Anytime Incremental ρPOMDP Planning in Continuous Spaces

Ron Benchetrit

Anytime Incremental ρPOMDP Planning in Continuous Spaces

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

Ron Benchetrit

Submitted to the Senate
of the Technion — Israel Institute of Technology
Tammuz 5785 Haifa July 2025

This research was carried out under the supervision of Associate Professor Vadim Indleman, in the Department of Computer Science, Technion

The author of this thesis states that the research, including the collection, processing and presentation of data, addressing and comparing to previous research, etc., was done entirely in an honest way, as expected from scientific research that is conducted according to the ethical standards of the academic world. Also, reporting the research and its results in this thesis was done in an honest and complete manner, according to the same standards.

Some results in this thesis have been published as articles by the author and research collaborators in conferences and journals during the course of the author's masters research period, the most up-to-date versions of which being:

Ron Benchetrit, Idan Lev-Yehudi, and Vadim Indelman. Anytime adaptive planning. To be submitted, 2025.

The Technion's funding of this research is hereby acknowledged.

Contents

List of Figures

Abstract

Partially Observable Markov Decision Processes (POMDPs) provide a principled framework for decision-making under uncertainty, with applications in domains such as autonomous driving and robotic exploration. Their extension, ρPOMDPs, introduces belief-dependent rewards, enabling explicit reasoning about uncertainty. However, existing online ρPOMDP solvers either rely on fixed belief representations or compute belief-dependent rewards inefficiently, significantly limiting their scalability and practical applicability. We present *ρPOMCPOW*, an anytime solver that dynamically refines belief representations over time. We formally show that, under mild assumptions, the belief representation—and consequently the belief-dependent rewards—converges as the algorithm progresses. To mitigate the high computational cost of evaluating belief-dependent rewards, we propose a novel incremental computation approach. We demonstrate its effectiveness for common entropy estimators, achieving orders-of-magnitude reductions in computational overhead. Experimental results show that *ρPOMCPOW* outperforms state-of-the-art solvers in both computational efficiency and solution quality. Building on this foundation, we introduce *ADAPT*, an extension of ρPOMCPOW that provides an anytime and adaptive belief representation using KLD-sampling. ADAPT dynamically allocates particles based on node visitation frequency and the structural complexity of the belief, enabling accurate and scalable planning in high-dimensional ρPOMDPs.

Abbreviations and Notations

POMDP	:	Partially Observable Markov Decision Process
ρPOMDP	:	POMDP with belief-dependent rewards
$\mathcal{S}$	:	State space
$\mathcal{A}$	:	Action space
$\mathcal{O}$	:	Observation space
$\mathcal{T}$	:	State transition model
$\mathcal{Z}$	:	Observation model
$\mathcal{R}$	:	State-dependent reward function
γ	:	Discount factor
ρ	:	Belief-dependent reward function
s_t	:	State at time t
a_t	:	Action at time t
o_t	:	Observation at time t
h_t	:	History at time t
b_t	:	Belief at time t
π	:	Policy mapping beliefs to actions
π^*	:	Optimal policy
$V^\pi(b)$	:	Value function under policy π for belief b
$Q^\pi(b,a)$	:	Action-value function under policy π for belief b and action a
MCTS	:	Monte Carlo Tree Search
$N(h)$	:	Number of visits to history h
$H(b)$	:	Entropy of belief b

Chapter 1

Introduction

In applications such as robotics, autonomous driving, environmental monitoring, and space exploration, autonomous agents must make informed decisions under uncertainty. This uncertainty arises both from stochastic dynamics in the environment and from limitations in the agent's sensing and modeling capabilities. Sources include sensor noise, incomplete or inaccurate models, and unpredictable changes in the world. A widely adopted framework for reasoning and acting under such conditions is the *Partially Observable Markov Decision Process* (POMDP).

In POMDPs, due to the partial observability, the underlying state is unknown and decision-making relies on the history of past actions and observations, but storing this history over long trajectories is impractical. Instead, beliefs—probability distributions over unobserved states—serve as sufficient statistics, encoding all necessary information for optimal decision-making [TBF05]. A solution to a POMDP is a policy that maps each belief to an action that maximizes the expected sum of future rewards. However, finding exact solutions is computationally infeasible except for trivial cases [PT87], prompting the development of approximate algorithms.

In POMDPs, the agent cannot directly observe the underlying state of the environment. As a result, decision-making must rely on the history of past actions and observations. However, storing and processing long histories is impractical in real-world applications. Instead, the agent maintains a *belief*—a probability distribution over possible states—which serves as a sufficient statistic for optimal decision-making [TBF05]. A solution to a POMDP is a *policy* that maps each belief to an action in order to maximize the expected cumulative reward. Unfortunately, computing an exact optimal policy is computationally infeasible except for trivial cases [PT87], motivating the development of approximate solution methods.

Tree search algorithms are a prominent method for approximating solutions to POMDPs and are the focus of this work. Instead of evaluating all possible belief states—an infeasible task due to the immense size of the belief space—these algorithms concentrate on the subset of beliefs that can be reached from the initial belief through a sequence of actions and observations. In the online paradigm, a planner builds a

search tree at each time-step to determine an approximately optimal action. Anytime algorithms are particularly valuable in this setting, as they can provide progressively better solutions as computation time permits.

A POMDP with state-dependent rewards addresses uncertainty only indirectly, limiting its suitability for tasks like uncertainty reduction and information gathering. In contrast, belief-dependent rewards—defined over the state distribution—offer a more intuitive and effective framework for these tasks. For example, in active localization [BFT97], the goal is to minimize uncertainty about the agent's state rather than reaching a specific destination. Belief-dependent rewards have also been applied to problems with sparse rewards, aiding decision-making through reward-shaping techniques [FPM$^+$19, FT20] or as heuristics for guiding tree search [dCAVESM23].

When the reward function depends on the belief rather than the underlying state, the problem generalizes to a ρPOMDP [ABTC10], also known as *Belief Space Planning* (BSP) [PTKLP10, VDBPA12, ICD15]. Belief-dependent rewards are often derived from information-theoretic quantities such as entropy or information gain, which naturally capture the agent's uncertainty or knowledge about the world. However, computing these quantities in continuous state spaces is particularly challenging. Exact computation is intractable, and practical solutions rely on sample-based methods such as kernel density estimation or particle filtering [BDBM10]. These approaches are computationally expensive—typically scaling quadratically with the number of samples—and provide only asymptotic accuracy, requiring a large number of samples to achieve reliable estimates.

In this thesis, we address the challenge of planning in continuous-spaces ρPOMDPs. We introduce ρPOMCPOW, an anytime online solver for ρPOMDPs that incrementally refines belief representations over time. We show that, under mild assumptions, the belief representation—and consequently the belief-dependent rewards—converges as the algorithm progresses. To mitigate the high computational cost of evaluating belief-dependent rewards, we propose a novel incremental computation approach, demonstrating it on common entropy estimators. We evaluate the effectiveness of our approach on a variety of continuous-space ρPOMDPs, showing that ρPOMCPOW outperforms state-of-the-art solvers in both computational efficiency and solution quality. Building on this foundation, we introduce ADAPT, an extension of ρPOMCPOW that provides an anytime and adaptive belief representation using KLD-sampling.

Chapter 2

Related Work

We begin by reviewing state-of-the-art POMDP solvers, with particular emphasis on online algorithms for large or continuous domains. We then focus on methods specifically designed for solving ρPOMDPs with belief-dependent rewards.

2.1 State of the Art POMDP Solvers

Existing approaches for POMDP planning can be broadly categorized into *offline* and *online* methods, depending on whether policy computation occurs before or during execution.

Offline Solvers

Offline POMDP solvers aim to precompute a policy over the belief space prior to execution. These methods often rely on value iteration techniques and exploit the fact that, for finite-horizon or discounted problems with discrete state spaces, the optimal value function over beliefs is piecewise-linear and convex. It can therefore be represented using a finite set of α-vectors, each corresponding to a linear value function over the belief simplex.

Point-based value iteration (PBVI) methods [PGT$^+$03] approximate the value function using a finite set of sampled belief points, enabling tractable solutions in small to medium-sized discrete problems. Notable methods in this category include:

- **PBVI** [PGT$^+$03], which incrementally builds the value function over a sampled belief set.
- **HSVI** [SS05], which uses heuristic-guided forward exploration and maintains upper and lower bounds on the value function.
- **SARSOP** [KHL08], which improves efficiency by restricting updates to beliefs that are reachable under the optimal policy.

While these methods offer strong theoretical guarantees and yield compact policies that can be executed rapidly at runtime, they scale poorly to problems with large or continuous state, action, or observation spaces. Discretizing such spaces becomes infeasible, and approximate backup operations in continuous domains often require expensive sampling-based techniques.

These limitations have motivated the development of *online solvers*, which compute policies incrementally during execution by focusing on a local subset of the belief space relevant to the current state.

Online Solvers

Online POMDP solvers interleave planning and execution by constructing a local belief tree rooted at the current belief. This approach is particularly effective in large or continuous domains, where computing a global policy offline is infeasible. These solvers typically approximate the belief using a set of state samples, or *particles*, which allow for flexible and scalable representations of complex, multimodal belief distributions [TBF05].

Online POMDP solvers for large or infinite state spaces typically approximate the belief using a set of state samples, commonly referred to as particles. This particle-based representation is flexible and well-suited for capturing complex, multimodal beliefs [TBF05].

For discussion purposes, online solvers can be broadly categorized into two families: *state simulators* and *belief simulators*, depending on whether they simulate trajectories through the state space or explicitly maintain transitions between belief states. Figure 2.1 illustrates this distinction.

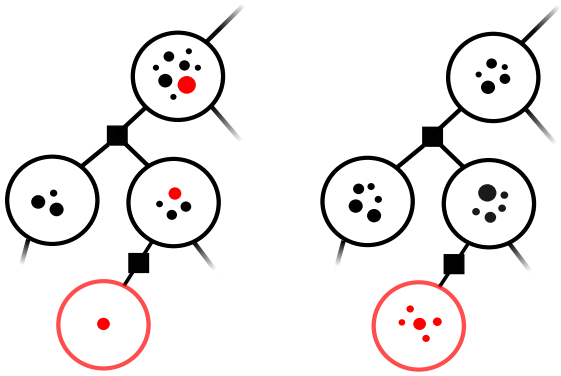

Figure 2.1: Illustration of belief tree construction by a state simulator (left) and a belief simulator (right). New particles and new nodes are marked in red. The state simulator updates beliefs by adding new particles along the trajectory, while the belief simulator maintains fixed beliefs once created.

State Simulators: State simulators operate by simulating state trajectories directly, incrementally updating visited beliefs with new particles at each visitation.

Examples of state simulators include:

- **POMCP** [SV10], which extends the UCT algorithm [KS06] to the POMDP framework.
- **DESPOT** [SYHL13] and its successors [YSHL17, GHL19], which use deterministic bounds and heuristic sampling to guide exploration.
- **POMCPOW** [SK18], which extends POMCP to continuous action and observation spaces by incorporating progressive widening.
- **LABECOP** [HK21], an algorithm for continuous observation spaces, extracts the belief from scratch for each sampled observation sequence.

A common trait of these algorithms is that each time a belief node is visited, the belief is updated with additional particles. Intuitively, this approach improves the belief representation in frequently visited nodes, aligning with the exploration-exploitation trade-off.

Belief Simulators: Belief simulators, on the other hand, treat POMDP belief states as nodes in an equivalent Belief-MDP. Examples include:

- **PFT-DPW** [SK18], which represents each belief node with a fixed number of particles.
- **AdaOPS** [WYZ+21], which dynamically adapts the number of particles per belief node and aggregates similar beliefs.

