Action-Gradient Monte Carlo Tree Search for Non-Parametric Continuous (PO)MDPs

Idan Lev-Yehudi[1], **Michael Novitsky**[1], **Moran Barenboim**[1], **Ron Benchetrit**[2] and **Vadim Indelman**[3,4]

[1]Technion Autonomous Systems Program (TASP)
[2]Faculty of Computer Science
[3]Stephen B. Klein Faculty of Aerospace Engineering
[4]Faculty of Data and Decision Sciences
Technion - Israel Institute of Technology, Haifa 32000, Israel
{idanlev, miken1990, moranbar, ronbenc}@campus.technion.ac.il, vadim.indelman@technion.ac.il,

Abstract

Online planning in continuous state, action, and observation spaces remains challenging for autonomous systems. While Monte Carlo Tree Search (MCTS) scales effectively via sampling, most continuous (PO)MDP solvers do not exploit gradient-based action optimization. We propose Action-Gradient MCTS (AGMCTS), a framework that combines global tree search with local gradient-based action refinement, while maintaining consistent value estimates. We provide three key theoretical contributions: (1) an action score gradient theorem for particle belief states; (2) the Multiple Importance Sampling (MIS) Tree that supports frequent action-branch updates by reusing prior samples without introducing estimator drift; and (3) tractable action score gradients for smooth generative models using the Area Formula. Empirical results demonstrate that AGMCTS outperforms state-of-the-art sample-based solvers in multiple challenging continuous MDP and POMDP benchmarks.

1 Introduction

Planning under uncertainty is central to AI and robotics, where state, action, and observation spaces are often continuous [Lauri *et al.*, 2022]. Markov Decision Processes (MDPs) and Partially Observable Markov Decision Processes (POMDPs) model such problems, but exact solutions are generally intractable [Papadimitriou and Tsitsiklis, 1987; Madani *et al.*, 2003], motivating efficient approximation.

Online solvers compute actions on demand from the current state or belief [Kurniawati, 2022], often via tree search such as Monte Carlo Tree Search (MCTS) [Silver and Veness, 2010] and Determinized Sparse Partially Observable Tree (DESPOT) [Ye *et al.*, 2017]. Earlier continuous-POMDP offline methods often used fixed-dimensional parametric beliefs, e.g., Gaussian beliefs [Porta *et al.*, 2006]; recent continuous MDP/POMDP online methods improve scalability via progressive widening, including Double Progressive Widening (DPW) for MDPs and POMCPOW/PFT-DPW for POMDPs [Couëtoux *et al.*, 2011; Sunberg and Kochenderfer, 2018].

Most sampling-based continuous POMDP planners keep sampled actions fixed. VG-UCT [Lee *et al.*, 2020] refines continuous-MDP actions, but does not correct value-estimate drift after action updates. Action-Gradient MCTS (AGMCTS) integrates gradient steps into global MCTS via MIS, and is the first such hybrid strategy for non-parametric continuous POMDPs with sample-based beliefs.

1.1 Contributions

Theoretical. We derive an action score gradient theorem for MDPs and POMDPs in online tree search with importance weighting. We introduce the Multiple Importance Sampling (MIS) Tree and its *action update* operation, allowing frequent gradient steps while maintaining value-estimate consistency. We also compute action score gradients tractably for smooth generative models via the Area Formula.

Algorithmic. We present AGMCTS (Algorithm 1), blending MCTS action exploration with online gradient refinement for MDPs/POMDPs under DPW or alternative continuous-action selection heuristics.

Experimental. Across continuous MDP and POMDP benchmarks, AGMCTS variants show notable gains in most domains and competitive performance overall.

1.2 Related Work

Online Optimization in MCTS. Lee *et al.* [2020] have proposed VG-UCT for integrating gradient steps into MCTS in continuous MDPs. Their Lipschitz-continuous transition and reward setting gives a deterministic-MDP convergence result; we give estimator gradients under weaker assumptions and address value-estimate drift from action updates.

Black-box optimization methods have been used to bias action selection in MCTS to more promising regions, e.g. Voronoi optimistic optimization (VOO) [Kim *et al.*, 2020] and Voronoi progressive widening (VPW) [Lim *et al.*, 2021; Hoerger *et al.*, 2024], kernel regression [Yee *et al.*, 2016] and Bayesian optimization [Morere *et al.*, 2018; Mern *et al.*,

2021] methods. These complement AGMCTS by improving action proposals rather than refinements.

(PO)MDP Action-Gradients. An action-gradient corresponds to the single-step gradient of a policy with a deterministic immediate action. Policy gradients in POMDPs typically assume stochastic policies [Baxter and Bartlett, 2001], memory-less policies [Azizzadenesheli *et al.*, 2018], or finite action spaces [Hong *et al.*, 2024], while we handle belief-dependent policies with continuous actions. Silver *et al.* [2014] studied deterministic policies in MDPs; we extend to non-parametric POMDPs.

Information Reuse in Planning. Leurent and Maillard [2020] considered information sharing in discrete MDPs by merging similar states. Novitsky *et al.* [2025] have introduced previous knowledge utilization in POMDPs via Multiple Importance Sampling (MIS) [Veach and Guibas, 1995]. We reuse samples across sibling action branches via the MIS Tree, avoiding retrieval from an external belief database. POMDP planning has used MIS for sample reuse, e.g. [Farhi and Indelman, 2021], mainly to compute observation likelihoods. AdaOps and CMCGS cluster values of similar observations/states [Wu *et al.*, 2021; Kujanpää *et al.*, 2024], whereas we recompute value estimates under updated actions.

2 Background

MDPs and POMDPs. We consider MDPs in the form $\langle \mathcal{S}, \mathcal{A}, p_T, r, \gamma, L, b_0 \rangle$. The state and action spaces are $\mathcal{S} \subseteq \mathbb{R}^{n_s}$ and $\mathcal{A} \subseteq \mathbb{R}^{n_a}$. The transition model $p_T(s'|s,a)$ is the probability of arriving at state s' when taking action $a \in \mathcal{A}$ at state $s \in \mathcal{S}$. The reward function $r(s,a,s') \in \mathbb{R}$ gives the immediate reward of transitioning from state s to s' by taking action a, and the expected reward is $r(s,a) \triangleq \mathbb{E}_{s'|s,a}[r(s,a,s')]$. The discount factor is $\gamma \in (0,1]$. The MDP starts at time 0 and terminates after $L \in \mathbb{N} \cup \{\infty\}$ steps, and if $L = \infty$ then we assume $\gamma < 1$. The initial state is drawn from the distribution $s_0 \sim b_0$.

We assume $\pi = (\pi_t)_{t=0}^{L}$ is a (possibly time-dependent) deterministic policy, but we note our theoretical results generalize to stochastic policies as well. The value function of a policy π at time t is the expected sum of discounted rewards until the horizon: $V_t^\pi(s_t) \triangleq \mathbb{E}_{s_{t+1:L}|s_t,\pi}[\sum_{i=t}^{L} \gamma^{i-t} r(s_i, \pi_i(s_i), s_{i+1})]$. We denote next-step variables with primes, e.g. s' follows s. We define the action-value function as $Q_t^\pi(s,a) \triangleq \mathbb{E}_{s'|s,a}[r(s,a,s') + \gamma V_{t+1}^\pi(s')]$. Our goal is to compute a policy that maximizes the value function, i.e. find $\pi^* \in \arg\max_\pi V_0^\pi(s_0)$.

A POMDP adds $\langle \mathcal{O}, p_O \rangle$ to the MDP tuple. The observation space is $\mathcal{O} \subseteq \mathbb{R}^{n_o}$, and the observation model $p_O(o|s)$ is the conditional probability of receiving an observation $o \in \mathcal{O}$ at $s \in \mathcal{S}$. Similarly to Sunberg and Kochenderfer [2018], we assume that we can both sample from and evaluate $p_O(o|s)$.

A history at time t is defined as a sequence of b_0, followed by actions taken and observations received until t: $H_t \triangleq (b_0, a_0, o_1, \ldots, a_{t-1}, o_t)$. In partially observable settings, the agent has to maintain a probability distribution of the current state given past actions and observations, known as the belief. The belief at time t is $b_t(s_t) \triangleq p(s_t|H_t)$. We define $H_t^- \triangleq (b_0, a_0, o_1, \ldots, a_{t-1})$, i.e. the history until time t without the last measurement, and correspondingly the propagated belief $b_t^-(s_t) \triangleq p(s_t|H_t^-)$. It has been shown that optimal decision-making can be made given the belief, instead of considering the entire history [Kaelbling *et al.*, 1998], meaning a POMDP is an MDP on the belief space [Åström, 1965]. Policies and value functions extend from MDPs to POMDPs by equivalent definitions on beliefs as states.

Importance Sampling. Importance Sampling (IS) [Kloek and Van Dijk, 1978] is a technique in Monte-Carlo (MC) estimation where a proposal distribution q is used to generate samples. The IS estimator for $g(x) = \mathbb{E}_{x\sim p}[f(x)]$ is $\hat{g}_q = N^{-1} \sum_{i=1}^{N} \rho_q^p(x^i) f(x^i)$, for importance ratios $\rho_q^p(x) \triangleq p(x)/q(x)$. The IS estimator $\hat{g}_q$ is unbiased if $q(x) = 0$ implies $p(x) = 0$. Often, the self-normalized IS (SNIS) estimator is formulated when we only have access to an unnormalized version of p, or for variance reduction purposes, and is given by $\tilde{g}_q = (\sum_{i=1}^{N} \rho_q^p(x^i))^{-1} \sum_{i=1}^{N} \rho_q^p(x^i) f(x^i)$. While biased, under weak assumptions it is consistent [Owen, 2013, 9.2]. We denote SNIS estimators throughout with $\tilde{[\cdot]}$.

