision quality under matched planning-time budgets and isolates reward-computation cost using fixed-iteration runs in which paired variants use identical seeds and parameters and expand the same tree.

Incremental ADAPT achieves a higher mean return at all four budgets. ADAPT sees a smaller speedup because its belief growth slows as  $k$  stabilizes and  $\varepsilon$  decreases polynomially, whereas  $\rho$ POMCPOW adds particles at a constant rate.

2) *Effect of the Tolerance Schedule*: We compare ADAPT’s polynomial schedule,  $\varepsilon(v) = v^{-0.4}$ , with fixed-tolerance variants, whose node beliefs are not refined after their initial construction. Figure 6 compares their decision quality and examines sensitivity to the exponent  $m$  in  $\varepsilon(v) = v^{-m}$ .

The scheduled and fixed variants perform similarly at the shortest budget, while the scheduled variant performs best at 1.0 s. Performance generally decreases for  $m > 0.4$ , suggesting that faster tolerance reduction can devote too much computation to belief refinement in this problem.

## VI. CONCLUSIONS AND LIMITATIONS

We introduced **ADAPT**, an online tree-search algorithm whose per-node particle belief is simultaneously anytime and density-adaptive, closing the representation gap left open by the state-simulator and belief-simulator families of particle-based POMDP solvers. We further showed that a single tolerance parameter bounds ADAPT’s per-node reward error, for

both state- and entropy-based rewards at finite computation, and, under suitable selection strategies, this error vanishes as planning time grows.

To make information-theoretic rewards affordable under continual belief refinement, we derived an incremental update scheme that lowers, by an order of magnitude, the per-particle cost of common entropy estimators in both discrete and continuous domains. This update is essential to ADAPT’s anytime refinement under belief-dependent rewards, and applies to other state-simulator  $\rho$ POMDP solvers.

Experimental results on continuous Light-Dark and Slippery-Roomba, spanning both POMDP goal navigation and  $\rho$ POMDP active localization, show that ADAPT outperforms state-of-the-art baselines from both families, and that incremental reward computation is essential for  $\rho$ POMDP solvers to scale.

Like other particle-based solvers, ADAPT remains subject to the curse of dimensionality: the number of occupied bins, and thus the required number of particles, can grow rapidly with state dimension and bin resolution. The bin resolution and tolerance schedule are ADAPT’s main additional domain-dependent parameters; both trade belief accuracy against computation and therefore require calibration for a new domain. ADAPT improves belief approximation under the supplied models, but cannot compensate for misspecified transition or observation models or an unsuitable belief-dependent reward. While we have shown that ADAPT’s per-node reward error vanishes with computation, convergence of the value function and the induced policy remains open. Bridging per-node reward accuracy to end-to-end optimality guarantees is a promising direction for future work.

## APPENDIX

### Formal statement of consistent selection (Assumption 1)

We make Assumption 1 precise in terms of the widening functions of the selection strategies.

**Definition 1** (Consistent selection). Let  $N(v; t)$  be the visitation count of node  $v$  at iteration  $t$ . The action- (resp. observation-) selection strategy is *consistent* if there exist non-decreasing functions  $f, F$  (resp.  $g, G$ ) with  $\lim_{n \rightarrow \infty} F(n) = \infty$  (resp.  $\lim_{n \rightarrow \infty} G(n) = \infty$ ) such that, for any node  $v$  with  $N(v; t) \geq f(i)$ , its  $i$ -th child  $v_i$  satisfies  $N(v_i; t) \geq F(N(v; t))$ .

Under Definition 1, for a belief-tree path  $h_\tau = a_{i_0} o_{j_1} \dots a_{i_{\tau-1}} o_{j_\tau}$ , once  $t \geq k(i_0, j_1, \dots, i_{\tau-1}, j_\tau)$  the visitation count satisfies

$$N(h_\tau; t) \geq K_\tau(t) = \underbrace{G \circ F \circ \dots \circ G}_{\tau \text{ times}}(t), \quad K_0(t) = t,$$

where the threshold pulls each path gate back through the corresponding partial composition,

$$k(i_0, \dots, j_\tau) = \max \{ K_0^{-1}(f(i_0)), \dots, K_\tau^{-1}(g(j_\tau)) \}.$$

Since  $F$  and  $G$  are non-decreasing and diverge,  $\lim_{t \rightarrow \infty} N(h_\tau; t) = \infty$ , so every node is visited infinitely often—the divergent lower bound invoked in the proof of

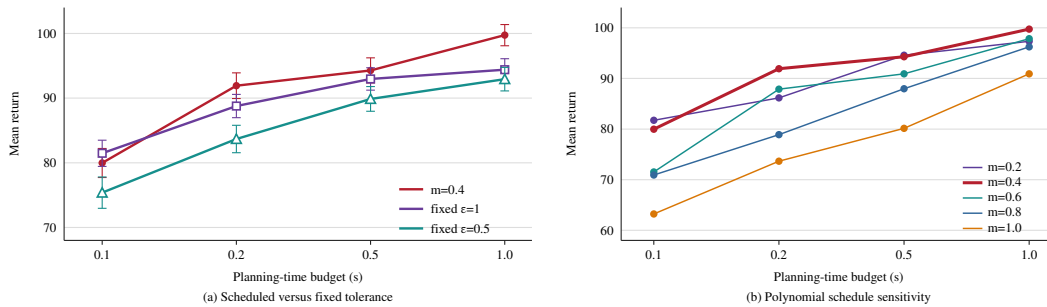


Fig. 6. Effect of the tolerance schedule on Continuous Light-Dark. (a) ADAPT's polynomial schedule compared with fixed-tolerance variants. (b) Sensitivity to the exponent  $m$ . Results show mean return  $\pm$  SE over 500 episodes.