A key limitation of belief simulators is their fixed belief representation, which does not improve over time. This inefficiency leads to unpromising regions of the search space receiving the same computational effort as promising ones. Moreover, these algorithms are less flexible when planning times vary. Given ample time, belief simulators can construct dense trees, but belief representations at individual nodes may remain suboptimal. Under time constraints, however, they often produce shallow, sparse trees, as significant computational effort is spent maintaining fixed belief representations rather than effectively exploring the search space.

2.2 ρPOMDP Solvers

Solving ρPOMDPs—POMDPs with belief-dependent rewards—poses additional computational challenges, particularly in continuous domains where belief representations and reward evaluations are sample-based and expensive. Existing approaches include both offline and online planners.

Offline solvers for ρPOMDPs typically extend point-based value iteration (PBVI) to accommodate belief-dependent rewards. Araya et al. [ABTC10] demonstrated that if the reward function is convex over the belief space, then the resulting value function

is also convex. This structural property enables the use of modified PBVI techniques for approximating the optimal value function in ρPOMDPs.

Fehr et al. [FBTD18] further established that if the belief-dependent reward is Lipschitz continuous, then the value function inherits this property. They leveraged this insight to derive theoretical bounds that can be integrated into the Heuristic Search Value Iteration (HSVI) framework to improve convergence and error control.

While these results provide valuable theoretical foundations for ρPOMDP planning, they remain limited by the scalability challenges inherent to offline methods, especially in high-dimensional or continuous domains.

Online solvers have therefore received increasing attention as a more scalable alternative. Several algorithms have been proposed to address the specific challenges of online planning in ρPOMDPs.

- **PFT-DPW** [SK18] was introduced to accommodate belief-dependent rewards in POMDPs, though it was not demonstrated for this application.

- **IPFT** [FT20] builds on PFT-DPW, introducing the concept of reward shaping using information-theoretic rewards. It reinvigorates particles at each traversal of posterior nodes and estimates information-theoretic rewards by a kernel density estimator (KDE).

- **AI-FSSS** [BI22] reduces the computational cost of information-theoretic rewards by aggregating observations, providing bounds on the expected reward and value function to guide the search. Despite this improvement, its approach remains constrained by a fixed observation branching factor and a fixed number of particles per belief node.

- **SITH-PFT** [ZSI24] introduce an adaptive multilevel simplification paradigm for ρPOMDPs, which accelerates planning by computing rewards from a smaller subset of particles while bounding the introduced error. While their current implementation builds upon PFT-DPW, future extensions could complement our approach.

All the algorithms above belong to the *belief simulator* family and share a key limitation: they maintain fixed belief representations that do not improve with additional computation. As a result, computational resources may be wasted on low-value nodes, and beliefs at important nodes may remain coarse, especially under tight time constraints.

ρPOMCP [THB21] is an exception, closely related to our work. It extends POMCP to handle belief-dependent rewards by propagating a fixed set of particles from the root instead of simulating a single particle per iteration. Their approach includes variants such as Last-Reward-Update (LRU) which Last-Value-Update (LVU), which use the most recent reward estimates to reduce bias, unlike POMCP's running average.

However, ρPOMCP is limited to discrete state spaces and recomputes belief-dependent rewards from scratch whenever a belief node is updated. This approach is costly, particularly in continuous domains where the number of particles per belief can grow indefinitely. These limitations highlight the need for *efficient, incremental reward updates* to avoid full recomputation—an issue directly addressed by our approach.

Chapter 3

Preliminaries

This section reviews mathematical formulations and notations used in this work.

3.1 Partially Observable Markov Decision Processes (POMDPs)

A Partially Observable Markov Decision Process (POMDP) extends a Markov Decision Process (MDP) to settings in which the agent does not have full access to the environment's underlying state. Formally, a POMDP is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{Z}, \mathcal{R}, \gamma \rangle$, where:

- $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{O}$ denote the state, action, and observation spaces, respectively;
- $\mathcal{T}(s' \mid s, a)$ is the state transition model, representing the probability of transitioning to state s' given that the agent is in state s and takes action a;
- $\mathcal{Z}(o \mid s, a, s')$ is the observation model, defining the probability of receiving observation o given the current state s, action a, and next state s';
- $\mathcal{R}(s, a, s')$ is the reward function, specifying the reward received when transitioning from state s to s' under action a;
- $\gamma \in (0, 1]$ is the discount factor, which determines discounting of future rewards.

Due to partial observability, the agent maintains a *belief* b_t, which is a probability distribution over the states at time t. The belief is updated based on the history of past actions and observations. Let the history at time t be denoted by $h_t = (a_0, o_1, \ldots, a_{t-1}, o_t)$, and define the belief as $b_t(s) = \mathbb{P}((s \mid h_t))$. The belief serves as a sufficient statistic for optimal decision-making in POMDPs [TBF05].

The agent's objective is to find a policy π that maps beliefs to actions and maximizes the expected discounted return:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, \pi(b_t), s_{t+1}) \mid b_0\right].$$

The value function $V^\pi(b)$ of a belief state b under policy π is defined as the expected return starting from belief b and following policy π:

$$V^\pi(b) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, \pi(b_t), s_{t+1}) \mid b_0 = b\right],$$

where the expectation is taken over state transitions and observations induced by the policy π and the POMDP dynamics.

The action-value function $Q^\pi(b, a)$ is defined as the expected return when taking action a in belief b and following the policy π thereafter:

$$Q^\pi(b, a) = \mathbb{E}\left[\mathcal{R}(s, a, s') + \gamma\, V^\pi(b') \mid b, a\right],$$

where b' is the updated belief after executing action a and receiving observation o.

From Bellman's principle of optimality, the optimal value function $V^*(b)$ can be expressed recursively as:

$$V^*(b) = \max_{a \in \mathcal{A}} Q^*(b, a),$$

where $Q^*(b, a)$ is defined analogously to $Q^\pi(b, a)$ with $V^*(b')$ in place of $V^\pi(b')$.

3.2 ρPOMDPs

In this work, we primarily focus on an extension of POMDPs, often referred to as ρPOMDPs, where the reward function depends on the belief state. We replace the state-dependent reward $\mathcal{R}$ with the belief-dependent reward ρ, structured as $\rho(b, a, b')$ The value function under policy π is given by:

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

where the expectation is over future beliefs. The corresponding action-value function satisfies:

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

For notational convenience, we may use hao to implicitly encode the relevant data of the belief b' resulting from history h, action a, and observation o, enabling shorthand expressions such as $\rho(hao)$ and $V^\pi(hao)$.

3.3 Online Planning in POMDPs

Solving infinite-horizon POMDPs exactly is computationally intractable. Online tree search methods approximate the optimal policy by constructing a search tree—referred to as a *belief tree*—in real time. Starting from the current belief, the algorithm incrementally expands the tree by simulating possible future actions and observations up

to a fixed planning horizon. However, constructing the full belief tree is infeasible in practice due to the exponential branching factor, a phenomenon commonly referred to as the *curse of history*. Figure 3.1 shows an example of a belief tree.

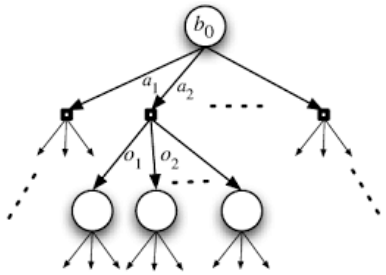

Figure 3.1: An illustration of a belief tree used in online POMDP planning. Nodes correspond to belief states, and edges correspond to actions and observations. The root node is the current belief, and the tree expands as the planner simulates future interactions. Adapted from [KHL08].

Monte Carlo Tree Search (MCTS) is powerful method that addresses this by iteratively performing four steps: selection, where the tree is traversed using a strategy like UCB [KS06] to balance exploration and exploitation; expansion, which adds new child nodes to the tree; simulation, where trajectories are simulated from the expanded nodes to estimate values; and backpropagation, which updates statistics along the path from the leaf to the root. This approach builds an asymmetric tree focused on promising regions of the search space and provides anytime solutions.

POMCP [SV10] extends UCT [KS06] to POMDPs by representing beliefs as particle sets and propagating state particles through the tree. This enables scalable planning for large discrete POMDPs but struggles in continuous spaces, where each simulation tends to create a new branch, resulting in sparse and shallow trees.

POMCPOW [SK18] extends POMCP to continuous action and observation spaces using progressive widening, which limits the creation of new branches and allows existing branches to accumulate particles. It also employs a weighted particle filter to mitigate particle degeneracy, ensuring a robust belief representation.

Chapter 4

ρPOMCPOW

We introduce ρPOMCPOW, an online tree search algorithm for solving ρPOMDPs with continuous state, action, and observation spaces. ρPOMCPOW extends POMCPOW by incorporating belief-dependent rewards and modifying the backpropagation step to adopt the Last-Value-Update (LVU) framework [THB21], ensuring that only the latest reward estimates are used in value updates.

4.1 Algorithm Overview

Similar to POMCPOW, ρPOMCPOW iteratively constructs a search tree by simulating state trajectories through alternating layers of action and observation nodes. Each iteration begins by sampling a state from the initial belief and traversing the tree using predefined selection strategies, such as progressive widening for continuous spaces. State particles are updated along the trajectory by simulating actions, weighting by the observation model, and resampling to maintain a representative particle distribution. The process continues until either a depth limit is reached or a leaf node is encountered, at which point a rollout is performed to estimate the node's value.

Unlike POMCPOW, ρPOMCPOW supports belief-dependent rewards, requiring modifications to the backpropagation step. Instead of using the classical Monte Carlo running average, which aggregates cumulative state-dependent rewards, ρPOMCPOW updates node values based on the most recent estimates from child nodes. While the LVU framework was introduced in [THB21], ρPOMCPOW further differs by implementing an incremental update mechanism for both value and action-value estimators. This mechanism efficiently adjusts estimates without recalculating them from scratch, significantly reducing computational overhead, particularly in continuous spaces. For full derivations of this novel incremental update mechanism, see Appendix A.1.

A key conceptual difference is that ρPOMCPOW simulation returns explicit value and action-value estimates rather than cumulative rewards along a trajectory. This adjustment is necessary for handling belief-dependent rewards and aligns with the LVU framework's emphasis on using the most recent reward estimates. To reflect

these changes, ρPOMCPOW restructures the simulation process into two procedures: `SimulateV` for propagating value estimates and `SimulateQ` for updating action-value estimates.

Algorithm 4.1 details the framework.

Algorithm 4.1 LVU ρPOMCPOW

procedure SIMULATEV(s, h, d)
  if $d = 0$ **then**
    return 0
  end if
  $a \leftarrow$ ActionSelection$(\dots)$
  $Q^{prev}(ha) \leftarrow Q(ha)$
  $Q(ha) \leftarrow$ SimulateQ(s, ha, d)
  $N(h) \leftarrow N(h) + 1$
  $V(h) \leftarrow V(h) + \frac{1}{N(h)}[N(ha)Q(ha) -$
    $(N(ha) - 1)Q^{prev}(ha) - V(h)]$
  return $V(h)$
end procedure

procedure SIMULATEQ(s', ha, d)
  Sample $s' \sim \mathcal{T}(\cdot|s, a)$
  $o \leftarrow$ ObservationSelection$(\dots)$
  Append s' to belief $B(hao)$
  Append $\mathcal{Z}(o|a, s')$ to weights $W(hao)$
  $\rho^{prev}(hao) \leftarrow \rho(hao)$, $V^{prev}(hao) \leftarrow V(hao)$
  $\rho(hao) \leftarrow$ UpdateReward$(\dots)$
  if $o \notin C(ha)$ **then** ▷ new node
    $C(ha) \leftarrow C(ha) \cup \{o\}$
    $V(hao) \leftarrow$ ROLLOUT$(s', hao, d - 1)$
  else
    Select $s' \in B(hao)$ based on weights $W(hao)$
    $V(hao) \leftarrow$ SimulateV$(s', hao, d - 1)$
  end if
  $N(ha) \leftarrow N(ha) + 1$
  $Q(ha) \leftarrow Q(ha) + \frac{N(hao)}{N(ha)}[\rho(hao) + \gamma V(hao)] -$
  $\frac{N(hao)-1}{N(ha)}[\rho^{prev}(hao) + \gamma V^{prev}(hao)] - \frac{1}{N(ha)}Q(ha)$
  return $Q(ha)$
end procedure

The `ActionSelection` and `ObservationSelection` functions are abstracted for flexibility and can incorporate strategies such as the ones presented in [SK18].