MIS [Veach and Guibas, 1995] is a method to leverage several sampling methods $\{q_i\}_{i=1}^{n}$, by calculating a weighted average of IS estimators with n_i samples each: The corresponding MIS estimator is $\hat{g}_{\text{MIS}} = \sum_{i=1}^{n} n_i^{-1} \sum_{j=1}^{n_i} w_i(x^{i,j}) \rho_{q_i}^p(x^{i,j}) f(x^{i,j})$. If the weights w_i satisfy requirements: (MIS-I) $\sum_{i=1}^{n} w_i(x) = 1$ whenever $f(x) \neq 0$; (MIS-II) $w_i(x) = 0$ whenever $q_i(x) = 0$; then $\hat{g}_{\text{MIS}}$ is unbiased. Many weighting strategies exist; the balance heuristic $w_i(x) = n_i q_i(x) / \sum_k n_k q_k(x)$, has its variance provably bounded from the optimal weighting strategy for independent samples. Analogously to SNIS, self-normalized MIS (SN-MIS) can be defined as $\tilde{g}_{\text{MIS}} = \beta^{-1} \sum_{i=1}^{n} \sum_{j=1}^{n_i} w_i(x^{i,j}) \rho_{q_i}^p(x^{i,j}) f(x^{i,j})$ for $\beta = \sum_{i=1}^{n} \sum_{j=1}^{n_i} w_i(x^{i,j}) \rho_{q_i}^p(x^{i,j})$ [Metelli *et al.*, 2020].

Particle Beliefs. Particle filters are often used to represent a belief non-parametrically [Doucet *et al.*, 2001, 1.3.2]. We denote particle beliefs with $\bar{b}$. The particle belief of J particles is the ordered set of state-weight pairs $\bar{b} = ((s^j, \lambda^j))_{j=1}^{J}$ (following [Lim *et al.*, 2023]), and is defined as the discrete distribution $p(s|\bar{b}) = \sum_{j=1}^{J} \lambda^j \cdot \delta(s - s^j) / \sum_{j=1}^{J} \lambda^j$. For computational simplicity, we assume throughout the paper that the bootstrap filter is used to update particle beliefs [Doucet *et al.*, 2001, 1.3.2]. At each time step, the bootstrap filter advances sampled particles using the transition model, reweights them based on the observation model, and resamples to avoid weight degeneracy [Kong *et al.*, 1994].

MCTS and DPW. MCTS is an algorithm used to quickly explore large state spaces [Browne *et al.*, 2012]. It iteratively repeats four steps to build a search tree that approximates the action-values, using a best-first strategy: (1) *Selection*: Starting from the root node, descend recursively until a node to which children can be added is found; (2) *Expansion*: A new node is added as a child, according to an action expansion strategy; (3) *Simulation*: A simulation is run from the new node according to the rollout (default) policy; (4) *Backpropagation*: The simulation result is backpropagated through the

selected nodes to update the action-values. Often UCT [Kocsis and Szepesvári, 2006] is used at *selection* to balance between exploration of new actions and exploitation of promising ones. Double Progressive Widening (DPW) [Couëtoux *et al.*, 2011] is a technique to limit the branching factor from being infinite in continuous settings. The number of children of a node is artificially limited to kN^α for N visitations, for fixed $k > 0$ and $0 < \alpha < 1$.

3 Action Gradients and MIS Trees

Here we present the theoretical basis of AGMCTS. We derive action score gradients for local refinement, introduce MIS Trees for consistent value estimates, and discuss density computation for smooth black-box simulators.

3.1 (PO)MDP Action Score Gradients

In POMDPs the posterior belief transition probability $p(b'|b,a)$ is generally intractable. We approach the belief update structure via the propagated belief density:

$$p(b'|b,a) = \int p(b'|b'^-, o')p(o'|b'^-)p(b'^-|b,a)\,\mathrm{d}b'^-\,\mathrm{d}o'. \quad (1)$$

The propagated belief b'^- is obtained after action a but before observation o'. Exact beliefs make $p(b'^-|b,a)$ a Dirac delta because b'^- is deterministic in (b,a); for particle beliefs $\bar{b}, \bar{b}'^-$ it is non-degenerate because particles are sampled through stochastic transitions:

Lemma 1. *Let $\bar{b} = ((s^j, \lambda^j))_{j=1}^J$, $\bar{b}'^- = ((s'^{-,j}, \lambda'^{-,j}))_{j=1}^J$ be ordered particle beliefs. The probability that $\bar{b}'^-$ is a propagated belief in the bootstrap filter, given $\bar{b}$ and action a, is:*

$$p(\bar{b}'^-|\bar{b},a) = \textstyle\prod_{j=1}^J p_T(s'^{-,j}|s^j,a), \quad (2)$$

whenever $\lambda'^{-,j} = \lambda^j$ for all $j = 1,\dots,J$, and zero otherwise. When $p(\bar{b}'^-|\bar{b},a) > 0$ it holds that:

$$\nabla_a \log p(\bar{b}'^-|\bar{b},a) = \textstyle\sum_{j=1}^J \nabla_a \log p_T(s'^{-,j}|s^j,a). \quad (3)$$

We use this to introduce importance ratios over the future propagated belief $\bar{b}'^-$ rather than the posterior belief $\bar{b}'$.

In tree search, after updating action a to $\breve{a}$, we estimate $Q_t^\pi(s,\breve{a})$ by reusing samples from $Q_t^\pi(s,a)$ instead of recomputing the subtree. Because repeated updates create samples from multiple proposals, we use IS/MIS. Equations (4)-(5) give the single-proposal action-reuse form:

$$Q_t^\pi(s,\breve{a}) = \mathbb{E}_{s'|q}[\rho_q^{p_T}(s')(r(s,\breve{a},s') + \gamma V_{t+1}^\pi(s'))], \quad (4)$$

$$Q_t^\pi(\bar{b},\breve{a}) = \mathbb{E}_{\bar{b}'^-,o',\bar{b}'|q}[\rho_q^p(\bar{b}'^-)(r(\bar{b},\breve{a},\bar{b}') + \gamma V_{t+1}^\pi(\bar{b}'))], \quad (5)$$

where $\rho_q^{p_T}(s') \triangleq p_T(s'|s,\breve{a})/q(s'|s,\breve{a})$, and $\rho_q^p(\bar{b}'^-) \triangleq p(\bar{b}'^-|\bar{b},\breve{a})/q(\bar{b}'^-|\bar{b},\breve{a})$ are the importance ratios. The ratios require proposal support to cover target support. We use $q(s'|s,\breve{a}) := p_T(s'|s,a)$, reusing samples from the previous action a, but generally other choices are possible too. In Section 3.2, we extend Equations (4)-(5) to an MIS setting with multiple proposals, where assumptions (MIS-I) and (MIS-II) suffice for unbiasedness (Section 2).

We introduce our main results of action score gradients.

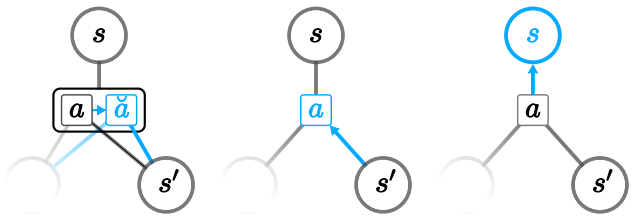


Figure 1: Update operations in the MIS Tree, left to right, highlighted in cyan: (1) *Action update*, a novel operation, computes a consistent action-value estimate for a new action $\breve{a}$ reusing previous successor states; (2) *Action backpropagation*; (3) *State backpropagation*.

Theorem 1. *Assume: (i) $\rho_q^{p_T}(s')$, $\rho_q^p(\bar{b}'^-)$ are well-defined; (ii) r, V are bounded; (iii) there exist integrable g_r, g_T w.r.t. the MDP/POMDP measures such that $\|\nabla_a r\| \le g_r$ and $\|\nabla_a \log p_T\| \le g_T$. For POMDPs, assume the assumptions of Lemma 1 hold. Then, the action score gradient satisfies:*

$$\nabla_{\breve{a}} Q_t^\pi(s,\breve{a}) = \mathbb{E}_{s'|q}[\rho_q^{p_T}(s')[\nabla_{\breve{a}} \log p_T(s'|s,\breve{a}) \\ (r(s,\breve{a},s') + \gamma V_{t+1}^\pi(s') - B(s)) + \nabla_{\breve{a}} r(s,\breve{a},s')]], \quad (6)$$

$$\nabla_{\breve{a}} Q_t^\pi(\bar{b},\breve{a}) = \mathbb{E}_{\bar{b}'^-,o',\bar{b}'|q}[\rho_q^p(\bar{b}'^-)[\nabla_{\breve{a}} \log p(\bar{b}'^-|\bar{b},\breve{a}) \\ (r(\bar{b},\breve{a},\bar{b}') + \gamma V_{t+1}^\pi(\bar{b}') - B(\bar{b})) + \nabla_{\breve{a}} r(\bar{b},\breve{a},\bar{b}')]], \quad (7)$$

where $B(s)$ (resp. $B(\bar{b})$) is any baseline independent of $\breve{a}$ for estimator variance reduction [Schulman et al.*, 2016], e.g. $B(s) = V_t^\pi(s)$.*

The proof follows policy-gradient arguments [Sutton *et al.*, 1999]. The result allows computing action gradients with samples taken from the proposal q. The gradient favors immediate reward and likely high-value successors; since s' is integrated out, $V_{t+1}^\pi(s')$ has no direct $\breve{a}$-gradient. Compared with recursive estimates like VG-UCT [Lee *et al.*, 2020], these score gradients are branch-local in the MIS Tree, and apply to belief-dependent policies via Equation (1).

3.2 MIS Trees for Action-Adaptive MCTS

We introduce the MIS Tree, a novel search-tree whose action-value estimates are MIS estimators. It supports standard backpropagation plus *action update*: replacing an action label while reusing previous successor states for a consistent updated action-value estimate. Figure 1 illustrates these operations. We formulate the MIS Tree recursively over state nodes s and action nodes (s,a). For POMDPs, we use a particle-filter tree (PFT) [Sunberg and Kochenderfer, 2018], replacing s by ordered particle belief states $\bar{b}$ throughout.