**Proposition 3.** The progressive-widening strategy of [30] is consistent in this sense, with

$$\begin{aligned} f(i) &= i^{\frac{1}{\alpha_a(1-\alpha_a)}}, & F(n) &= \frac{1}{4}n^{e(d)(1-\alpha_a)}, \\ g(i) &= \left\lceil (i+1)^{1/\alpha_o} \right\rceil, & G(n) &= \frac{n}{\lfloor n \rfloor^{\alpha_o}} - 1, \end{aligned}$$

where  $\alpha_a, \alpha_o$  are the widening exponents and  $e(d)$  the depth dependent exploration exponent.

#### REFERENCES

- [1] M. Lauri, D. Hsu, and J. Pajarinen, "Partially observable Markov decision processes in robotics: A survey," *IEEE Trans. Robotics*, vol. 39, no. 1, pp. 21–40, 2023.
- [2] C. Papadimitriou and J. Tsitsiklis, "The complexity of Markov decision processes," *Mathematics of operations research*, vol. 12, no. 3, pp. 441–450, 1987.
- [3] H. Kurniawati, "Partially observable Markov decision processes and robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 253–277, 2022.
- [4] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2010, pp. 2164–2172.
- [5] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "DESPOT: Online POMDP planning with regularization," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 13, 2013, pp. 1772–1780.
- [6] Z. Sunberg and M. Kochenderfer, "Online algorithms for POMDPs with continuous state, action, and observation spaces," in *Intl. Conf. on Automated Planning and Scheduling (ICAPS)*, vol. 28, no. 1, 2018.
- [7] 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.
- [8] C. Wu, G. Yang, Z. Zhang, Y. Yu, D. Li, W. Liu, and J. Hao, "Adaptive online packing-guided search for POMDPs," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 28419–28430, 2021.
- [9] 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.
- [10] M. Barenboim and V. Indelman, "Adaptive information belief space planning," in *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, July 2022.
- [11] 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.
- [12] D. Fox, "Adapting the sample size in particle filters through KLD-sampling," *Intl. J. of Robotics Research*, vol. 22, no. 12, pp. 985–1003, 2003.
- [13] W. Burgard, D. Fox, and S. Thrun, "Active mobile robot localization," in *Intl. Joint Conf. on AI (IJCAI)*. Citeseer, 1997, pp. 1346–1352.
- [14] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using Rao-Blackwellized particle filters," in *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.
- [15] G. Flaspohler, V. Preston, A. P. Michel, Y. Girdhar, and N. Roy, "Information-guided robotic maximum seek-and-sample in partially observable continuous environments," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 4, pp. 3782–3789, 2019.
- [16] M. A. do Carmo Alves, A. Varma, Y. Elkhatib, and L. Soriano Marcolino, "Information-guided planning: an online approach for partially observable problems," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, pp. 69 157–69 177, 2023.
- [17] M. Araya, O. Buffet, V. Thomas, and F. Charpillet, "A POMDP extension with belief-dependent rewards," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2010, pp. 64–72.
- [18] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Pérez, "Belief space planning assuming maximum likelihood observations," in *Robotics: Science and Systems (RSS)*, Zaragoza, Spain, 2010, pp. 587–593.
- [19] J. Van Den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *Intl. J. of Robotics Research*, vol. 31, no. 11, pp. 1263–1278, 2012.
- [20] V. Indelman, L. Carlone, and F. Dellaert, "Planning in the continuous domain: a generalized belief space approach for autonomous navigation in unknown environments," *Intl. J. of Robotics Research*, vol. 34, no. 7, pp. 849–882, 2015.
- [21] 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.
- [22] V. Thomas, G. Hutin, and O. Buffet, "Monte Carlo information-oriented planning," in *Proc. 24th European Conference on Artificial Intelligence (ECAI)*, 2020, pp. 2378–2385.
- [23] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT press, Cambridge, MA, 2005.
- [24] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," in *European Conf. on Machine Learning (ECML)*. Springer, 2006, pp. 282–293.
- [25] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of Monte Carlo tree search methods," *IEEE Trans. on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.
- [26] A. Couëtoux, J.-B. Hoock, N. Sokolovska, O. Teytaud, and N. Bonnard, "Continuous upper confidence trees," in *Learning and Intelligent Optimization (LION)*, ser. LNCS, vol. 6683. Springer, 2011, pp. 433–445.
- [27] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. Springer, 2001.
- [28] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Wiley-Interscience, 2006.
- [29] L. Paninski, "Estimation of entropy and mutual information," *Neural Computation*, vol. 15, no. 6, pp. 1191–1253, 2003.
- [30] D. Auger, A. Couëtoux, and O. Teytaud, "Continuous upper confidence trees with polynomial exploration-consistency," in *European Conf. on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*. Springer, 2013, pp. 194–209.
- [31] Y. He, J. Jiang, D. Dai, and K. Fabrice, "An incremental kernel density estimator for data stream computation," *Complexity*, vol. 2020, pp. 1–17, 02 2020.
- [32] M. Egorov, Z. N. Sunberg, E. Balaban, T. A. Wheeler, J. K. Gupta, and M. J. Kochenderfer, "POMDPs.jl: A framework for sequential decision making under uncertainty," *J. of Machine Learning Research*, vol. 18, no. 1, pp. 831–835, 2017.
- [33] K. Menda, Z. Sunberg, and M. Kochenderfer, "RoombaPOMDPs.jl," <https://github.com/sisl/RoombaPOMDPs.jl>, 2021.