To enhance the accuracy of initial belief-dependent rewards and enable rollouts with belief-dependent rewards, a potential extension involves propagating a "bag of particles"—a fixed set of particles initialized at the root node and carried through each tree traversal, as suggested in [THB21]. While promising, this extension lies beyond the scope of this work and is left for future exploration.

4.2 Challenges and Discussion

The ρPOMCPOW algorithm introduces a novel approach for solving ρPOMDPs, but two critical aspects require deeper exploration to fully harness its capabilities.

First, belief representation within the search tree is critical for tasks such as information gathering, where accurately modeling uncertainty is essential. Although the algorithm accumulates particles at belief nodes based on visitation frequency, this mechanism alone does not ensure uniform convergence across the tree. Some nodes may remain underexplored, resulting in poorly approximated beliefs and reduced planning effectiveness. In the next section, we formally analyze this issue and show that, under action and observation selection strategies satisfying a consistency condition, every belief node is visited infinitely often. As a result, each node accumulates infinitely many particles, guaranteeing convergence to the true belief distribution.

Second, since ρPOMCPOW updates each visited belief along the simulated trajectory, belief-dependent rewards must also be updated, posing a significant computational challenge. These rewards are typically non-linear functions of the belief, making efficient updates non-trivial. Recomputing rewards from scratch for every new particle is expensive, particularly in continuous state spaces, where the number of particles grows unbounded. For instance, both KDE and [BDBM10] entropy estimators scale quadratically with the number of particles. In a subsequent section, we introduce an incremental belief-dependent reward computation, demonstrating how it enables efficient updates for various reward functions.

Chapter 5

Belief Convergence

In this chapter, we show that belief nodes converge to the true belief throughout the belief tree, thereby enabling convergence of belief-dependent rewards. We begin by defining a *consistent selection strategy*, which allows us to derive lower bounds on node visitation counts, thereby ensuring that each node is visited infinitely often. Since ρPOMCPOW updates belief nodes by accumulating particles in proportion to their visitation frequency, this result guarantees that each node will eventually contain infinitely many particles, leading to convergence to the true belief distribution under mild assumptions.

5.1 Node Visitation Count

Online tree search algorithms must balance broad exploration of the search space with refining estimates of existing nodes, a challenge known as the exploration-exploitation trade-off.

Algorithms such as UCT and POMCP address this trade-off by adopting the principle of optimism in the face of uncertainty, where actions are assumed to be promising until sufficient evidence suggests otherwise.

This challenge is amplified in continuous spaces, where the search tree can grow indefinitely, making it difficult to ensure adequate visitation across all nodes. In fact, it has been shown that in POMCP-DPW [SK18], when operating in continuous spaces, posterior nodes in the belief tree are visited only once, severely limiting their representation and hindering uncertainty estimation.

POMCPOW mitigates this issue through Progressive Widening, which controls the expansion rate of new child nodes. However, its effect on node visitation has not been formally analyzed.

In this section, we introduce the notion of *consistent selection strategies* and derive a deterministic lower bound on the visitation count of each node in the belief tree. Although these results apply generally to tree search algorithms, we focus on their implications for ρPOMCPOW. In particular, we show that under consistent selection

strategies, every belief node is guaranteed to be visited infinitely often. As a result, each node accumulates infinitely many particles over time, ensuring convergence of beliefs under mild assumptions.

Consistent Selection Strategies

Let $N(v;t)$ denote the visitation count of node v at the t^{th} iteration of the algorithm. We define a consistent selection strategy as follows:

Definition 5.1.1 (Consistent Selection Strategy)**.** A selection strategy is consistent if there exist non-decreasing functions f and F, where $\lim_{n\to\infty} F(n) = \infty$, such that for any node v with $N(v;t) \geq f(i)$, the visitation count of its i^{th} child vi satisfies:

$$N(vi;t) \geq F(N(v;t)).$$

This definition ensures that once a parent node has been visited sufficiently often, its child nodes are also visited proportionally. The function F guarantees that visitation counts grow over time, enabling all parts of the tree to be explored adequately. In Example 5.1.2, we provide specific instances of consistent selection strategies, derived from the work of [ACT13].

Node Visitation Lower Bound

Using consistent selection strategies, we establish a deterministic lower bound on the visitation count of each node in the belief tree.

Theorem 5.1 (Node Visitation Lower Bound)**.** *Assume the action and observation selection strategies are consistent with functions f, F and g, G, respectively. For a belief tree path h_τ, the visitation counts satisfy: For $h_\tau = a_{i_0} o_{j_1} a_{i_1} o_{j_2} \dots a_{i_{\tau-1}} o_{j_\tau}$, with $t \geq k(i_0, j_1, \dots, i_{\tau-1}, j_\tau)$:*

$$N(h_\tau;t) \geq K_\tau(t) = \underbrace{G \circ F \circ \cdots \circ G}_{\tau \text{ times}}(t).$$

Here, $k(i_0, \dots, j_\tau)$ ensures sufficient initial visitation counts. A more detailed version of this theorem, including the explicit closed-form expression for k and its complete proof, is provided in Appendix A.2.

Remark. Since F and G are non-decreasing and $\lim_{n\to\infty} F(n) = \lim_{n\to\infty} G(n) = \infty$, it follows that:

$$\lim_{t\to\infty} N(h_\tau;t) = \infty.$$

This guarantees that visitation counts grow indefinitely over time, ensuring sufficient exploration of nodes in the belief tree as the algorithm progresses.

Example 5.1.2. Algorithms A.1 and A.2 from [ACT13] establish a consistent selection strategy for action and observation nodes. A detailed proof, the algorithm descriptions, and specific instances of the functions f, F, g, and G are provided in Appendix A.3.

Convergence of Belief Nodes

In this section, we show that the belief nodes in the search tree converge to the true belief distribution as the number of particles approaches infinity. We begin by stating the assumptions required to establish this convergence.

Particle Filter Convergence 5.1.3. Let $\hat{b}_N$ denote the empirical belief represented by the particle filter at a node, where N is the number of particles. We assume that $\hat{b}_N$ converges in distribution to the true belief b as $N \to \infty$:

$$\hat{b}_N \xrightarrow{d} b.$$

While this assumption may appear standard—given that particle filters are well studied and known to converge under suitable conditions [Dou04]—the particle filter used in ρPOMCPOW is not conventional. Due to subtle deviations in the resampling stage introduced by the incremental nature of the algorithm, convergence is not automatically guaranteed. We discuss this assumption and its implications in more detail in chapter 8.

We will now conclude this section with a corollary that follows from Theorem 5.1 and Assumption 5.1.3.

Corollary 5.2 (Belief Convergence)**.** *In ρPOMCPOW, under consistent action and observation selection strategies, Theorem 5.1 guarantees that the visitation count of each node increases over time. Under Assumption 5.1.3, this implies that the belief nodes converge to the true belief distribution as the number of particles approaches infinity.*

The convergence of belief-dependent rewards is, in general, specific to the chosen reward function, but it typically follows from the convergence of the underlying belief nodes. In some cases, additional assumptions may be required to establish this result; see [BDBM10] for a concrete example.

Chapter 6

Incremental Reward Computation

As discussed in previous sections, updating belief-dependent rewards from scratch each time a belief is updated is prohibitively inefficient, particularly in continuous spaces where the number of particles grows indefinitely.

Most common belief-dependent rewards are information-theoretic and rely on entropy. We demonstrate incremental computation of Shannon entropy and the entropy estimator proposed by [BDBM10], referred to as the Boers entropy estimator, significantly reducing computational overhead compared to full recomputation.

These principles are general and extend to other belief-dependent rewards. For example, [HJDF20] proposes an incremental KDE update, which could be applied to KDE-based entropy estimators used in [FT20]. Additionally, they may benefit other algorithms, such as ρPOMCP.

We represent the belief as a set of particles $\{s_i, \hat{w}_i\}_{i=1}^{N}$, where s_i is a state sample and $\hat{w}_i$ is the normalized weight of the particle. The normalized weight is computed as $\hat{w}_i = \frac{w_i}{\sum_j w_j}$, where w_i is the unnormalized weight of particle s_i.

6.1 Incremental Computation of Shannon Entropy

While Shannon entropy is traditionally defined for discrete distributions, which are not the primary focus of this work, we present an incremental computation method tailored for particle-based belief representations to illustrate the broader feasibility of incremental computation for belief-dependent rewards.

The Shannon entropy of the particle belief is defined as:

$$\hat{H}(b) = -\sum_{s_i \in \mathcal{S}} \hat{w}_i \log \hat{w}_i. \tag{6.1}$$

Recomputing the entropy from scratch after introducing new particles incurs a computational cost proportional to the total number of particles, $O(N)$. As the number of

particles grows, this cost can become prohibitive. To address this, we use the following factorization:

$$\hat{H}(b) = -\frac{1}{\sum_{s_j \in \mathcal{S}} w_j} \sum_{s_i \in \mathcal{S}} w_i \log w_i + \log(\sum_{s_j \in \mathcal{S}} w_j), \tag{6.2}$$

When a new particle s_k is introduced, only w_k changes, while other weights remain unchanged.[1] By caching the previous entropy $\hat{H}(b)$ and the sum of weights $\sum_{s_j \in \mathcal{S}} w_j$, the entropy can be incrementally updated in $O(1)$ time, avoiding the need of recalculating from scratch. For detailed derivations of the incremental update formula, see Appendix A.4.

6.2 Incremental Computation of Boers Entropy Estimator

While Shannon entropy offers a simple and widely used measure of uncertainty, it is not well-suited for continuous state spaces, which are the focus of this work. For a detailed discussion of its limitations in such settings, see [BDBM10]. The Boers entropy estimator provides a more appropriate alternative, as it adopts a Bayesian perspective to capture dependencies between particles and to represent local densities. Moreover, it has been shown to converge to the true entropy under mild assumptions.

However, the computational cost of the Boers entropy estimator scales quadratically with the number of particles, making it impractical to compute from scratch each time the belief is updated. To address this, we present a method to incrementally update the Boers entropy estimator, significantly reducing computational overhead.

The Boers entropy estimator is defined as:

$$\hat{H}(b') = \log\left(\sum_{i=1}^{N} \mathcal{Z}(o|a, s'_i)\hat{w}_i\right) - \sum_{i=1}^{N} \hat{w}'_i \log\left(\mathcal{Z}(o|a, s'_i) \underbrace{\sum_{j=1}^{N} \mathcal{T}(s'_i|s_j, a)\hat{w}_j}_{T_i}\right) \tag{6.3}$$

where ($'$) denotes quantities associated with the posterior belief, e.g., s'_i and $\hat{w}'_i$ represent the state and normalized weight of the ith particle in the posterior belief, respectively. When the beliefs b and b' are updated with a new particle, the affected terms are denoted with a tilde, e.g., $\tilde{w}_i$, $\tilde{w}'_i$, and $\tilde{T}_i$.