Each state node s has a set of child actions $\mathcal{A}_s \triangleq \{a^i\}_i$. Each action node (s,a) has a set of child successor states $\mathcal{S}_{sa} \triangleq \{s'^{,i}\}_i$. The visitation counts of state and action nodes are $n(s)$ and $n(s,a)$. We store for each successor $s'^{,i}$ the proposal action a^i_{prop}, namely the action under which $s'^{,i}$ was originally sampled. This stored proposal action may differ from the current action label a after action updates. We also store the value estimate $\hat{V}(s'^{,i})$, defined by:

$$\hat{V}(s) \triangleq n(s)^{-1} \textstyle\sum_{i=1}^{|\mathcal{A}_s|} n(s,a^i)\hat{Q}(s,a^i). \quad (8)$$

Algorithm 1 AGMCTS. Novel components are highlighted in blue.

procedure SIMULATE(s, d)
if $d = 0$ **then**
  $v \leftarrow$ ROLLOUT(s) {*Rollout until terminal state*}
  UPDATETERMINAL(s, v) {*Eq.* (17)}
  return v
$a \leftarrow$ ACTIONPROGWIDEN(s) {*Vanilla or VPW*}
$addBranch \leftarrow$ ACTIONOPT (s, a, d)
if $|\mathcal{S}_{sa}| \leq k_o \cdot n(s,a)^{\alpha_o}$ OR $addBranch$ **then**
  $s', r \sim G(s, a)$ {$\bar{b}'^{-}, \bar{b}', r \sim G(\bar{b}, a)$ *in POMDPs*}
  $\mathcal{S}_{sa} \leftarrow \mathcal{S}_{sa} \cup \{s'\}$
  $v \leftarrow$ ROLLOUT$(s', d-1)$
else
  $s' \sim$ Unif$(\mathcal{S}_{sa})$, $r \leftarrow$ REWARD(s, a, s')
  $v \leftarrow$ SIMULATE$(s', d-1)$
  UPDATEMIS(s, a, s', r) {*Eqs.* (15), (16), (18)}

procedure ACTIONOPT(s, a, d)
$addBranch \leftarrow$ FALSE
for all $k = 1, \ldots, K_{\text{opt}}$ **do**
  $g_a^q \leftarrow \hat{\nabla}_a \tilde{Q}(s, a)$ {*Eq.* (21)}
  $\breve{a} \leftarrow$ OPT(s, a, g_a^q) {*Adam/other*}
  $\breve{a} \leftarrow$CLIPNORM$(\breve{a}, T_{d_a}^{\max})$
  ACTIONUPDATE$(s, a, \breve{a})$ {*Eq.* (11)-(14), (18)}
  $addBranch \leftarrow addBranch \vee (\forall s'^{,i} \in \mathcal{S}_{s\breve{a}} : \rho_{\breve{a}}^i(s'^{,i}) \leq T_\rho^{add})$
return $addBranch$

The visitation counts satisfy:

$$n(s) = \sum_{i=1}^{|\mathcal{A}_s|} n(s, a^i), \quad n(s,a) = \sum_{i=1}^{|\mathcal{S}_{sa}|} n(s'^{,i})_{+1}. \quad (9)$$

Here, $n(s')_{+1} \triangleq n(s') + 1$ is the post-expansion count contribution of successor s'. The $+1$ accounts for the initial rollout value stored when the successor node is created, while $n(s')$ counts later visits. For action nodes, $\hat{Q}(s,a) \triangleq \hat{r}(s,a) + \gamma \hat{V}_f(s,a)$, where $\hat{r}$ estimates immediate reward and $\hat{V}_f$ estimates next-step value. Our key observation is that this decomposition lets *action update* recompute $\hat{V}_f$ from older successor samples and their stored value estimates.

Generally in MIS, each proposal distribution q_i has n_i samples. In our situation, we assume that each $s'^{,i} \in \mathcal{S}_{sa}$ was sampled from $p_T(\cdot|s, a^i_{\text{prop}})$, with its associated single value estimate $V(s'^{,i})$. Therefore, we adapt the MIS definition to combine successor-specific estimators, based on the associated proposal action, branch importance ratio $\rho_a^i(s')$ and MIS weights $w_i(s')$:

$$\hat{V}_f(s,a) \triangleq \sum_{i=1}^{|\mathcal{S}_{sa}|} w_i(s'^{,i}) \rho_a^i(s'^{,i}) \hat{V}(s'^{,i}), \quad (10)$$

$$\rho_a^i(s') \triangleq p_T(s'|s,a) / p_T(s'|s, a^i_{\text{prop}}), \quad (11)$$

and for POMDPs $\rho_a^i(\bar{b}') \triangleq p(\bar{b}'^{-}|\bar{b}, a) / p(\bar{b}'^{-}|\bar{b}, a^i_{\text{prop}})$. Analogously, replace $\hat{V}(s'^{,i})$ with $r(s,a,s'^{,i})$ to obtain $\hat{r}(s,a)$.

The following theorem shows that the MIS conditions suffice for unbiasedness of the MIS Tree:

Theorem 2. *If $w_i(s')$ satisfy requirements (MIS-I) and (MIS-II) (see Section 2), then the MIS Tree yields unbiased value and action-value estimates for a given tree structure.*

The MIS balance heuristic requires $O(|\mathcal{S}_{sa}|)$ density evaluations per recursive sample update [Novitsky *et al.*, 2025]. For efficiency, we use self-normalized MIS (SN-MIS) [Metelli *et al.*, 2020] with naive weights $w_i(s') \propto n(s'^{,i})_{+1}$, which we found to perform well in practice:

$$\tilde{V}_f(s,a) \triangleq \sum_{i=1}^{|\mathcal{S}_{sa}|} n(s'^{,i})_{+1} \rho_a^i(s'^{,i}) \hat{V}(s'^{,i}) / \eta(s,a), \quad (12)$$

$$\eta(s,a) \triangleq \sum_{i=1}^{|\mathcal{S}_{sa}|} n(s'^{,i})_{+1} \rho_a^i(s'^{,i}). \quad (13)$$

Using SN-MIS, all MCTS operations are $O(1)$ except $O(|\mathcal{S}_{sa}|)$ *action update*. The Appendix gives full derivations, numerically stable log-likelihood forms, and analogous immediate-reward updates.

Action Update. Action update replaces (s,a) by $(s,\breve{a})$, recomputes all ratios, (12), (13), and immediate rewards in $O(|\mathcal{S}_{sa}|)$ time.

In POMDPs, the explicit computation of the new transition likelihood $p(\bar{b}'^{-}|\bar{b}, \breve{a})$ via (2) requires J evaluations of the transition density, for J particles in $\bar{b}$ and $\bar{b}'^{-}$, which may be computationally expensive. As $p(\bar{b}'^{-}|\bar{b}, \breve{a})$ is a product of probabilities, a direct MC estimate based on a subset of particles results in a large bias. We resort to estimating changes in $\log p(\bar{b}'^{-}|\bar{b}, \breve{a})$. For small $\|\breve{a} - a\|$ we approximate[1]

$$\log \rho^i_{\breve{a}}(\bar{b}'^{-}) - \log \rho^i_a(\bar{b}'^{-}) \approx (\nabla_a \log \rho^i_a(\bar{b}'^{-}))^T \delta a$$
$$= (\nabla_a \log p(\bar{b}'^{-}|\bar{b}, a))^T \delta a = (\text{Eq. (3)})^T \delta a, \quad (14)$$

and due to Equation (3) being a sum, it admits an unbiased MC estimate by subsampling particles.

Action Backpropagation. Let $s'^{,i}$ be an updated node. Hence, we updated $n(s'^{,i})$ to $n'(s'^{,i})$ and $\hat{V}(s'^{,i})$ to $\hat{V}'(s'^{,i})$. We update $n(s,a)$ by (9) and perform

$$\eta'(s,a) = \eta(s,a) + \rho_a^i(s'^{,i})(n'(s'^{,i}) - n(s'^{,i})), \quad (15)$$

$$\tilde{V}'_f(s,a) = (\eta'(s,a))^{-1} \big(\eta(s,a) \tilde{V}_f(s,a) + \rho_a^i(s'^{,i})(n'(s'^{,i}) \hat{V}'(s'^{,i}) - n(s'^{,i}) \hat{V}(s'^{,i})) \big). \quad (16)$$

Assuming a general $n'(s')$ also supports cases where we delete branches in the tree, i.e. $n'(s') < 1$. For *state expansion*, the same equations apply by initializing $n(s') = 0$.

State Backpropagation. Let s be a state node. If s is at the maximum tree depth, $\hat{V}(s)$ is based only on rollouts. We update the running average with the new rollout value v',

$$\hat{V}'(s) = \hat{V}(s) + (n'(s) - n(s))(v' - \hat{V}(s)) / n'(s)_{+1}, \quad (17)$$

If s is not a terminal state, let its recently updated action-value child be (s,a), with updated $n'(s,a)$ and $\tilde{Q}'(s,a)$. We update $n'(s)$ by (9) and perform

$$\hat{V}'(s) = (n'(s))^{-1} (n(s) \hat{V}(s) + n'(s,a) \tilde{Q}'(s,a) - n(s,a) \tilde{Q}(s,a)). \quad (18)$$

[1] This is obtained by the first-order approximation $f(x + \delta x) \approx f(x) + \nabla f(x)^T \delta x$, applied to $\log \rho_a^i(s')$ where $\delta a = \breve{a} - a$.

3.3 Densities of Smooth Generative Models via the Area Formula

Thus far we have assumed computable $p_T(s'|s,a)$ and $\nabla_a \log p_T(s'|s,a)$. In practice, many simulators provide instead $s' = f(s,a,\xi)$ with noise $\xi \sim p_\xi(\cdot|s,a)$. If $n_\xi = n_s$ and $\xi \mapsto f(s,a,\xi)$ is bijective and differentiable, the change-of-variables formula states that for $s' = f(s,a,\xi^*)$:

$$p_T(s'|s,a) = p_\xi(\xi^*|s,a)\,|D_\xi f(s,a,\xi^*)|^{-1}, \qquad (19)$$

where $D_\xi f$ is the Jacobian matrix of f w.r.t. ξ. However, it is often the case that $n_\xi < n_s$ and $\xi \mapsto f(s,a,\xi)$ is an embedding into an n_ξ-dimensional manifold rather than a bijection. Then $p_T(s'|s,a)$ is a density with respect to the induced manifold measure, not Lebesgue measure, and the Area Formula [Federer, 1969, Theorem 3.2.3] applies. Negro [2022] gives the following locally Lipschitz form for f and the induced density p_T:

Theorem 4.1. *[Negro, 2022] If f is locally Lipschitz, and it holds that $\operatorname{rank}(D_\xi f) = n_\xi$ a.e., then the induced probability measure over s' is absolutely continuous w.r.t. the Hausdorff measure $\mathcal{H}^{n_\xi}$ on $\mathbb{R}^{n_s}$, and its Radon-Nikodym derivative is*

$$p_T(s'|s,a) = \textstyle\sum_{\Xi^*} p_\xi(\xi^*|s,a)(J_{n_\xi} f(s,a,\xi^*))^{-1}, \qquad (20)$$

for $\Xi^* = \{\xi^* : s' = f(s,a,\xi^*)\}$, and 0 when $\Xi^* = \emptyset$. The n_ξ-dimensional Jacobian is given by the Cauchy-Binet formula: $J_{n_\xi} f(s,a,\xi) = \sqrt{\det((D_\xi f)^\intercal \cdot (D_\xi f))}$.