While both terms in Equation 6.3 can be updated incrementally, we focus on the incremental update of the latter, as it is the computational bottleneck of the Boers

[1] We assume identical state particles are merged and their weights added.

entropy estimator. The updated $\tilde{T}_i$ for $i = 1, \ldots, N$ is:

$$\begin{aligned}\tilde{T}_i &= \sum_{j=1}^{N} \mathcal{T}(s_i|s_j, a)\tilde{w}_j + \mathcal{T}(s_i|s_{N+1}, a)\tilde{w}_{N+1} \\ &= \frac{\sum_{j=1}^{N} w_j}{\sum_{j=1}^{N+1} w_j} T_i + \mathcal{T}(s_i|s_{N+1}, a)\tilde{w}_{N+1}. \end{aligned} \tag{6.4}$$

Thus, by caching T_i and the sum of weights and reusing previously computed terms, Equation 6.4 can be updated in $O(1)$ time for $i = 1, \ldots, N$. For $i = N + 1$, the term $\tilde{T}_{N+1}$ must be computed from scratch, incurring $O(N)$ cost. However, this computation is only required once for each new particle. Overall, the Boers entropy estimator can be updated incrementally in $O(N)$ significantly reducing computational cost.

Chapter 7

Expierments

We implemented ρPOMCPOW in Julia via the POMDPs.jl framework [ESB+17].

We evaluate it on two benchmark problems: the Continuous 2D Light-Dark problem and the Active Localization problem. We compare its performance against IPFT and PFT-DPW[1], two state-of-the-art ρPOMDP solvers, under varying planning time budgets. Performance is measured as the mean return with standard error over 1000 trials.

Next, we assess the impact of incremental reward computation by comparing planning time across iterations for ρPOMCPOW with and without incremental updates.

Detailed hardware specifications and solver hyperparameters used in the experiments are provided in Appendix A.5.

7.1 Benchmark Problems

Both problems share a common structure. The agent operates in a continuous 2D environment with an uncertain initial belief. Several beacons are scattered throughout the environment, and the agent receives noisy relative pose observations from the nearest beacon, where accuracy improves with proximity. The action space consists of movement in eight directions on the unit circle, along with a "stay" action that terminates the episode. State transitions are stochastic, and each step incurs a movement cost.

The key difference between the two problems lies in their objectives. In the Light-Dark problem, the agent aims to reach a goal region, while in Active Localization, the objective is to minimize uncertainty about its position. For detailed parameters for both problems, see Appendix A.6.

Continuous 2D Light-Dark Problem

In the 2D Light-Dark problem, the agent's task is to navigate toward a goal, starting with a highly uncertain initial belief. The reward function is sparse, granting

[1] We modified PFT-DPW to support belief-dependent rewards.

a large positive reward upon termination if the agent reaches the goal region and a large penalty otherwise. To reach the goal successfully, the agent must rely on beacons to improve localization. To encourage this behavior, information gain, defined as $IG(b, b') = \hat{H}(b) - \hat{H}(b')$, is used as reward shaping, benefiting solvers that explicitly handle belief-dependent rewards. We also evaluate POMCPOW without information gain to highlight ρPOMCPOW's ability to incorporate belief-dependent rewards. Results are presented in Table 7.1.

Continuous 2D Light-Dark POMDP Scenario			
Algorithm	0.1 Seconds	0.2 Seconds	1.0 Seconds
ρPOMCPOW†	**22.3** $\pm$ 1.2	**25.9** $\pm$ 1.1	**26.2** $\pm$ 1.1
POMCPOW	17.2 $\pm$ 1.4	17.5 $\pm$ 1.4	18.5 $\pm$ 0.9
IPFT†	-2.3 ± 1.8	6.4 $\pm$ 1.7	17.2 $\pm$ 1.2
PFT-DPW†	5.3 $\pm$ 1.6	13.4 $\pm$ 1.4	20.5 $\pm$ 1.0

Table 7.1: Performance comparison for the Continuous 2D Light-Dark POMDP scenario. Algorithms marked with † use information gain as reward shaping.

Results show that ρPOMCPOW finds better solutions significantly faster. The comparison with POMCPOW highlights ρPOMCPOW's ability to utilize belief-dependent rewards and shows that while these rewards introduce expensive computations, potentially slowing down simulations, they can significantly aid in finding better policies.

Active Localization Problem

The Active Localization problem tasks the agent with minimizing uncertainty about its position. As in the Light-Dark problem, the agent starts with a highly uncertain belief and must use beacon observations for localization. However, unlike Light-Dark, obstacles are scattered throughout the environment, incurring a penalty upon collision, and distant beacons provide more informative observations, making decision-making more challenging. Figure 7.1 shows simulated trajectories resulted from each solver in this problem. Unlike Light-Dark, the reward is pure information gain, directly driving uncertainty reduction. Results are presented in Table 7.2.

Active Localization POMDP Scenario			
Algorithm	0.1 Seconds	0.2 Seconds	1.0 Seconds
ρPOMCPOW	29.0 $\pm$ 0.5	**38.1** $\pm$ 0.7	**45.9** $\pm$ 0.8
IPFT	27.1 $\pm$ 0.4	27.7 $\pm$ 0.4	27.0 $\pm$ 0.4
PFT-DPW	**36.7** $\pm$ 0.7	37.6 $\pm$ 0.7	38.7 $\pm$ 0.7

Table 7.2: Performance comparison for the Active Localization POMDP scenario.

Initially, ρPOMCPOW is outperformed by PFT-DPW but surpasses it as planning progresses. We attribute this to poor initial belief nodes that refine over time. IPFT

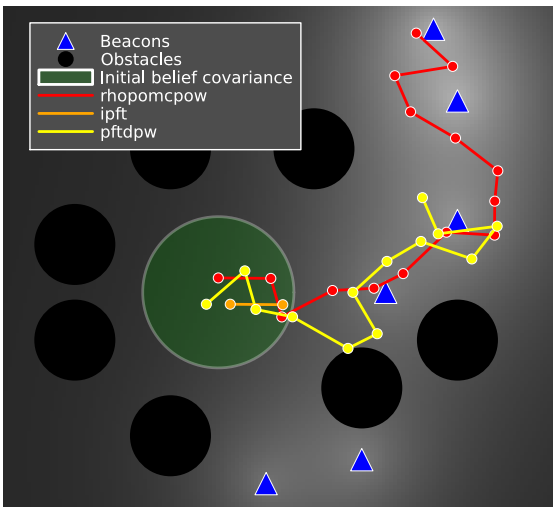

Figure 7.1: Simulated trajectories in the Active Localization problem

lags behind; likely due to slower simulation speed, due to the particle reinvigoration step, which is not implemented in ρPOMCPOW and PFT-DPW.

7.2 Effect of Incremental Reward Computation

To evaluate the advantage of incremental reward computation, we compare the planning time of ρPOMCPOW with and without incremental reward updates in the Continuous 2D Light-Dark problem. Using the same random seed and parameters for both variants ensures identical search tree expansion, isolating the impact of incremental updates on efficiency. Figure A.1 presents planning time as a function of iterations. The results reveal a significant performance gap, with full recomputation scaling at a higher order, demonstrating that incremental updates are crucial for ρPOMCPOW to be scalable.

A complexity analysis and a similar comparison with POMCPOW, assessing the cost of belief-dependent rewards, are presented in Appendix A.7.

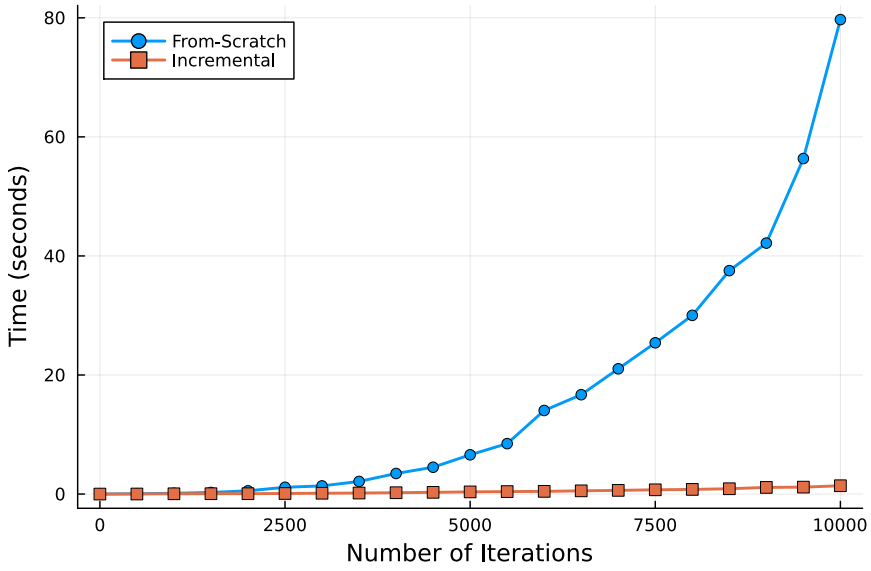

Figure 7.2: Planning time comparison for ρPOMCPOW with and without incremental reward computation as a function of iterations.

Chapter 8

Incremental Particle Filtering

In this section, we analyze the particle filter used in ρPOMCPOW, hereafter referred to as the *incremental particle filter*. We first examine the bias introduced by the incremental resampling procedure, and then discuss its implications for convergence as well as potential directions for future work.

8.1 Bias in Resampling within Incremental Particle Filters

The incremental particle filter differs from conventional particle filters in that it operates incrementally—updating the belief with new particles upon each visitation—whereas traditional particle filters process all particles in a single batch.

The primary difference lies in the resampling stage. In a standard particle filter, resampling is performed after all particles have been processed. In contrast, the incremental particle filter performs resampling incrementally: each time a new particle is added to the belief, one particle is immediately resampled and propagated to the next belief.

This induces a bias in the resampling stage, as resampling is not performed over the entire belief but rather over the subset of particles available at the time. As a result, particles introduced earlier may be resampled more frequently than newer ones, regardless of their weights.

More formally, let N_i be a random variable indicating the number of times the ith particle is resampled. In a conventional particle filter, the resampling procedure is unbiased: in expectation, each particle is resampled in proportion to its normalized weight. That is,

$$\mathbb{E}[N_i] = \hat{w}_i \cdot N,$$

where $\hat{w}_i$ is the normalized weight of the ith particle and N is the total number of particles.

Since resampling in incremental particle filters is performed incrementally, we will further denote by N_i^t the number of times the ith particle is resampled up to time t and by w_i^t the weight of particle i at time t. The resampling procedure in incremental particle filters is biased, as it does not consider the entire belief but only the particles available at time t. Thus, we have:

$$\begin{aligned}\mathbb{E}[N_i^t] &= 1\,[\text{particle } i \text{ is resampled at time } t] + \mathbb{E}[N_i^{t-1}] \\ &= \hat{w}_i^t + \mathbb{E}[N_i^{t-1}] \\ &= \hat{w}_i^t + \hat{w}_i^{t-1} + \cdots + \hat{w}_i^i\end{aligned}$$

A simple way to illustrate the bias toward earlier particles is to consider the case in which all particles have uniform weights.

8.2 Convergence of Incremental Particle Filters

Unbiased resampling is not a necessary condition for the convergence of particle filters, as shown in [Dou04], and we suspect that any bias introduced by incremental resampling may diminish as the number of particles increases. However, to the best of our knowledge, all particle filters that are proven to converge in distribution rely on unbiased resampling mechanisms. Furthermore, the bias introduced by the incremental resampling procedure may lead to suboptimal performance in practice, as it can cause the algorithm to disproportionately favor older particles over newer ones, even when the latter carry higher weights.