Robotics simulators often integrate continuous-time dynamics with stochasticity injected at the input or output. In both cases below, once p_T is computable, $\nabla_a \log p_T$ follows by chain rule and automatic differentiation.

Input Noise to Simulator. Let the input noise be described by the function $h(s,a,\xi) = (\tilde{s},\tilde{a})$. The simulator then produces a successor state via $s' = f(\tilde{s},\tilde{a})$. If for a new action $\breve{a}$ there exists a unique ξ^* satisfying $h(s,\breve{a},\xi^*) = (\tilde{s},\tilde{a})$, i.e. such that the input to the simulator remains unchanged, then we can evaluate the sensitivity of the output state with respect to the noise: $D_\xi f(s,\breve{a},\xi^*) = D_{(\tilde{s},\tilde{a})} f \cdot \frac{\partial h}{\partial \xi}|_{s,\breve{a},\xi^*}$. Moreover, the Jacobian matrix $D_{(\tilde{s},\tilde{a})} f$ can be cached from the original sample (s,a,ξ,s'), allowing to compute the density without requiring additional simulator queries.

Noise in Simulator Output. Let the simulator output be $h(s,a)$, and the next state be given by $s' = g(h(s,a),\xi)$. If we can find all ξ^* such that $s' = g(h(s,\breve{a}),\xi^*)$, then we compute the Jacobian via $D_\xi f(s,\breve{a},\xi^*) = \frac{\partial g}{\partial \xi}|_{h(s,\breve{a}),\xi^*}$, requiring a single forward simulation $h(s,\breve{a})$ for all ξ^*.

4 AGMCTS Algorithm

AGMCTS combines MCTS with frequent action-gradient optimization, and correcting online value estimates via MIS. It uses a forward simulator $G(s,a)$ with assumed access to p_T and p_O, controls branching via DPW, and extends to POMDPs via ordered particle-belief states like PFT-DPW [Sunberg and Kochenderfer, 2018]. Algorithm 1 keeps the standard simulation loop; blue components add ACTIONOPT and MIS *action update* before expansion so gradients influence expansion. The Appendix details implementation.

Monte Carlo Gradient Estimation. In POMDPs, we estimate (3) with K_b^∇ state particles via $\nabla_a \log p(\bar{b}'^-|\bar{b},a) \approx (J/K_b^\nabla)\sum_{l=1}^{K_b^\nabla} \nabla_a \log p_T(s^{-,j_l}|s^{j_l},a)$, where indices j_l are sampled uniformly from $[1,\dots,J]$, and J is the particle count of $\bar{b}$ and $\bar{b}'^-$. We choose a baseline function of the current value estimate $B(s) = \hat{V}(s)$ ($B(\bar{b}) = \hat{V}(\bar{b})$ in POMDPs), effectively estimating advantage gradients. As we linearize the weight updates as in Equation (14), $\nabla_a \rho_a^i(s'^{,i})$ has to be computed for all $s'^{,i} \in \mathcal{S}_{sa}$. To save computations, we sum over all successor states $s' \in \mathcal{S}_{sa}$ when computing $\hat{\nabla}\tilde{Q}(s,a)$:

$$\hat{\nabla}_a \tilde{Q}(s,a) = \textstyle\sum_{i=1}^{|\mathcal{S}_{sa}|} n(s'^{,i})_{+1}\rho_a^i(s'^{,i}) \cdot (\nabla_a \log p_T(s'^{,i}|s,a) \cdot (r(s,a,s'^{,i}) + \gamma\hat{V}(s'^{,i}) - B(s)) + \nabla_a r(s,a,s'^{,i}))/\eta(s,a), \qquad (21)$$

and we cache $\nabla_a \log p_T(s'^{,i}|s,a)$ for later *action updates*.

Gradient Optimization. Adam's step-size normalization is crucial under high-variance gradients [Kingma and Ba, 2015]. We perform K_{opt} consecutive gradient updates to control the accuracy-compute trade-off, and gradient clipping of $\|\breve{a} - a\|$ to $T_{d_a}^{\max}$ to stabilize updates.

Threshold for Adding Successor Nodes. If $\rho_{\breve{a}}^i(s'^{,i}) < T_\rho^{add}$ for all $s'^{,i} \in \mathcal{S}_{s\breve{a}}$, we force a new successor sample, detailed in the Appendix with optional low-relevance deletion.

Action Sampling Heuristics. We use uniform action sampling and Voronoi progressive widening (VPW) [Lim *et al.*, 2021]; VPW biases samples through Voronoi partitions, demonstrating that AGMCTS can benefit from different action proposal mechanisms.

5 Experiments

We evaluate AGMCTS on continuous MDP/POMDP benchmarks. MDP baselines are DPW/VPW; POMDP baselines are PFT-DPW/PFT-VPW and POMCPOW/VOMCPOW from POMDPs.jl [Egorov *et al.*, 2017]. AGMCTS also uses POMDPs.jl. VPW and result analysis follow the codebase of [Lim *et al.*, 2021]; the Appendix gives full domain details.

***d*D-Continuous Light-Dark POMDP.** This domain generalizes the Light-Dark POMDP [Platt *et al.*, 2010] to arbitrary dimension d, combining an accuracy-sensitive continuous reward, a non-Gaussian spherical initial belief, and a state-dependent nonuniform observation density.

Here $\mathcal{S} = \mathcal{O} = \mathbb{R}^d$, $b_0 = S_{0.5}^{d-1}$, the goal lies above the origin, and the beacon lies to the right. The episode ends on entering the goal zone ($T_{\text{goal}} = 0.2$) or after $L = 6$ steps. Actions satisfy $\|a\| \le 1.5$ and transitions are linear with additive Gaussian noise $\mathcal{N}(0,(0.025)^2\mathbf{I})$. Observations are Gaussian around relative beacon position with covariance increasing rapidly with distance, and rewards depend only on successor state. Rollouts use a noisy goal-directed controller; we benchmark $d = 2,3,4$.

Two-Agent *d*D-Continuous Light-Dark POMDP. This variant rewards success only when both agents reach the goal. Parameters match the single-agent case except $\mathcal{S} = \mathcal{O} = \mathbb{R}^{2d}$; agent 1 observes its relative beacon position, while agent 2 observes its relative position to agent 1. We benchmark $d = 2$.

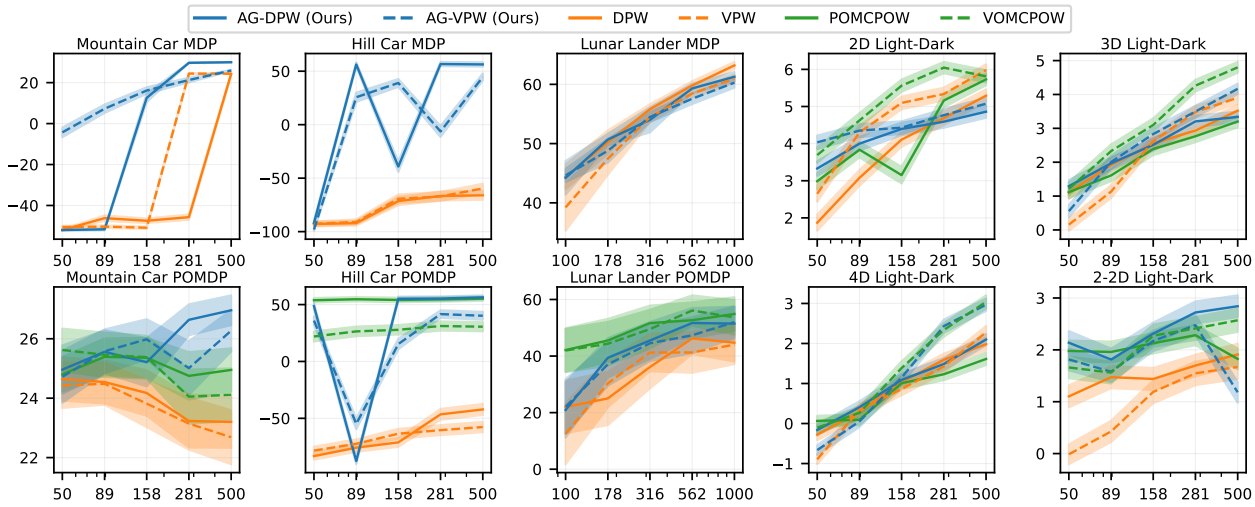


Figure 2: Average returns over 1000 seeds, shaded areas show ± 2 standard error. The x-axis shows the simulations budget per planning session. Blue (ours): AGMCTS algorithms Action-Gradient DPW (AG-DPW) for MDPs, and its particle-filter tree (PFT) adaptation for POMDPs. Orange: DPW for MDPs and PFT-DPW for POMDPs. Green: POMCPOW (POMDP domains only). We test both vanilla action sampling (solid lines) and VPW (dashed lines), resulting in 4 MDP and 6 POMDP algorithms.

Mountain Car and Hill Car MDP/POMDP. These domains are based on Mountain Car [Singh and Sutton, 1996] and Car on the Hill [Ernst *et al.*, 2005]; $\mathcal{S} = \mathbb{R}^2$ is position and velocity, and $\mathcal{A} = [-a_{\max}, a_{\max}]$ accelerates/brakes the car while avoiding the left boundary. Episodes terminate with a large penalty above a speed threshold, encouraging careful momentum control. In POMDPs, velocity is unobserved and observations are noisy positions.

Mountain Car has horizons up to 200 steps; Hill Car has horizon 30 and ODE-integrated dynamics. In both, noisy actions are clamped before simulation, yielding mixed continuous/discrete transition distributions.

Lunar Lander MDP/POMDP. We use Lunar Lander as in [Mern *et al.*, 2021; Lim *et al.*, 2021], with state $(x, y, \theta, \dot{x}, \dot{y}, \omega)$, noisy observations of angular rate, horizontal speed, and ground range, and deterministic dynamics perturbed by additive Gaussian noise.

5.1 Performance Evaluation

We tune hyperparameters with cross-entropy maximization at the maximum simulation budget. For AG variants this includes UCT/DPW and Adam step size; other AG settings are manual. VPW Voronoi parameters use reasonable values from [Lim *et al.*, 2021]; the Appendix lists all settings. For wall-clock matching, we set POMCPOW's simulation budget to PFT-DPW's empirical runtime. AG variants were 10–35$\times$ slower than DPW/VPW in MDPs and 2–20$\times$ slower than PFT-DPW/VPW in POMDPs.