A trivial way to avoid this bias is to omit resampling entirely. In that case, the incremental particle filter behaves similarly to a conventional particle filter without resampling. However, resampling is a critical component of particle filtering, as it mitigates particle degeneracy and concentrates computational effort on particles with high posterior weight. Omitting it often leads to degraded performance in practice.

We leave the development of unbiased resampling mechanisms for incremental particle filters, or a formal analysis of the bias and a proof or refutation of convergence under biased resampling, to future work. We believe that incremental particle filters have potential not only in planning algorithms such as ρPOMCPOW, but also in broader applications, including asynchronous particle filtering where real-time updates are required.

Chapter 9

Anytime Density Adaptive Particle Tree

In this section, we propose a novel algorithm for online planning in ρPOMDPs, which we refer to as the *Anytime Density Adaptive Particle Tree* (ADAPT). This algorithm builds upon the principles and generalizes ρPOMCPOW.

We will first outline the motivation behind ADAPT, then present the main methodology. Empirical validation of the algorithm is left for future work, as it is not the primary focus of this thesis.

9.1 Motivation

In Section 2.1, we categorized online POMDP solvers that utilize particle filtering into two main classes: *state simulators* and *belief simulators*.

State simulators offer an *anytime* belief representation—i.e., the belief becomes more accurate as more particles are added with increased computation time. However, they are not *adaptive* to the structure or density of the belief. For example, the dimensionality of the belief is not directly considered, even though it is known that particle filtering suffers from the curse of dimensionality, requiring exponentially more particles to represent a belief as its dimensionality increases.

Belief simulators, in contrast, can be *adaptive* to the density of the belief—either through fixed hyperparameters or via mechanisms that locally and dynamically adjust the number of particles, as demonstrated in AdaOPS [WYZ+21]. However, these representations are not *anytime*: once a belief node is instantiated, it remains static and does not improve with additional computational effort.

ADAPT aims to combine the advantages of both classes of algorithms by providing an *anytime* belief representation that is also *adaptive* to the density and structure of the belief. While the above discussion applies to POMDPs in general, it is especially relevant in the context of ρPOMDPs, where the belief representation is critical for computing belief-dependent rewards and has a significant impact on runtime.

9.2 Methodology

We will now present the main components of ADAPT, which builds upon the principles of ρPOMCPOW and extends them to provide an anytime adaptive belief representation. We will first outline the algorithm, then discuss its relation to ρPOMCPOW.

Algorithm

ADAPT is an anytime online planning algorithm for ρPOMDPs that incrementally constructs a belief tree. As in standard online POMDP solvers, the tree is expanded by simulating future actions and observations from the current belief. The main `Simulate` procedure is outlined in Algorithm 9.1.

The action and observation selection steps (lines 5 and 6) are left abstract to support various strategies, such as the double progressive widening technique used in [SK18].

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). The number of particles to add is determined by the current belief's density and a tolerance parameter ε, which is computed in line 7 using the function $E(N(hao))$. An instance of `AddParticles` based on KLD-sampling [Fox03] is provided in Algorithm 9.2.

The function E embodies the anytime nature of the algorithm. It can be designed to decrease as the visitation count $N(hao)$ increases—e.g., $E(N(hao)) = 1/N(hao)^m$ for some $m > 0$—enabling the algorithm to allocate more particles to frequently visited nodes while controlling computational cost.

For newly encountered nodes, ADAPT performs a rollout to estimate the value. For previously visited nodes, it recursively calls `Simulate` to propagate value estimates through the tree.

Incremental Adaptive Sampling

To implement the `AddParticles` procedure in a principled manner, ADAPT can leverage *KLD-sampling* [Fox03], a method originally developed for adaptive particle filtering. Assuming the belief can be represented as a piecewise-constant distribution over a discretized set of bins, KLD-sampling incrementally draws particles until the Kullback–Leibler divergence (KLD) between the empirical particle distribution and the true posterior is bounded with high confidence. The method adaptively determines the number of particles required to approximate the belief up to a specified error tolerance ε *with probability* $1-\delta$, while accounting for the complexity of the distribution (i.e., its effective support size). This yields a compact yet accurate particle representation, particularly in regions of the state space with high uncertainty or multimodality.

Algorithm 9.2 outlines how KLD-sampling is applied incrementally within ADAPT to refine the belief at each visited node.

When the belief-dependent reward is entropy-based, KLD-sampling can reduce bias

Algorithm 9.1 ADAPT

```
1: procedure SIMULATE(h, d)
2:     if d = 0 then
3:         return 0
4:     end if
5:     a ← SelectAction(h)
6:     o ← SelectObservation(h, a)
7:     ε ← E(N(hao))
8:     AddParticles(hao, ε)                    ▷ update belief with new particles
9:     if o ∉ C(ha) then                                          ▷ new node
10:        C(ha) ← C(ha) ∪ {o}
11:        total ← ρ(hao) + γROLLOUT(hao, d − 1)
12:    else
13:        total ← ρ(hao) + γSIMULATE(hao, d − 1)
14:    end if
15:    N(h) ← N(h) + 1
16:    N(ha) ← N(ha) + 1
17:    Q(ha) ← Q(ha) + (total − Q(ha)) / N(ha)
18:    return total
19: end procedure
```

in the entropy estimate. Under the same assumptions used for KLD-sampling, the bias of the maximum likelihood entropy estimator is directly related to the KL divergence between the empirical and true distributions.

As shown by [Pan03],

$$H_{\text{MLE}} - H = \sum_{i=1}^{k} (p_i - p_i^N) \log p_i - D_{\text{KL}}(p^N \,\|\, p)$$

where H_{MLE} is the estimated entropy, H is the true entropy, p_i is the true bin probability, and p_i^N is the empirical estimate from N samples.

Taking expectation over samples, the first term vanishes, and the bias becomes:

$$\text{Bias}(H_{\text{MLE}}) = -\mathbb{E}[D_{\text{KL}}(p^N \,\|\, p)]$$

Since KLD-sampling ensures that the empirical distribution stays within a KL divergence of ε from the true distribution with probability $1 - \delta$, it directly bounds the entropy estimation bias. This makes it especially well-suited for accurate belief-dependent reward computation in ADAPT.

As noted in [Fox03], an alternative approach to determining the number of particles is to use the entropy of the distribution. While this method avoids the need to discretize the state space, it lacks formal theoretical guarantees. We leave the investigation of this approach within the context of ADAPT to future work.

Algorithm 9.2 Incremental KLD-sampling

procedure AddParticles(hao, ε)
  $n \leftarrow |b(hao)|$ ▷ number of particles in $b(hao)$
  $k \leftarrow |supp(b(hao))|$ ▷ number of supported bins in $b(hao)$
  repeat
    $n \leftarrow n + 1$
    sample s_j from $b(h)$
    sample s'_n from $\mathcal{T}(\cdot \mid s_j, a)$
    $w_n \leftarrow \mathcal{Z}(o \mid s_j, a, s'_n)$
    if $s'_n \notin supp(b(hao))$ **then** ▷ s'_n falls into empty bin
      $k \leftarrow k + 1$
      if $n \geq N_{min}$ **then**
        $N_{particles} \leftarrow \frac{k-1}{2\varepsilon} \left\{1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}} z_{1-\delta}\right\}^3$
      end if
    end if
  until $n \geq N_{particles}$ and $n \geq N_{min}$
end procedure

Relation to ρPOMCPOW

ADAPT builds upon the principles of ρPOMCPOW by introducing an anytime and adaptive belief representation.

Assuming that $\varepsilon \to 0$ as the number of node visits increases, the theoretical analysis presented in Chapter 5 remains applicable to ADAPT.

Furthermore, since beliefs are updated incrementally, the principles of incremental reward computation discussed in Chapter 6 also apply.

Altogether, we believe that ADAPT can serve as a powerful framework for online planning in ρPOMDPs—combining the strengths of anytime and adaptive belief representations, leveraging theoretical convergence guarantees, and utilizing incremental belief-dependent reward computation.

Chapter 10

Conclusion

In this work, we addressed the challenging problem of solving ρPOMDPs, with a particular focus on non-parametric continuous spaces where the complexity is further exacerbated. To this end, we introduced *ρPOMCPOW*, an online tree search algorithm that extends POMCPOW to support belief-dependent rewards in continuous domains.

We provided theoretical guarantees by proving the convergence of belief nodes, which in turn ensures the convergence of belief-dependent reward estimates, under a set of mild assumptions. Furthermore, we proposed an incremental reward computation framework and demonstrated its effectiveness when applied to common entropy estimators. This approach significantly reduces the computational overhead of belief-dependent rewards, thereby improving scalability to high-dimensional continuous spaces.

Empirical evaluations confirm the importance of incremental reward computation and demonstrate that *ρPOMCPOW* achieves strong performance in terms of both solution quality and computational efficiency.

While we have established that belief nodes converge to the true underlying belief, the overall convergence of the algorithm remains an open question. Nevertheless, the convergence of beliefs constitutes a necessary prerequisite for the convergence of the algorithm itself, and thus provides a solid foundation for further theoretical analysis.

We suggest further investigation of the proposed non-standard incremental particle filter, both in the context of POMDP planning and for real-time inference tasks.

Our proposed extension to ρPOMCPOW, *ADAPT*, introduces a novel approach to planning in POMDPs by dynamically adjusting belief node representations based on visitation counts and representational complexity. We plan to validate this approach through extensive empirical evaluations, comparing it against existing state-of-the-art methods across a range of benchmark domains.

Appendix A

Appendix

A.1 Incremental Last Value Update

[THB21] introduced the concept of Last-Value Update (LVU) for ρPOMDPs, which focuses on updating the value function of belief nodes based on the most recent reward estimates. To the best of our knowledge, [THB21] does not provide an incremental update mechanism for the value and action-value functions within the LVU framework.

In this section, we present an incremental update mechanism for these functions, which is crucial for efficiently handling belief-dependent rewards in ρPOMCPOW.

Incremental Update of the Value Function

[THB21] introduced value estimates for belief nodes, initialized with rollout values when $N(h) = 1$.[1] This leads to the following formula for the value function of a node h:

$$\hat{V}(h) = \frac{1}{N(h)} \left[Rollout(h) + \sum_{a \in Ch(h)} N(ha)\hat{Q}(ha) \right], \tag{A.1}$$

where $Ch(h)$ represents the set of action child nodes of node h.

Assume that at iteration t, node h is visited and action a' is selected. The updated

[1] In practice, POMCP and POMCPOW initialize visitation counts as $N(h) = 0$. To ensure consistency in state-dependent rewards, $N(hao)$ should be replaced with $N(hao) + 1$.

value function is given by:

$$\hat{V}(h;t) = \frac{1}{N(h;t)}\left[Rollout(h) + \sum_{a\in Ch(h;t)} N(ha;t)\hat{Q}(ha;t)\right] \tag{A.2}$$

$$= \frac{1}{N(h;t)}\left[Rollout(h) + \sum_{a\in Ch(h;t)\setminus a'} N(ha;t)\hat{Q}(ha;t) + N(ha';t)\hat{Q}(ha';t)\right] \tag{A.3}$$

$$= \frac{1}{N(h;t)}\left[Rollout(h) + \sum_{a\in Ch(h;t)\setminus a'} N(ha;t-1)\hat{Q}(ha;t-1) + N(ha';t)\hat{Q}(ha';t)\right] \tag{A.4}$$

$$= \frac{1}{N(h;t)}\Big[Rollout(h) + \sum_{a\in Ch(h;t-1)} N(ha;t-1)\hat{Q}(ha;t-1) - N(ha';t-1)\hat{Q}(ha';t-1) + N(ha';t)\hat{Q}(ha';t)\Big] \tag{A.5}$$

$$= \frac{N(h;t-1)}{N(h;t)}\hat{V}(h;t-1) - \frac{N(ha';t-1)}{N(h;t)}\hat{Q}(ha';t-1) + \frac{N(ha';t)}{N(h;t)}Q(ha';t). \tag{A.6}$$

Thus, the value function is incrementally updated as:

$$\hat{V}(h) \leftarrow \hat{V}(h) + \frac{1}{N(h)}\Big[N(ha')\hat{Q}(ha') - (N(ha')-1)\hat{Q}^{prev}(ha') - \hat{V}(h)\Big], \tag{A.7}$$

where $\hat{Q}^{prev}(ha')$ is the previous value of the action-value function for action a'.