Figure 2 shows performance. At maximum budget, an AG variant was preferable to its non-AG baselines in 6/10 scenarios (all Hill/Mountain Car, 4D and 2x2D Light-Dark). Against matched baselines, AG improved significantly in 10/20 cases, was insignificant in 7/20, and underperformed in 3/20.

By best algorithm per scenario, AGMCTS led in Mountain/Hill Car MDPs, was best or tied in 5/7 POMDPs, and underperformed in 2/7. Overall, it was competitive with POMCPOW/VOMCPOW and usually improved over PFT-DPW/PFT-VPW. This suggests AGMCTS helps when small action changes strongly affect states/rewards, as in long-horizon Mountain Car or narrow-goal Light-Dark; sparse returns can make score gradients noisy, as in Lunar Lander.

6 Conclusions

We introduced action-gradient continuous-POMDP tree search with MIS-consistent action-branch updates. Using the Area Formula to obtain transition log-probabilities, AGMCTS improved several continuous MDP/POMDP benchmarks and was competitive overall, showing that online gradient refinement can strengthen MCTS. Future work includes convergence analysis and faster, less tuning-sensitive variants.

Practical Considerations. AGMCTS currently takes more computation per simulation, driven by transition-density overhead, simulation budget, tree shape, and heuristics. Our runs were single-threaded, but particle-belief planners such as AGMCTS and PFT-DPW are highly parallelizable. The extra computation, transition-model requirements, and tuning burden should therefore be weighed against application latency and modeling constraints.

Acknowledgments

This work was supported by the Israel Ministry of Innovation, Science and Technology.

References

[Åström, 1965] Karl Johan Åström. Optimal control of Markov processes with incomplete state information

i. *Journal of mathematical analysis and applications*, 10:174–205, 1965.

[Azizzadenesheli *et al.*, 2018] Kamyar Azizzadenesheli, Yisong Yue, and Animashree Anandkumar. Policy gradient in partially observable environments: Approximation and convergence. *arXiv preprint arXiv:1810.07900*, 2018.

[Baxter and Bartlett, 2001] Jonathan Baxter and Peter L Bartlett. Infinite-horizon policy-gradient estimation. *journal of artificial intelligence research*, 15:319–350, 2001.

[Botev *et al.*, 2013] Zdravko I Botev, Dirk P Kroese, Reuven Y Rubinstein, and Pierre L'ecuyer. The cross-entropy method for optimization. In *Handbook of statistics*, volume 31, pages 35–59. Elsevier, 2013.

[Browne *et al.*, 2012] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[Cheng, 2006] Steve Cheng. Differentiation under the integral sign with weak derivatives, 2006. [Online].

[Couëtoux *et al.*, 2011] Adrien Couëtoux, Jean-Baptiste Hoock, Nataliya Sokolovska, Olivier Teytaud, and Nicolas Bonnard. Continuous upper confidence trees. In *International conference on learning and intelligent optimization*, pages 433–445. Springer, 2011.

[Doucet *et al.*, 2001] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods In Practice*. Springer-Verlag, New York, 2001.

[Egorov *et al.*, 2017] 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.

[Ernst *et al.*, 2005] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6, 2005.

[Farhi and Indelman, 2021] E. Farhi and V. Indelman. ix-bsp: Incremental belief space planning. *arXiv preprint arXiv:2102.09539*, 2021.

[Federer, 1969] Herbert Federer. *Geometric measure theory*, volume 153 of *Grundlehren Math. Wiss.* Springer, Cham, 1969.

[Folland, 1999] Gerald B Folland. *Real analysis: modern techniques and their applications*. John Wiley & Sons, 1999.

[Hoerger *et al.*, 2024] Marcus Hoerger, Hanna Kurniawati, Dirk Kroese, and Nan Ye. Adaptive discretization using Voronoi trees for continuous POMDPs. *The International Journal of Robotics Research*, 43(9):1283–1298, 2024.

[Hong *et al.*, 2024] Mao Hong, Zhengling Qi, and Yanxun Xu. A policy gradient method for confounded POMDPs. In *The Twelfth International Conference on Learning Representations*, 2024.

[JuliaPOMDP, 2024] JuliaPOMDP. ParticleFilters.jl: Simple particle filter implementation in Julia - works with POMDPs.jl models or others. GitHub repository, June 2024. Version v0.5.7.

[JuliaPOMDP, 2025a] JuliaPOMDP. MCTS.jl: Monte Carlo Tree Search for Markov decision processes using the POMDPs.jl framework. GitHub repository, October 2025. Version v0.5.7.

[JuliaPOMDP, 2025b] JuliaPOMDP. POMCPOW.jl: Online solver based on Monte Carlo tree search for POMDPs with continuous state, action, and observation spaces. GitHub repository, April 2025. Version v0.3.11.

[Kaelbling *et al.*, 1998] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.

[Kim *et al.*, 2020] Beomjoon Kim, Kyungjae Lee, Sungbin Lim, Leslie Kaelbling, and Tomás Lozano-Pérez. Monte Carlo tree search in continuous spaces using Voronoi optimistic optimization with regret bounds. In *AAAI Conf. on Artificial Intelligence*, volume 34, pages 9916–9924, 2020.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[Kloek and Van Dijk, 1978] Teun Kloek and Herman K Van Dijk. Bayesian estimates of equation system parameters: an application of integration by Monte Carlo. *Econometrica: Journal of the Econometric Society*, pages 1–19, 1978.

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

[Kong *et al.*, 1994] A. Kong, J. S. Liu, and W. H. Wong. Sequential imputations and Bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):278–288, 1994.

[Kujanpää *et al.*, 2024] Kalle Kujanpää, Amin Babadi, Yi Zhao, Juho Kannala, Alexander Ilin, and Joni Pajarinen. Continuous Monte Carlo graph search. In *Intl. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 1047–1056, 2024.

[Kurniawati, 2022] Hanna Kurniawati. Partially observable Markov decision processes and robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 5(1):253–277, 2022.

[Lauri *et al.*, 2022] Mikko Lauri, David Hsu, and Joni Pajarinen. Partially observable Markov decision processes in robotics: A survey. *IEEE Transactions on Robotics*, 39(1):21–40, 2022.

[Lee *et al.*, 2020] Jongmin Lee, Wonseok Jeon, Geon-Hyeong Kim, and Kee-Eung Kim. Monte-Carlo tree

search in continuous action spaces with value gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 4561–4568, 2020.

[Leurent and Maillard, 2020] Edouard Leurent and Odalric-Ambrym Maillard. Monte-Carlo graph search: the value of merging similar states. In Sinno Jialin Pan and Masashi Sugiyama, editors, *Asian Conference on Machine Learning (ACML 2020)*, pages 577 – 592, Bangkok, Thailand, November 18-20 2020.

[Lim *et al.*, 2021] Michael H Lim, Claire J Tomlin, and Zachary N Sunberg. Voronoi progressive widening: efficient online solvers for continuous state, action, and observation POMDPs. In *2021 60th IEEE conference on decision and control (CDC)*, pages 4493–4500. IEEE, 2021.

[Lim *et al.*, 2023] Michael H Lim, Tyler J Becker, Mykel J Kochenderfer, Claire J Tomlin, and Zachary N Sunberg. Optimality guarantees for particle belief approximation of POMDPs. *Journal of Artificial Intelligence Research*, 77:1591–1636, 2023.

[Madani *et al.*, 2003] Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*, 147(1-2):5–34, 2003.

[Mern *et al.*, 2021] John Mern, Anil Yildiz, Zachary Sunberg, Tapan Mukerji, and Mykel J Kochenderfer. Bayesian optimized Monte Carlo planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11880–11887, 2021.

[Metelli *et al.*, 2020] Alberto Maria Metelli, Matteo Papini, Nico Montali, and Marcello Restelli. Importance sampling techniques for policy optimization. *J. Mach. Learn. Res.*, 21(141):1–75, 2020.

[Morere *et al.*, 2018] Philippe Morere, Roman Marchant, and Fabio Ramos. Continuous state-action-observation POMDPs for trajectory planning with Bayesian optimisation. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 8779–8786. IEEE, 2018.

[Moses and Churavy, 2020] William Moses and Valentin Churavy. Instead of rewriting foreign code for machine learning, automatically synthesize fast gradients. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12472–12485. Curran Associates, Inc., 2020.

[Negro, 2022] Luigi Negro. Sample distribution theory using coarea formula. *Communications in Statistics-Theory and Methods*, pages 1–26, 2022.

[Novitsky *et al.*, 2025] Michael Novitsky, Moran Barenboim, and Vadim Indelman. Previous knowledge utilization in online anytime belief space planning. *IEEE Robotics and Automation Letters (RA-L)*, 2025.

[Owen, 2013] Art B. Owen. *Monte Carlo theory, methods and examples*. https://artowen.su.domains/mc/, 2013.

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

[Platt *et al.*, 2010] 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.

[Porta *et al.*, 2006] J. M. Porta, N. Vlassis, M. T. Spaan, and P. Poupart. Point-based value iteration for continuous POMDPs. *J. of Machine Learning Research*, 7:2329–2367, 2006.

[Rackauckas *et al.*, 2020] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, and Ali Ramadhan. Universal differential equations for scientific machine learning, 2020.

[Revels *et al.*, 2016] J. Revels, M. Lubin, and T. Papamarkou. Forward-mode automatic differentiation in Julia, 2016.

[Rubinstein and Kroese, 2004] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2004.

[Schulman *et al.*, 2016] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Intl. Conf. on Learning Representations (ICLR)*, 2016.

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

[Silver *et al.*, 2014] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. Pmlr, 2014.

[Singh and Sutton, 1996] Satinder P Singh and Richard S Sutton. Reinforcement learning with replacing eligibility traces. *Machine learning*, 22:123–158, 1996.

[Sunberg and Kochenderfer, 2018] 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.

[Sutton *et al.*, 1999] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[Tsitouras, 2011] Ch Tsitouras. Runge–kutta pairs of order 5 (4) satisfying only the first column simplifying as-

sumption. *Computers & mathematics with applications*, 62(2):770–775, 2011.

[Veach and Guibas, 1995] Eric Veach and Leonidas J Guibas. Optimally combining sampling techniques for Monte Carlo rendering. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 419–428. ACM, 1995.

[Wu *et al.*, 2021] 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.

[Ye *et al.*, 2017] Nan Ye, Adhiraj Somani, David Hsu, and Wee Sun Lee. DESPOT: Online POMDP planning with regularization. *JAIR*, 58:231–266, 2017.

[Yee *et al.*, 2016] Timothy Yee, Viliam Lisỳ, Michael H Bowling, and S Kambhampati. Monte Carlo Tree Search in continuous action spaces with execution uncertainty. In *IJCAI*, pages 690–697, 2016.

A Proofs and Derivations

A.1 Lemma 1

Lemma 1. *Let $\bar{b} = ((s^j, \lambda^j))_{j=1}^J$, $\bar{b}'^- = ((s'^{-,j}, \lambda'^{-,j}))_{j=1}^J$ be ordered particle beliefs. The probability that $\bar{b}'^-$ is a propagated belief in the bootstrap filter, given $\bar{b}$ and action a, is:*

$$p(\bar{b}'^-|\bar{b}, a) = \prod_{j=1}^J p_T(s'^{-,j}|s^j, a), \tag{22}$$

whenever $\lambda'^{-,j} = \lambda^j$ for all $j = 1, \ldots, J$, and zero otherwise. When $p(\bar{b}'^-|\bar{b}, a) > 0$ it holds that:

$$\nabla_a \log p(\bar{b}'^-|\bar{b}, a) = \sum_{j=1}^J \nabla_a \log p_T(s'^{-,j}|s^j, a). \tag{23}$$

Proof. In the bootstrap filter algorithm, each particle is sampled $s'^{-,j} \sim p_T(\cdot|s^j, a)$ independently for each $j = 1, \ldots, J$. Therefore:

$$p(\bar{b}'^-|\bar{b}, a) = p(((s'^{-,j}, \lambda'^{-,j}))_{j=1}^J|((s^l, \lambda^l))_{l=1}^J, a), \tag{24}$$

due to the beliefs being ordered, and the independent sampling, we can factorize the terms into a product:

$$= \prod_{i=1}^J p(((s'^{-,j}, \lambda'^{-,j}))|((s^l, \lambda^l))_{l=1}^J, a) \tag{25}$$

$$= \prod_{i=1}^J p(((s'^{-,j}, \lambda'^{-,j}))|(s^j, \lambda^j), a). \tag{26}$$

Since $\lambda'^{-,j} = \lambda^j$, we can drop the weights to obtain

$$p(\bar{b}'^-|\bar{b}, a) = \prod_{j=1}^J p_T(s'^{-,j}|s^j, a). \tag{27}$$

□

A.2 Theorem 1

We first highlight that the required "well-behaved" conditions for the theorem are for changing the order of derivatives and integrals by the Leibniz integral rule. Sufficient conditions are characterized by Folland, Theorem 2.27 [1999]:

Theorem 2.27. *Suppose that $f : X \times [a, b] \to \mathbb{C}(-\infty < a < b < \infty)$ and that $f(\cdot, t) : X \to \mathbb{C}$ is integrable for each $t \in [a, b]$. Let $F(t) = \int_X f(x, t)\, \mathrm{d}\mu(x)$.*

1. *Suppose that there exists $g \in L^1(\mu)$ such that $|f(x, t)| \leq g(x)$ for all x, t. If $\lim_{t\to t_0} f(x, t) = f(x, t_0)$ for every x, then $lim_{t\to t_0} F(t) = F(t_0)$; in particular, if $f(x, \cdot)$ is continuous for each x, then F is continuous.*
2. *Suppose that $\partial f/\partial t$ exists and there is a $g \in L^1(\mu)$ such that $|\partial f/\partial t(x, t)| \leq g(x)$ for all x, t. Then F is differentiable and $F' = \int (\partial f/\partial t)(x, t)\, \mathrm{d}\mu(x)$.*

More general versions of the theorem have been proven [Cheng, 2006]. Since such generalizations are not central to our contribution, we present Theorem 1 under a set of sufficient but non-minimal assumptions. We note that Lebesgue integrability may hold for a very broad class of functions, including functions with a countable set of discontinuities, and densities of common distributions such as regular and clipped Gaussians. This is important for many MDP and POMDP problem formulations, in which sparse rewards and discontinuous transition and observation models are often used.

For clarity, we present the results for a generic action $a \in \mathcal{A}$. Unlike the main text, where $\breve{a}$ denotes an updated action relative to the sampling action, such a distinction is unnecessary here due to the use of general proposal distributions.

Theorem 1. *Assume: (i) $\rho_q^{p_T}(s')$, $\rho_q^p(\bar{b}'^-)$ are well-defined; (ii) r, V are bounded; (iii) there exist integrable g_r, g_T w.r.t. the MDP/POMDP measures such that $\|\nabla_a r\| \leq g_r$ and $\|\nabla_a \log p_T\| \leq g_T$. For POMDPs, assume the assumptions of Lemma 1 hold. Then, from the dominated-convergence form of Leibniz integral rule [Folland, 1999, Theorem 2.27], the action score gradient satisfies:*

$$\nabla_a Q_t^\pi(s, a) = \mathbb{E}_{s'|q}\Big[\rho_q^{p_T}(s')[\nabla_a \log p_T(s'|s, a) \\ (r(s, a, s') + \gamma V_{t+1}^\pi(s') - B(s)) + \nabla_a r(s, a, s')]\Big], \tag{28}$$

$$\nabla_a Q_t^\pi(\bar{b}, a) = \mathbb{E}_{\bar{b}'^-, o', \bar{b}'|q}\Big[\rho_q^p(\bar{b}'^-)[\nabla_a \log p(\bar{b}'^-|\bar{b}, a) \\ (r(\bar{b}, a, \bar{b}') + \gamma V_{t+1}^\pi(\bar{b}') - B(\bar{b})) + \nabla_a r(\bar{b}, a, \bar{b}')]\Big], \tag{29}$$

where $B(s)$ (resp. $B(\bar{b})$) is any baseline independent of a for estimator variance reduction [Schulman et al.*, 2016], e.g. $B(s) = V_t^\pi(s)$.*

Proof. Recall the definition of the importance weights

$$\rho_q^{p_T}(s') = \frac{p_T(s'|s, a)}{q(s'|s, a)}, \quad \rho_q^p(\bar{b}'^-) = \frac{p(\bar{b}'^-|\bar{b}, a)}{q(\bar{b}'^-|\bar{b}, a)}. \tag{30}$$

MDP Case, $B(s) = 0$. In this case, the action-value function is given by:

$$\nabla_a Q_t^\pi(s, a)$$

$$= \nabla_a \mathbb{E}_{s'|q}\left[\rho_q^{p_T}(s')\left(r(s, a, s') + \gamma V_{t+1}^\pi(s')\right)\right] \tag{31}$$

$$= \nabla_a \int_{s'} q(s'|s, a) \frac{p_T(s'|s, a)}{q(s'|s, a)} \cdot \left(r(s, a, s') + \gamma V_{t+1}^\pi(s')\right) \mathrm{d}s' \tag{32}$$

$$= \nabla_a \int_{s'} p_T(s'|s, a)\left(r(s, a, s') + \gamma V_{t+1}^\pi(s')\right) \mathrm{d}s'. \tag{33}$$

From the assumption that r and V are bounded, hence their sum is also bounded by some $M > 0$:

$$\left|r(s, a, s') + \gamma V_{t+1}^\pi(s')\right| \leq M. \tag{34}$$

For each a_i (a coordinate of $a \in \mathcal{A}$), the partial derivative satisfies

$$\frac{\partial}{\partial a_i}\Big(p_T(s'|s, a)\left(r(s, a, s') + \gamma V_{t+1}^\pi(s')\right)\Big) \\ = \frac{\partial}{\partial a_i} p_T(s'|s, a) \cdot \left(r(s, a, s') + \gamma V_{t+1}^\pi(s')\right) \\ + p_T(s'|s, a) \cdot \frac{\partial}{\partial a_i} r(s, a, s'), \tag{35}$$

where importantly, $\frac{\partial}{\partial a_i} V_{t+1}^{\pi}(s') = 0$ due to s' being an integration variable that is independent of a. We can assume that $p_T(s'|s,a) > 0$, as the set of s' where $p_T(s'|s,a) = 0$ does not contribute to the integral. Thus, we can apply the log-gradient transform to obtain:

$$
\begin{aligned}
&(35)\\
&= p_T(s'|s,a) \cdot \frac{\partial}{\partial a_i} \log p_T(s'|s,a) \cdot (r(s,a,s') \\
&\quad + \gamma V_{t+1}^{\pi}(s')) + p_T(s'|s,a) \cdot \frac{\partial}{\partial a_i} r(s,a,s'), && (36)\\
&= p_T(s'|s,a) \cdot \big(\frac{\partial}{\partial a_i} \log p_T(s'|s,a) \\
&\quad \cdot (r(s,a,s') + \gamma V_{t+1}^{\pi}(s')) + \frac{\partial}{\partial a_i} r(s,a,s')\big). && (37)
\end{aligned}
$$

By our assumptions, it follows that there exist integrable $g_1(s')$, $g_2(s')$ w.r.t. p_T, for which it holds that $\frac{\partial}{\partial a_i} \log p_T(s'|s,a) \leq g_1(s')$ and $\frac{\partial}{\partial a_i} r(s,a,s') \leq g_2(s')$ for all s, a, s'. Therefore, there exists an integrable envelope $g(s') \triangleq M g_1(s') + g_2(s')$ for (37):

$$
\begin{aligned}
&\Big| p_T(s'|s,a) \cdot \big(\frac{\partial}{\partial a_i} \log p_T(s'|s,a) \\
&\quad \cdot (r(s,a,s') + \gamma V_{t+1}^{\pi}(s')) + \frac{\partial}{\partial a_i} r(s,a,s')\big)\Big| \\
&\leq p_T(s'|s,a)\,(M g_1(s') + g_2(s')) && (38)\\
&= p_T(s'|s,a) g(s'). && (39)
\end{aligned}
$$

Summation and scalar multiplication preserve integrability, and it follows that $g(s')$ is integrable w.r.t. p_T [Folland, 1999]. Therefore, from Theorem 2.27 (clause 2), we can apply the Leibniz integral rule (in vector form):

$$
\begin{aligned}
&\nabla_a \int_{s'} p_T(s'|s,a)(r(s,a,s') + \gamma V_{t+1}^{\pi}(s'))\,\mathrm{d}s' && (40)\\
&= \int_{s'} \nabla_a \big(p_T(s'|s,a)(r(s,a,s') + \gamma V_{t+1}^{\pi}(s'))\big)\,\mathrm{d}s' \\
&= \int_{s'} p_T(s'|s,a)\big(\nabla_a \log p_T(s'|s,a) \\
&\quad \cdot (r(s,a,s') + \gamma V_{t+1}^{\pi}(s')) + \nabla_a r(s,a,s')\big)\,\mathrm{d}s'. && (41)
\end{aligned}
$$