This incremental approach allows the value function to be updated in $O(1)$, compared to $O(|Ch(h)|)$ for recomputing the entire sum of child nodes. This efficiency is particularly beneficial in large trees with numerous child nodes.

Incremental Update of the Action-Value Function

Similarly, the action-value function for a node ha is given by

$$\hat{Q}(ha) \leftarrow \frac{1}{N(ha)} \sum_{o\in Ch(ha)} N(hao)\left[\hat{\rho}(hao) + \gamma\hat{V}(hao)\right] \tag{A.8}$$

Where $Ch(ha)$ is the set of observation children nodes of node ha. Assume at iteration t node ha is visited and that observation o' is selected. The updated action-value

function is given by:

$$\hat{Q}(ha;t) = \frac{1}{N(ha;t)} \sum_{o \in Ch(ha;t)} N(hao;t) \left[\hat{\rho}(hao;t) + \gamma \hat{V}(hao;t)\right] \tag{A.9}$$

$$= \frac{1}{N(ha;t)} \sum_{o \in Ch(ha;t) \setminus o'} N(hao;t) \left[\hat{\rho}(hao;t) + \gamma \hat{V}(hao;t)\right] + \frac{N(hao';t)}{N(ha;t)} \left[\hat{\rho}(hao';t) + \gamma \hat{V}(hao';t)\right] \tag{A.10}$$

$$= \frac{1}{N(ha;t)} \sum_{o \in Ch(ha;t-1)} N(hao;t-1) \left[\hat{\rho}(hao;t-1) + \gamma \hat{V}(hao;t-1)\right] - \frac{N(hao';t-1)}{N(ha;t)} \left[\hat{\rho}(hao';t-1) + \gamma \hat{V}(hao';t-1)\right] + \frac{N(hao';t)}{N(ha;t)} \left[\hat{\rho}(hao';t) + \gamma \hat{V}(hao';t)\right] \tag{A.11}$$

$$= \frac{N(ha;t-1)}{N(ha;t)} \hat{Q}(ha;t-1) - \frac{N(hao';t-1)}{N(ha;t)} \left[\hat{\rho}(hao';t-1) + \gamma \hat{V}(hao';t-1)\right] + \frac{N(hao';t)}{N(ha;t)} \left[\hat{\rho}(hao';t) + \gamma \hat{V}(hao';t)\right] \tag{A.12}$$

Thus, the action-value function is incrementally updated as:

$$\hat{Q}(ha) \leftarrow \hat{Q}(ha) + \frac{1}{N(ha)} \Big[N(hao') \left[\hat{\rho}(hao') + \gamma \hat{V}(hao')\right] - (N(hao') - 1) \left[\hat{\rho}^{prev}(hao') + \gamma \hat{V}^{prev}(hao')\right] - \hat{Q}(ha) \Big]. \tag{A.13}$$

Similarly, this incremental update allows the action-value function to be updated in $O(1)$ time, compared to $O(|Ch(ha)|)$ for recomputing the entire sum of child nodes. This efficiency is particularly beneficial when the number of sampled observations is large.

A.2 Theorem 5.1

Extended Theorem

Assume the action and observation selection strategies are consistent with functions f, F and g, G, respectively. For belief tree paths h_τ^- and h_τ, the visitation counts satisfy: For a belief tree path h_τ, the visitation counts satisfy:

- For $h_\tau^- = a_{i_0} o_{j_1} a_{i_1} o_{j_2} \dots a_{i_{\tau-1}}$, with $t \geq k(i_0, j_1, \dots, i_{\tau-1})$:

$$N(h_\tau^-;t) \geq K_\tau^-(t) = F \circ \underbrace{G \circ F \circ \cdots \circ G}_{\tau-1 \text{ times}}(t) \tag{A.14}$$

- For $h_\tau = a_{i_0} o_{j_1} a_{i_1} o_{j_2} \dots a_{i_{\tau-1}} o_{j_\tau}$, with $t \geq k(i_0, j_1, \dots, i_{\tau-1}, j_\tau)$:

$$N(h_\tau;t) \geq K_\tau(t) = \underbrace{G \circ F \circ \cdots \circ G}_{\tau \text{ times}}(t) \tag{A.15}$$

$$k(i_0, \ldots, j_\tau) = \max\left\{K_0^{-1}(f(i_0)), \ldots, K_\tau^{-1}(g(j_\tau))\right\} \tag{A.16}$$

Proof of Theorem 5.1

Proof. We will prove this by induction.

- **Base Case:** For $h_0 = b_0$, the statement is trivial with $K_0(t) = t$ and $k_0 = 0$.

- **Induction Hypothesis:** Assume that for some $\tau \geq 0$, the following holds:

$$n(h_\tau; t) \geq K_\tau(t) \tag{A.17}$$

  for all $t \geq k_\tau(i_0, j_1, ..., i_{\tau-1}, j_\tau)$.

- **Induction Step:** We need to show that the hypothesis holds for $\tau + 1$.

  - For $h_{\tau+1}^- = h_\tau a_{i_\tau}$:

$$n(h_{\tau+1}^-; t) \geq F(n(h_\tau; t)) \geq F(K_\tau(t)) = K_{\tau+1}^-(t) \tag{A.18}$$

    Where the first inequality holds when $n(h_\tau; t) \geq f(i_\tau)$. Which in particular holds when $n(h_\tau; t) \geq K_\tau(t) \geq f(i_\tau)$ which is true for all $t \geq k_\tau$ and $t \geq K_\tau^{-1} f(i_\tau)$. The second inequality holds by the induction hypothesis and the monotonicity of F for all $t \geq k_\tau$. Overall, both inequalities hold for all $t \geq \max\{k_\tau(i_0, j_1, ..., i_{\tau-1}, j_\tau), K_\tau^{-1}(f(i_\tau))\} = k_{\tau+1}^-(i_0, j_1, ..., i_\tau)$.

  - For $h_{\tau+1} = h_{\tau+1}^- o_{j_{\tau+1}}$:

$$n(h_{\tau+1}; t) \geq G(n(h_{\tau+1}^-; t)) \geq G(K_{\tau+1}^-(t)) = K_{\tau+1}(t) \tag{A.19}$$

    Where the first inequality holds when $n(h_{\tau+1}^-; t) \geq g(j_{\tau+1})$. Which in particular holds when $n(h_{\tau+1}^-; t) \geq K_{\tau+1}^-(t) \geq g(j_{\tau+1})$ which is true for all $t \geq k_{\tau+1}^-(i_0, j_1, ..., i_\tau)$ and $t \geq K_{\tau+1}^{-1} g(j_{\tau+1})$. The second inequality holds by the induction hypothesis and monotonicity of G for all $t \geq k_{\tau+1}^-(i_0, j_1, ..., i_\tau)$. Overall, both inequalities hold for all $t \geq \max\{k_{\tau+1}^-(i_0, j_1, ..., i_\tau), K_{\tau+1}^{-^{-1}}(g(j_{\tau+1}))\} = k_{\tau+1}(i_0, j_1, ..., i_\tau, j_{\tau+1})$.

By induction, the hypothesis holds for all $\tau \geq 0$. ■

Algorithm A.1 Consistent Action Selection Procedure

if $\lfloor N(h)^{\alpha_a} \rfloor > \lfloor N(h) - 1 \rfloor^{\alpha_a}$ **then**
    Sample a new action a and add a new child ha
else
    Select the child ha that maximizes the score:

$$sc(ha) = \hat{Q}(ha; t) + \sqrt{N(h)^{e(d)} / N(ha)}.$$

end if

Algorithm A.2 Consistent Observation Selection Procedure

if $\lfloor N(ha)^{\alpha_o} \rfloor > \lfloor N(ha) - 1 \rfloor^{\alpha_o}$ **then**
    Sample $o \sim \mathcal{Z}(\cdot|s, a, s')$ and add a new child hao
else
    Select the least visited child hao of ha
end if

A.3 Example of consistent selection strategies

Algorithms A.1 and A.2 constitute a consistent selection strategies with the functions f, F, g and G respectively:

$$f(i) = i^{\frac{1}{\alpha_a(1-\alpha_a)}}, \qquad G(n) = \frac{n}{\lfloor n \rfloor^{\alpha_o}} - 1 \tag{A.20}$$

$$g(i) = \left\lceil (i+1)^{\frac{1}{\alpha_o}} \right\rceil, \qquad F(n) = \frac{1}{4} n^{\, e(d)\,(1-\alpha_a)}. \tag{A.21}$$

The consistency of Algorithm A.1 is established by Lemma 3 in [ACT13], while the consistency of Algorithm A.2 follows from Corollary 2 in [ACT13].

A.4 Incremental Computation of Shannon Entropy

We will demonstrate the incremental update of the Shannon entropy for a particle belief. Assume the belief is updated with a new particle s_k. Denote updated quantities with $\tilde{}$, e.g., $\tilde{b}$, $\tilde{\hat{w}}_i$, and $\tilde{w}_i$. The updated Shannon entropy is given by:

$$\hat{H}(\tilde{b}) = -\sum_{s_i \in \mathcal{S}} \tilde{\hat{w}}_i \log \tilde{\hat{w}}_i \tag{A.22}$$

$$= -\sum_{s_i \in \mathcal{S}} \frac{\tilde{w}_i}{\sum_{s_j \in \mathcal{S}} \tilde{w}_j} \log \frac{\tilde{w}_i}{\sum_{s_j \in \mathcal{S}} \tilde{w}_j} \tag{A.23}$$

$$= -\frac{1}{\sum_{s_j \in \mathcal{S}} \tilde{w}_j} \sum_{s_i \in \mathcal{S}} \tilde{w}_i \log \tilde{w}_i + \log(\sum_{s_j \in \mathcal{S}} \tilde{w}_j) \tag{A.24}$$

$$= -\frac{1}{\sum_{s_j \in \mathcal{S}} \tilde{w}_j} \sum_{s_i \in \mathcal{S}} w_i \log w_i - \frac{\tilde{w}_k \log \tilde{w}_k - w_k \log w_k}{\sum_{s_j \in \mathcal{S}} \tilde{w}_j} + \log(\sum_{s_j \in \mathcal{S}} \tilde{w}_j) \tag{A.25}$$

$$= -\frac{1}{\sum_{s_j \in \mathcal{S}} \tilde{w}_j} \sum_{s_i \in \mathcal{S}} w_i \log \frac{w_i}{\sum_{s_j \in \mathcal{S}} w_j} - \frac{\tilde{w}_k \log \tilde{w}_k - w_k \log w_k}{\sum_{s_j \in \mathcal{S}} \tilde{w}_j} + \log(\sum_{s_j \in \mathcal{S}} \tilde{w}_j) - \log(\sum_{s_j \in \mathcal{S}} w_j) \tag{A.26}$$

$$= \frac{\sum_{s_j \in \mathcal{S}} w_j}{\sum_{s_j \in \mathcal{S}} \tilde{w}_j} \hat{H}(b) - \frac{\tilde{w}_k \log \tilde{w}_k - w_k \log w_k}{\sum_{s_j \in \mathcal{S}} \tilde{w}_j} - \log(\frac{\sum_{s_j \in \mathcal{S}} w_j}{\sum_{s_j \in \mathcal{S}} \tilde{w}_j}) \tag{A.27}$$

Thus, the Shannon entropy can be incrementally updated as:

$$\hat{H}(b) \leftarrow \frac{\sum_{s_j \in \mathcal{S}} w_j}{\sum_{s_j \in \mathcal{S}} \tilde{w}_j} \hat{H}(b) - \frac{\tilde{w}_k \log \tilde{w}_k - w_k \log w_k}{\sum_{s_j \in \mathcal{S}} \tilde{w}_j} - \log(\frac{\sum_{s_j \in \mathcal{S}} w_j}{\sum_{s_j \in \mathcal{S}} \tilde{w}_j}) \tag{A.28}$$

By caching the previous entropy $\hat{H}(b)$ and the sum of weights $\sum_{s_j \in \mathcal{S}} w_j$, the Shannon entropy can be updated in $O(1)$ time. This approach avoids the need to recompute the entropy from scratch, which would require $O(N)$ time.