By bringing back the importance ratio, we obtain the required result (28):

$$
\begin{aligned}
&(41)\\
&= \int_{s'} q(s'|s,a) \frac{p_T(s'|s,a)}{q(s'|s,a)} \big(\nabla_a \log p_T(s'|s,a) \\
&\quad (r(s,a,s') + \gamma V_{t+1}^{\pi}(s')) + \nabla_a r(s,a,s')\big)\,\mathrm{d}s' && (42)\\
&= \mathbb{E}_{s'|q}\Big[\rho_q^{p_T}(s')[\nabla_a \log p_T(s'|s,a)(r(s,a,s') \\
&\quad + \gamma V_{t+1}^{\pi}(s')) + \nabla_a r(s,a,s')]\Big]. && (43)
\end{aligned}
$$

MDP Case, $B(s) \neq 0$**.** Let $F_t^{\pi}(s,a) \triangleq Q_t^{\pi}(s,a) - B(s)$. On the one hand, it holds that:

$$
\begin{aligned}
&\nabla_a F_t^{\pi}(s,a) \\
&= \nabla_a Q_t^{\pi}(s,a) - \nabla_a B(s) && (44)\\
&= \nabla_a Q_t^{\pi}(s,a), && (45)
\end{aligned}
$$

as $B(s)$ is independent of a. On the other hand, we can apply the steps of the previous case to $F_t^{\pi}(s,a)$, resulting in:

$$
\begin{aligned}
&\nabla_a F_t^{\pi}(s,a) \\
&= \nabla_a \mathbb{E}_{s'|s,a}\Big[\big(r(s,a,s') + \gamma V_{t+1}^{\pi}(s') - B(s)\big)\Big] && (46)\\
&= \mathbb{E}_{s'|s,a}\Big[\nabla_a \log p_T(s'|s,a) \\
&\quad \cdot \big(r(s,a,s') + \gamma V_{t+1}^{\pi}(s') - B(s)\big) + \nabla_a r(s,a,s')\Big] && (47)\\
&= \mathbb{E}_{s'|q}\Big[\rho_q^{p_T}(s')[\nabla_a \log p_T(s'|s,a) \\
&\quad \cdot (r(s,a,s') + \gamma V_{t+1}^{\pi}(s') - B(s)) + \nabla_a r(s,a,s')]\Big]. && (48)
\end{aligned}
$$

By equating the two expressions for $\nabla_a F_t^{\pi}(s,a)$, we obtain the required result.

POMDP case. We shall prove for $B(\bar{b} = 0)$, as the case $B(\bar{b} \neq 0)$ follows similarly to the MDP case.

We begin by showing propagated belief decomposition applied to the action-value function:

$$
\begin{aligned}
&Q_t^{\pi}(\bar{b},a) \\
&= \mathbb{E}_{\bar{b}'|\bar{b},a}\big[r(\bar{b},a,\bar{b}') + \gamma V_{t+1}^{\pi}(\bar{b}')\big] && (49)\\
&= \int_{\bar{b}'} p(\bar{b}'|\bar{b},a)(r(\bar{b},a,\bar{b}') + \gamma V_{t+1}^{\pi}(\bar{b}'))\,\mathrm{d}\bar{b}', && (50)
\end{aligned}
$$

and we continue by marginalizing over the propagated belief $\bar{b}'^{-}$ and the observation o':

$$
\begin{aligned}
&= \int_{\bar{b}'^{-}} \int_{o'} \int_{\bar{b}'} p(\bar{b}'^{-}, o', \bar{b}'|\bar{b},a)(r(\bar{b},a,\bar{b}') \\
&\quad + \gamma V_{t+1}^{\pi}(\bar{b}'))\,\mathrm{d}\bar{b}'\,\mathrm{d}o'\,\mathrm{d}\bar{b}'^{-}. && (51)
\end{aligned}
$$

By its definition, the propagated belief $\bar{b}'^{-}$ is the generated belief by conditioning on the history without the last observation. Therefore, by using the chain rule and removing variables that are conditionally independent, we obtain:

$$
\begin{aligned}
&Q_t^{\pi}(\bar{b},a) \\
&= \int_{\bar{b}'^{-}} \int_{o'} \int_{\bar{b}'} p(\bar{b}'|o',\bar{b}'^{-},\cancel{\bar{b},a}) p(o'|\bar{b}'^{-},\cancel{\bar{b},a}) \\
&\quad p(\bar{b}'^{-}|\bar{b},a)(r(\bar{b},a,\bar{b}') + \gamma V_{t+1}^{\pi}(\bar{b}'))\,\mathrm{d}\bar{b}'\,\mathrm{d}o'\,\mathrm{d}\bar{b}'^{-} && (52)\\
&= \int_{\bar{b}'^{-}} \int_{o'} \int_{\bar{b}'} p(\bar{b}'|o',\bar{b}'^{-}) p(o'|\bar{b}'^{-}) p(\bar{b}'^{-}|\bar{b},a) \\
&\quad (r(\bar{b},a,\bar{b}') + \gamma V_{t+1}^{\pi}(\bar{b}'))\,\mathrm{d}\bar{b}'\,\mathrm{d}o'\,\mathrm{d}\bar{b}'^{-} && (53)\\
&= \mathbb{E}_{\bar{b}'^{-}|\bar{b},a} \mathbb{E}_{o'|\bar{b}'^{-}} \mathbb{E}_{\bar{b}'|\bar{b}'^{-},o'}\big[r(\bar{b},a,\bar{b}') + \gamma V_{t+1}^{\pi}(\bar{b}')\big]. && (54)
\end{aligned}
$$

We introduce the importance ratio $\rho_q^p(\bar{b}'^-) \triangleq \frac{p(\bar{b}'^-|\bar{b},a)}{q(\bar{b}'^-|\bar{b},a)}$ over the propagated belief:

$$
\begin{aligned}
&Q_t^\pi(\bar{b},a) \\
&= \mathbb{E}_{\bar{b}'^-|\bar{b},a}\mathbb{E}_{o'|\bar{b}'^-}\mathbb{E}_{\bar{b}'|\bar{b}'^-,o'}\Big[\\
&\qquad \frac{q(\bar{b}'^-|\bar{b},a)}{q(\bar{b}'^-|\bar{b},a)}(r(\bar{b},a,\bar{b}') + \gamma V_{t+1}^\pi(\bar{b}'))\Big], \qquad (55)\\
&= \mathbb{E}_{\bar{b}'^-|q}\mathbb{E}_{o'|\bar{b}'^-}\mathbb{E}_{\bar{b}'|\bar{b}'^-,o'}[\rho_q^p(\bar{b}'^-)(r(\bar{b},a,\bar{b}') + \gamma V_{t+1}^\pi(\bar{b}'))], \qquad (56)
\end{aligned}
$$

Similarly to the derivation of the MDP case, we can apply the Leibniz integral rule to obtain the result:

$$
\begin{aligned}
&\nabla_a Q_t^\pi(\bar{b},a) \\
&= \mathbb{E}_{\bar{b}'^-|q}\mathbb{E}_{o'|\bar{b}'^-}\mathbb{E}_{\bar{b}'|\bar{b}'^-,o'}\Big[\rho_q^p(\bar{b}'^-)\Big(\nabla_a \log p(\bar{b}'^-|\bar{b},a) \\
&\qquad \cdot (r(\bar{b},a,\bar{b}') + \gamma V_{t+1}^\pi(\bar{b}')) + \nabla_a r(\bar{b},a,\bar{b}')\Big)\Big]. \qquad (57)
\end{aligned}
$$

□

A.3 Theorem 1 Extended (Immediate Reward Gradient)

This theorem was not presented in the main paper due to space limitations. It is an alternate form of Theorem 1 for MDPs and state-reward POMDPs, and it was used in the gradient calculation of some of the experimental domains, hence it is brought here for completeness.

Theorem 1 Extended (Immediate Reward Gradient) modifies the immediate reward gradient for MDPs and state-reward POMDPs, by basing it on state samples from the updated action rather than scoring previous value estimates. It may give more accurate gradient estimates, albeit possibly more expensive to compute.

We refer to a *state-reward POMDP* when the reward function can be expressed as $r(b,a,b') = \mathbb{E}_{s,s'|b,a,b'}[r(s,a,s')]$.

We highlight that as an extension of Theorem 1, it requires the same conditions. Additionally, for the simplification of the immediate reward expectation, we make use of Fubini's theorem for changing integration orders. Sufficient conditions are that $r(s,a,s')$ is always integrable w.r.t. the POMDP measure.

Theorem 1 Extended (Immediate Reward Gradient). *Under the conditions of Theorem 1, the action-gradient of the MDP action-value function is given by:*

$$
\begin{aligned}
&\nabla_a Q_t^\pi(s,a) \\
&= \mathbb{E}_{s'|s,a}\Big[\nabla_a \log p_T(s'|s,a)\cdot r(s,a,s') + \nabla_a r(s,a,s')\Big] \\
&+ \mathbb{E}_{s'|q}\Big[\rho_q^{p_T}(s')\cdot \nabla_a \log p_T(s'|s,a)\cdot(\gamma V_{t+1}^\pi(s') - B(s))\Big]. \qquad (58)
\end{aligned}
$$

For a state-reward POMDP, the action-gradient of the action-value function is given by:

$$
\begin{aligned}
&\nabla_a Q_t^\pi(\bar{b},a) \\
&= \mathbb{E}_{s|\bar{b}}\mathbb{E}_{s'|s,a}\Big[\nabla_a \log p_T(s'|s,a)\cdot r(s,a,s') \\
&\qquad + \nabla_a r(s,a,s')\Big] \\
&+ \mathbb{E}_{\bar{b}'^-|q}\mathbb{E}_{o|\bar{b}'^-}\mathbb{E}_{\bar{b}'|\bar{b}'^-,o}\Big[\rho_q^p(\bar{b}'^-) \\
&\qquad \cdot \nabla_a \log p(\bar{b}'^-|\bar{b},a)\cdot(\gamma V_{t+1}^\pi(\bar{b}') - B(\bar{b}))\Big]. \qquad (59)
\end{aligned}
$$

Proof. The central part of this proof is the decomposition of the belief action-value function to an expectation over state trajectories of immediate rewards, and an expectation over belief trajectories of future values. Afterwards, we apply the Leibniz integral rule similarly to Theorem 1. We show the derivation of the POMDP case, as the MDP case is a simplified version of it.