A.5 Experimental Details

All experiments were conducted on an 11th Gen Intel® Core™ i9-11900K CPU (3.5,GHz) with 64,GB of RAM.

To ensure a fair comparison with POMCPOW, ρPOMCPOW adopts the same action and observation selection strategies as in [SK18]. Building on the proof of Lemma 3 from [ACT13], one can show that the action selection strategy, which relies on the classical UCB algorithm, is a consistent selection strategy. We believe that the same holds—albeit in a probabilistic sense—for the observation selection strategy, but we leave the formal proof for future work.

Continuous 2D Light-Dark Problem

Because this problem has a relatively small action space, we do not use progressive widening for actions, making the parameters α_a and K_a unnecessary. We tuned ρPOMCPOW and PFT-DPW by performing a grid search over the exploration bonus C and the observation widening parameters K_o and α_o, using a seed different from the main experiment. For PFT-DPW, we additionally searched over m, the number

of particles. Due to the algorithms' similarity, the parameters found for ρPOMCPOW served as a starting point for POMCPOW, while those identified for PFT-DPW served as a starting point for IPFT.

ρPOMCPOW, IPFT and PFT-DPW used information-gain as reward shaping such that the reward is structured as:

$$\rho(b, a, b') = \mathbb{E}_{s,s'}[\mathcal{R}_s(s, a, s')] + \lambda \cdot IG(b, b') \tag{A.29}$$

with $\mathcal{R}_s(s, a, s')$ being the standard reward function defined by the problem, $\lambda = 30.0$ and $IG(b, b')$ being the information gain between the belief states b and b'.

The chosen parameters are reported in table A.1.

Algorithm	c	k_o	α_o	m
ρPOMCPOW	120	6	1/30	-
POMCPOW	100	4	1/30	-
IPFT	100	3	1/40	20
PFT-DPW	80	3	1/40	50

Table A.1: Parameters for the Continuous 2D Light-Dark POMDP Scenario

ρPOMCPOW uses a larger c, k_o, and α_o than PFT-DPW and IPFT. A possible explanation is because ρPOMCPOW is a state simulator, and therefore runs faster simulations, which allows for more exploration.

Active Localization Problem

Since both problems share common structure, we used the same parameters for the Active Localization problem as for the Continuous 2D Light-Dark problem with the only exception that now ρPOMCPOW uses a belief node initialization of 10 particles instead of a single particle.

A.6 Detailed Problems Description

Both problems share a common structure. The agent can move in 8 directions on the unit circle or choose a "stay" action, which terminates the problem. Each step incurs a cost of -1, and transitions are linear-gaussian and are defined by:

$$\mathcal{T}(\cdot|s, a) = \mathcal{N}(s + a, \Sigma_{\mathcal{T}}), \quad \Sigma_{\mathcal{T}} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \tag{A.30}$$

The agent receives noisy relative pose observations from the nearest beacon, with accuracy improving with proximity. The observation model is defined as:

$$\mathcal{Z}(\cdot|s,a,s') = \mathcal{N}(x_b - s', \Sigma_{\mathcal{Z}}), \quad \Sigma_{\mathcal{Z}} = \frac{\sqrt{2}}{2}\|x_b - s'\|_2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \Sigma_{x_b} \tag{A.31}$$

where x_b is the position of the nearest beacon, and Σ_{x_b} changes between problems. The initial belief is given by:

$$b_0 = \mathcal{N}(x_0, \Sigma_0), \quad \Sigma_0 = \begin{bmatrix} 2.5 & 0 \\ 0 & 2.5 \end{bmatrix} \tag{A.32}$$

where x_0 changes between the problems. Both problem use a discount factor, $\gamma = 0.95$.

Continuous 2D Light-Dark Problem

In the Continuous 2D Light-Dark Problem the agent taks is to reach a goal area. The agent receives a large reward of $+100$ upon termination if it is within a unit circle center around the goal or a large penalty of -100 otherwise. In this problem $\Sigma_{x_b} = 0.5 \times I$ for all beacons.

Active Localization Problem

In the Active Localization Problem the agent's goal is to minimize uncertainty about its position in a continuous 2D environment. The reward is structured as:

$$\rho(b,a,b') = \mathbb{E}_{s,s'}[\mathcal{R}_s(s,a,s')] + \lambda \cdot IG(b,b') \tag{A.33}$$

with $\mathcal{R}_s(s,a,s')$ being the standard reward function defined by the problem, $\lambda = 30.0$ and $IG(b,b')$ being the information gain between the belief states b and b'.

In this problem $\Sigma_{x_b} = \frac{0.5}{\|x_b\|_2} \times I$, meaning that beacons that are further from the origin have a lower observation noise, encouraging the agent to explore the environment. Obstacles are scattered in the environment, and the agent incurs a large penalty of -50 for collisions.

A.7 Effect of Belief-Dependent rewards

Complexity Analysis

We analyze the complexity of ρPOMCPOW with and without belief-dependent rewards to assess the advantage of incremental updates. Additionally, we examine vanilla POMCPOW to evaluate the impact of belief-dependent rewards. Let T be the iteration budget, D the tree depth, and $R(N)$ the complexity of computing the belief-dependent reward function where N is the number of particles in the belief.

Ignoring the branching of action and observation nodes, as well as rollouts, the number of particles in each node along the simulated trajectory is $O(t)$, where t is the current iteration. Since belief-dependent reward computation is the main bottleneck, the complexity of running ρPOMCPOW with belief-dependent rewards for T iterations is:

$$O(D \cdot R(1) + D \cdot R(2) + \cdots + D \cdot R(T)) = O\left(D \sum_{t=1}^{T} R(t)\right). \tag{A.34}$$

We now assume that computing the belief-dependent reward from scratch has complexity $O(N^2)$, while incremental updates have complexity $O(N)$, as is the case for the KDE and Boers entropy estimators.

Thus, the complexity of ρPOMCPOW when computing belief-dependent rewards from scratch is:

$$O\left(D \sum_{i=1}^{T} i^2\right) = O\left(D \cdot \frac{T(T+1)(2T+1)}{6}\right) = O(D \cdot T^3). \tag{A.35}$$

Similarly, with incremental belief-dependent rewards, the complexity reduces to:

$$O\left(D \sum_{i=1}^{T} i\right) = O\left(D \cdot \frac{T(T+1)}{2}\right) = O(D \cdot T^2). \tag{A.36}$$

For POMCPOW, state-dependent reward computation is $O(1)$, while the main computational bottleneck is particle resampling at each node, which takes $O(\log N)$ time.

Thus, the complexity of running POMCPOW for T iterations is:

$$O\left(D \sum_{i=1}^{T} \log i\right) = O(D \cdot T \log T). \tag{A.37}$$

Although this is a simplified analysis, it aligns with our empirical results, highlighting the need for efficient incremental updates for belief-dependent rewards. However, even with such updates, belief-dependent rewards remain the main computational bottleneck, limiting the scalability of ρPOMCPOW and other ρPOMDP solvers. A more detailed analysis is left for future work.

Empirical Runtime Comparison with POMCPOW

We ran both POMCPOW and ρPOMCPOW on the Continuous 2D Light-Dark problem. Both algorithms share the same parameters and random seeds. While ρPOMCPOW incrementally computes belief-dependent rewards, it does not use them. These factors

ensure that both algorithms construct the same search tree.

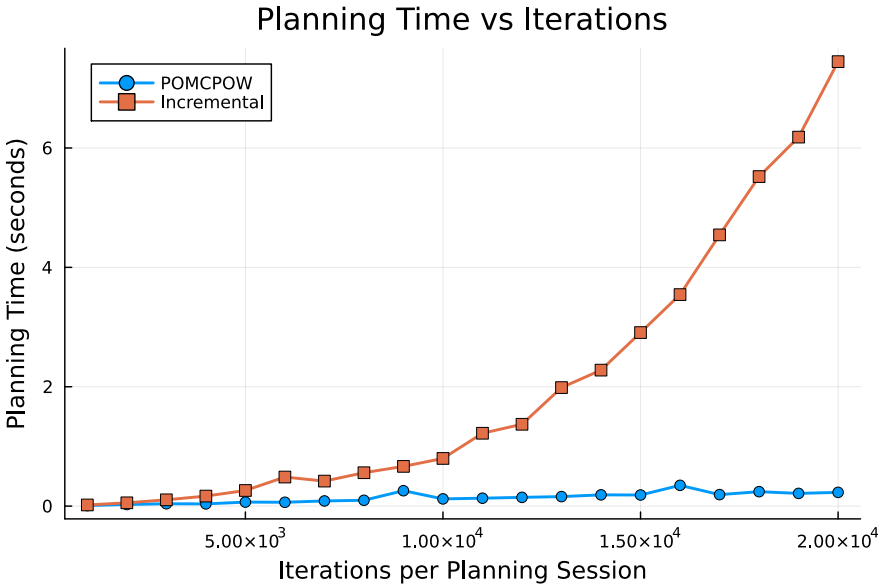

Figure A.1: Planning time comparison for POMCPOW and ρPOMCPOW with incremental reward computation as a function of iterations.

Results show that even with efficient incremental updates, belief-dependent rewards remain the primary computational bottleneck in ρPOMCPOW and other ρPOMDP solvers.

Bibliography

[ABTC10] Mauricio Araya, Olivier Buffet, Vincent Thomas, and Françcois Charpillet. A pomdp extension with belief-dependent rewards. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 64–72, 2010.

[ACT13] David Auger, Adrien Couetoux, and Olivier Teytaud. Continuous upper confidence trees with polynomial exploration–consistency. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part I 13*, pages 194–209. Springer, 2013.

[BDBM10] Y. Boers, H. Driessen, A. Bagchi, and P. Mandal. Particle filter based entropy. In *2010 13th International Conference on Information Fusion*, pages 1–8, 2010.

[BFT97] Wolfram Burgard, Dieter Fox, and Sebastian Thrun. Active mobile robot localization. In *Intl. Joint Conf. on AI (IJCAI)*, pages 1346–1352. Citeseer, 1997.

[BI22] 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.

[dCAVESM23] Matheus Aparecido do Carmo Alves, Amokh Varma, Yehia Elkhatib, and Leandro Soriano Marcolino. Information-guided planning: an online approach for partially observable problems. *Advances in Neural Information Processing Systems*, 36:69157–69177, 2023.

[Dou04] Arnaud Doucet. Sequential monte carlo methods. *Encyclopedia of Statistical Sciences*, 12, 2004.

[ESB+17] Maxim Egorov, Zachary N Sunberg, Edward Balaban, Tim A Wheeler, Jayesh K Gupta, and Mykel J Kochenderfer. Pomdps. jl: A framework for sequential decision making under uncertainty. *The Journal of Machine Learning Research*, 18(1):831–835, 2017.

[FBTD18] Mathieu Fehr, Olivier Buffet, Vincent Thomas, and Jilles Dibangoye. rho-pomdps have lipschitz-continuous epsilon-optimal value functions. *Advances in neural information processing systems*, 31, 2018.

[Fox03] Dieter Fox. Adapting the sample size in particle filters through kld-sampling. *The international Journal of robotics research*, 22(12):985–1003, 2003.

[FPM+19] Genevieve Flaspohler, Victoria Preston, Anna PM Michel, Yogesh Girdhar, and Nicholas Roy. Information-guided robotic maximum seek-and-sample in partially observable continuous environments. *IEEE Robotics and Automation Letters (RA-L)*, 4(4):3782–3789, 2019.

[FT20] Johannes Fischer and Omer Sahin Tas. Information particle filter tree: An online algorithm for pomdps with belief-based rewards on continuous domains. In *Intl. Conf. on Machine Learning (ICML)*, Vienna, Austria, 2020.

[GHL19] Neha P Garg, David Hsu, and Wee Sun Lee. Despot-α: Online pomdp planning with large state and observation spaces. In *Robotics: Science and Systems (RSS)*, 2019.

[HJDF20] Yulin He, Jie Jiang, Dexin Dai, and Klohoun Fabrice. An incremental kernel density estimator for data stream computation. *Complexity*, 2020:1–17, 02 2020.

[HK21] Marcus Hoerger and Hanna Kurniawati. An on-line pomdp solver for continuous observation spaces. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 7643–7649. IEEE, 2021.

[ICD15] V. Indelman, L. Carlone, and F. Dellaert. Planning in the continuous domain: a generalized belief space approach for autonomous navigation in unknown environments. *Intl. J. of Robotics Research*, 34(7):849–882, 2015.

[KHL08] H. Kurniawati, D. Hsu, and W. S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems (RSS)*, 2008.

[KS06] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[Pan03] Liam Paninski. Estimation of entropy and mutual information. *Neural Computation*, 15(6):1191–1253, 2003.

[PGT+03] Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. Point-based value iteration: An anytime algorithm for pomdps. In *Ijcai*, volume 3, pages 1025–1032, 2003.

[PT87] C. Papadimitriou and J. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[PTKLP10] R. Platt, R. Tedrake, L.P. Kaelbling, and T. Lozano-Pérez. Belief space planning assuming maximum likelihood observations. In *Robotics: Science and Systems (RSS)*, pages 587–593, Zaragoza, Spain, 2010.

[SK18] Zachary Sunberg and Mykel Kochenderfer. Online algorithms for pomdps with continuous state, action, and observation spaces. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 28, 2018.

[SS05] T. Smith and R. Simmons. Point-based pomdp algorithms: Improved analysis and implementation. In *Conf. on Uncertainty in Artificial Intelligence (UAI)*, pages 542–547, 2005.

[SV10] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2164–2172, 2010.

[SYHL13] Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. Despot: Online pomdp planning with regularization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 13, pages 1772–1780, 2013.

[TBF05] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT press, Cambridge, MA, 2005.

[THB21] Vincent Thomas, Geremy Hutin, and Olivier Buffet. Monte carlo information-oriented planning. *arXiv preprint arXiv:2103.11345*, 2021.

[VDBPA12] J. Van Den Berg, S. Patil, and R. Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *Intl. J. of Robotics Research*, 31(11):1263–1278, 2012.

[WYZ+21] Chenyang Wu, Guoyu Yang, Zongzhang Zhang, Yang Yu, Dong Li, Wulong Liu, and Jianye Hao. Adaptive online packing-guided search for pomdps. *Advances in Neural Information Processing Systems*, 34:28419–28430, 2021.

[YSHL17] Nan Ye, Adhiraj Somani, David Hsu, and Wee Sun Lee. Despot: Online pomdp planning with regularization. *J. of Artificial Intelligence Research*, 58:231–266, 2017.

[ZSI24] Andrey Zhitnikov, Ori Sztyglic, and Vadim Indelman. No compromise in solution quality: Speeding up belief-dependent continuous pomdps via adaptive multilevel simplification. *Intl. J. of Robotics Research*, 2024.

בעבודה זו אנו מציגים את ρPOMCPOW, אלגוריתם מקוון לפתרון ρPOMDPs המייצג את האמונה על בסיס חלקיקים ומשפר את הייצוג עם הזמן על ידי הוספת חלקיקים נוספים, ככל שמוקצה לו משאבים חישוביים נוספים anytime. אנו מראים כי תחת הנחות מתונות, שעליהן אנו דנים באריכות, האמונה שמייצג האלגוריתם מתכנסת לארוך זמן לאמונה האמיתית, וכפועל יוצא מכך גם התגמול תלוי האמונה. עם זאת, עדכון האמונה דורש את עדכון התגמול מבוסס האמונה. פעולה אשר יכולה להיות יקרה מאוד עם תחושב בצורה נאיבית.

כדי להתמודד עם מגבלה זו, אנו מציגים את גישת החישוב האינקרמנטלי incremental של תגמולים תלויי-אמונה. בשיטה זו, במקום לחשב את התגמול מחדש בכל שלב, מתבצע עדכון הדרגתי של ההערכה בהתאם לשינויים במבנה האמונה. אנו מדגימים גישה זאת על תגמולים מבוססי אנטרופיה אשר משמשים למדידת אי-ודאות או תפוסת מידע ומראים שיפור ביצועים משמעותי בכמה סדרי גודל ביחס לחישוב הנאיבי. תוצאות אמפיריות מראות שρPOMCPOW עוקף את המתחרים הן בטיב הפתרון והן ביעילות.

כדי להרחיב את יכולות האלגוריתם, הצגנו את ADAPT, הרחבה של ρPOMCPOW הכוללת התאמה דינמית של ייצוג האמונה על ידי שילוב התכנון האינקרמנטלי עם מנגנון מבוסס KLD-sampling.

ADAPT משלב מספר יתרונות מאלגוריתמים קיימים וגם משלב את תכונות ההתכנסות והחישוב האינקרמנטלי של ρPOMCPOW.

תקציר

תהליכי החלטה מרקוביים עם תצפית חלקית – (Partially Observable Markov Decision Processes (POMDPs מהווים מסגרת פורמלית חזקה לקבלת החלטות תחת אי-ודאות ומתאימים למגוון יישומים בעולם האמיתי, כגון נהיגה אוטונומית, חקירת חלל, מערכות רפואה ועוד. בPOMDP לסוכן אין גישה למצב האמיתי של הבעיה ולכן עליו לתכנן את פעולותיו על בסיס האמונה שלו (belief) שמוגדרת באופן פורמלי כהסתברות על מרחב המצבים. מטרת הסוכן היא למקסם את התוחלת על התגמול המצטבר לאורך זמן, כאשר בכל רגע נתון, ההחלטות מתקבלות ביחס לאמונה הנוכחית של הסוכן לגבי מצב העולם.

הרחבה של מסגרת זו, המכונה ρPOMDP מוסיפה תגמול תלוי אמונה לבעיה, המאפשר לסוכן להתייחס לאי-ודאות באופן ישיר. מסגרת זאת רלוונטית במיוחד בבעיות הדורשות איסוף מידע או הפחתת אי הודאות כדי להשיג מטרה. למשל, חקר רובוטי בסביבה לא ידועה או ניווט במצבים מסוכנים.

האלגוריתמים הקיימים לפתרון בעיות POMDP נחלקים לשתי קטגוריות עיקריות: אלגוריתמים מקוונים ואלגוריתמים לא-מקוונים (online and offline). אלגוריתמים לא-מקוונים פותרים את הבעיה כולה מראש, טרם ההפעלה בעולם האמיתי, ומחשבים מדיניות (policy) — פונקציה אשר בהינתן אמונה מחזירה את הפעולה שעל הסוכן לבצע. לעומתם, אלגוריתמים מקוונים פועלים בזמן אמת, תוך כדי הביצוע בפועל. לרוב, הם מבצעים פרישת עץ חיפוש מתוך האמונה הנוכחית, שבו מדמים רצפים של פעולות ותצפיות, ומעדכנים את ההערכות כלפי מעלה בהתאם לתגמולים שהתקבלו בסימולציה. בסיום כל איטרציה, האלגוריתם בוחר פעולה, הסוכן מבצע אותה, מקבל תצפית חדשה, מעדכן את האמונה שלו, וחוזר לתכנון — וחוזר חלילה, עד לסיום התהליך.

הגישה המקוונת הינה יותר סקלבילית מכיוון שהחיפוש מתמקד באמונות שניתן להגיע אליהן מהאמונה הנוכחית. לכן, לבעיות גדולות ולבעיות רציפות הגישה המקוונת עדיפה, עליה מתבסס המחקר העדכני בתכנון בבעיות POMDP ובפרט מחקר זה.

מרבית האלגוריתמים הקיימים לפתרון בעיות POMDP בצורה מקוונת מייצגים את האמונה על בסיס דגימות המכונות חלקיקים (particles). ייצוג זה מאפשר לייצג התפלגויות כלליות ומולטי-מודאליות (multi-modal)

תכנון מקוון במסגרתρPOMDP מציב אתגרים חישוביים משמעותיים, במיוחד כאשר האמונה מיוצגת על ידי חלקיקים. הבעיה קשה יותר מתכנון ב-POMDP רגיל מכיוון שהתגמול תלוי בהתפלגות על המצבים כולה ולא רק במצב בודד ולעתים יקר לחישוב.

למשל, העלות של חישוב האנטרופיה (entropy) לאמונה המיוצגת על ידי חלקיקים הינה ריבועית במספר החלקיקים.

האלגוריתמים הקיימים לפתרון ρPOMDP בצורה מקוונת מוגבלים בכך שהם מיציגים את האמונה על ידי מספר קבוע של חלקיקים או מחשבים את התגמול באופן לא יעיל. מגבלות אלו מפחיתות את השימושיות באלגוריתם אלו.

המחקר בוצע בהנחייתו של פרופסור חבר ואדים אינדלמן בפקולטה למדעי המחשב, טכניון.

מחבר חיבור זה מצהיר כי המחקר, כולל איסוף הנתונים, עיבודם והצגתם, התייחסות והשוואה למחקרים קודמים וכו', נעשה כולו בצורה ישרה, כמצופה ממחקר מדעי המבוצע לפי אמות המידה האתיות של העולם האקדמי. כמו כן, הדיווח על המחקר ותוצאותיו בחיבור זה נעשה בצורה ישרה ומלאה, לפי אותן אמות מידה.

חלק מן התוצאות בחיבור זה פורסמו כמאמרים מאת המחבר ושותפיו למחקר בכנסים ובכתבי-עת במהלך תקופת מחקר הדוקטורט של המחבר, אשר גרסאותיהם העדכניות ביותר הינן:

Ron Benchetrit, Idan Lev-Yehudi, and Vadim Indelman. Anytime adaptive planning. To be submitted, 2025.

הכרת תודה מסורה לטכניון על מימון מחקר זה.

תכנון מקוון אינקרמנטלי ב-rhoPOMDP
במרחבים רציפים

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
מגיסטר למדעים במדעי המחשב

רון בן שטרית

הוגש לסנט הטכניון — מכון טכנולוגי לישראל
תמוז התשפ"ה חיפה יולי 2025

תכנון מקוון אינקרמנטלי ב-rhoPOMDP
במרחבים רציפים

רון בן שטרית