Similarly to Theorem 1, for any baseline $B(\bar{b})$ that is independent of a, we have:

$$
\begin{aligned}
&\nabla_a(Q_t^\pi(\bar{b},a) - B(\bar{b})) \\
&= \nabla_a Q_t^\pi(\bar{b},a) \qquad (60)\\
&= \nabla_a \mathbb{E}_{\bar{b}'|\bar{b},a}\left[r(\bar{b},a,\bar{b}') + \gamma V_{t+1}^\pi(\bar{b}') - B(\bar{b})\right], \qquad (61)
\end{aligned}
$$

as $B(\bar{b})$ is independent of a and it holds that $\nabla_a B(\bar{b}) = 0$. We rewrite the belief action-value function:

$$
\begin{aligned}
&Q_t^\pi(\bar{b},a) - B(\bar{b}) \\
&= \mathbb{E}_{\bar{b}'|\bar{b},a}\left[r(\bar{b},a,\bar{b}') + \gamma V_{t+1}^\pi(\bar{b}')\right] - B(\bar{b}) \qquad (62)\\
&= \mathbb{E}_{\bar{b}'|\bar{b},a}\left[r(\bar{b},a,\bar{b}')\right] + \mathbb{E}_{\bar{b}'|\bar{b},a}\left[\gamma V_{t+1}^\pi(\bar{b}') - B(\bar{b})\right]. \qquad (63)
\end{aligned}
$$

Expanding the first expectation, and using the chain rule and then Fubini's theorem to cancel the expectation over the observation and posterior belief, we obtain:

$$
\begin{aligned}
&\mathbb{E}_{\bar{b}'|\bar{b},a}\left[r(\bar{b},a,\bar{b}')\right] \\
&= \mathbb{E}_{\bar{b}'^-|\bar{b},a}\mathbb{E}_{o|\bar{b}'^-}\mathbb{E}_{\bar{b}'|\bar{b}'^-,o}\left[\mathbb{E}_{s,s'|\bar{b},a,\bar{b}'}[r(s,a,s')]\right] \qquad (64)\\
&= \int_{\bar{b}'^-}\int_o\int_{\bar{b}'}\int_s\int_{s'} p(s,s',o,\bar{b}'^-,\bar{b}'|\bar{b},a) \\
&\qquad\qquad r(s,a,s')\,\mathrm{d}s'\,\mathrm{d}s\,\mathrm{d}\bar{b}'\,\mathrm{d}o\,\mathrm{d}\bar{b}'^- \qquad (65)\\
&= \int_{\bar{b}'^-}\int_o\int_{\bar{b}'}\int_s\int_{s'} p(o,\bar{b}',\bar{b}'^-|s,s',\bar{b},a)p(s,s'|\bar{b},a) \\
&\qquad\qquad r(s,a,s')\,\mathrm{d}s'\,\mathrm{d}s\,\mathrm{d}\bar{b}'\,\mathrm{d}o\,\mathrm{d}\bar{b}'^- \qquad (66)\\
&= \int_s\int_{s'} p(s,s'|\bar{b},a)r(s,a,s') \\
&\cancel{\left(\int_{\bar{b}'^-}\int_o\int_{\bar{b}'} p(o,\bar{b}',\bar{b}'^-|s,s',\bar{b},a)\,\mathrm{d}\bar{b}'\,\mathrm{d}o\,\mathrm{d}\bar{b}'^-\right)}\,\mathrm{d}s'\,\mathrm{d}s \qquad (67)\\
&= \int_s\int_{s'} p(s'|s,a,\cancel{\bar{b}})p(s|\bar{b},\cancel{a})r(s,a,s')\,\mathrm{d}s'\,\mathrm{d}s \qquad (68)\\
&= \mathbb{E}_{s|\bar{b}}\mathbb{E}_{s'|s,a}[r(s,a,s')]. \qquad (69)
\end{aligned}
$$

Now, we apply ∇_a on the obtained terms. The gradient of the immediate reward expectation, following steps similar to Theorem 1, is given by:

$$\begin{aligned}
&\nabla_a \mathbb{E}_{s|\bar{b}} \mathbb{E}_{s'|s,a} \left[r(s,a,s') \right] \\
&= \mathbb{E}_{s|\bar{b}} \mathbb{E}_{s'|s,a} \Big[\Big(\nabla_a \log p_T(s'|s,a) \cdot r(s,a,s') \\
&\qquad + \nabla_a r(s,a,s') \Big) \Big]. \qquad (70)
\end{aligned}$$

Next, the gradient of the future value expectation, when adding the importance ratio, is calculated by:

$$\begin{aligned}
&\nabla_a \mathbb{E}_{\bar{b}'|\bar{b},a} \Big[\gamma V^{\pi}_{t+1}(\bar{b}') - B(\bar{b}) \Big] \\
&= \nabla_a \mathbb{E}_{\bar{b}'^-|\bar{b},a} \mathbb{E}_{o|\bar{b}'^-} \mathbb{E}_{\bar{b}'|\bar{b}'^-,o} \Big[\frac{q(\bar{b}'^-|\bar{b},a)}{q(\bar{b}'^-|\bar{b},a)} \\
&\qquad \cdot (\gamma V^{\pi}_{t+1}(\bar{b}') - B(\bar{b})) \Big] \qquad (71) \\
&= \mathbb{E}_{\bar{b}'^-|q} \mathbb{E}_{o|\bar{b}'^-} \mathbb{E}_{\bar{b}'|\bar{b}'^-,o} \Big[\rho^p_q(\bar{b}'^-) \\
&\qquad \cdot \nabla_a \log p(\bar{b}'^-|\bar{b},a) \cdot (\gamma V^{\pi}_{t+1}(\bar{b}') - B(\bar{b})) \Big]. \qquad (72)
\end{aligned}$$

Combining the two results, we obtain the final expression (59). □

A.4 Theorem 2

We first start giving the formal statement of Lemma 9.1 by Veach and Guibas [1995]:

Lemma 9.1. *Let F be any estimator of the form*

$$F = \sum_{i=1}^{n} \frac{1}{n_i} \sum_{j=1}^{n_i} w_i(X_{i,j}) \frac{f(X_{i,j})}{p_i(X_{i,j})},$$

where $n_i \geq 1$ for all i, and the weighting functions w_i satisfy conditions

$$\begin{aligned}
&\textit{(MIS-I)} \quad \sum_{i=1}^{n} w_i(x) = 1 \quad \textit{whenever } f(x) \neq 0, \\
&\textit{(MIS-II)} \quad w_i(x) = 0 \quad \textit{whenever } p_i(x) = 0.
\end{aligned}$$

The proof follows from the linearity of the expectation.

An important corollary brought by Veach and Guibas [1995] is that these conditions imply that at any point where $f(x) \neq 0$, at least one of the $p_i(x)$ must be positive. Thus, it is not necessary for every p_i to sample the whole domain, and it is allowable for some p_i to be specialized sampling techniques that concentrate on specific regions of the integrand.

We recall that our target function to estimate is given by the equations:

$$\begin{aligned}
Q^{\pi}_t(s,a) &= \mathbb{E}_{s'|q}[\rho^{p_T}_q(s')(r(s,a,s') + \gamma V^{\pi}_{t+1}(s'))] \qquad (73) \\
&= \mathbb{E}_{s'|q}[\frac{p_T(s'|s,a)}{q(s'|s,a)}(r(s,a,s') + \gamma V^{\pi}_{t+1}(s'))]. \qquad (74)
\end{aligned}$$

Hence, for each successor branch in $s'^{,i} \in \mathcal{S}_{sa}$, the target function to estimate is the same $p_T(s'^{,i}|s,a)(r(s,a,s'^{,i}) + \gamma V^{\pi}_{t+1}(s'^{,i}))$, but with the proposal distribution $q(s'^{,i}|s,a) = p_T(s'^{,i}|s,a^i_{\text{prop}})$. We denote the branch importance ratio:

$$\rho^i_a(s') = \frac{p_T(s'|s,a)}{p_T(s'|s,a^i_{\text{prop}})}, \qquad (75)$$

and we define the following MIS estimators of a single sample of each proposal distribution:

$$\hat{V}_f(s,a) \triangleq \sum_{i=1}^{|\mathcal{S}_{sa}|} w_i(s'^{,i}) \rho^i_a(s'^{,i}) \hat{V}(s'^{,i}), \qquad (76)$$

$$\hat{r}(s,a) \triangleq \sum_{i=1}^{|\mathcal{S}_{sa}|} w_i(s'^{,i}) \rho^i_a(s'^{,i}) r(s,a,s'^{,i}), \qquad (77)$$

for weighting functions $w_i(s'^{,i})$ that may in general depend on s, a, s' and a.

We've defined the value estimate of each state node recursively as:

$$\hat{V}(s) \triangleq \sum_{a \in C(s)} \frac{n(s,a)}{n(s)} \hat{Q}(s,a), \qquad (78)$$

unless s' is at the maximum tree depth, in which case its value estimate is the mean of the rollout values:

$$\hat{V}(s) = \frac{1}{n(s)_{+1}} \sum_{i=1}^{n(s)_{+1}} v^i, \qquad (79)$$

$$(80)$$

for all v^i rollout values from the terminal node s.

Theorem 2. *If $w_i(s')$ satisfy requirements (MIS-I) and (MIS-II) (see Section 2), then the MIS Tree yields unbiased value and action-value estimates for a given tree structure.*

Proof. The proof goes by a mutual induction over $\hat{V}$ and $\hat{Q}$, proving that a node's value or action-value estimate is unbiased if all its children have unbiased estimates.

Base case: leaf state nodes are unbiased. As stated in the tree construction, $\hat{V}(s)$ when s is a leaf node is the mean of the rollout values, and therefore:

$$\begin{aligned}
&\mathbb{E}_{s \sim \pi_{rollout}} \left[\hat{V}(s) \right] \\
&= \mathbb{E}_{s \sim \pi_{rollout}} \left[\frac{1}{n(s)_{+1}} \sum_{i=1}^{n(s)_{+1}} v^i \right] \qquad (81) \\
&= \frac{1}{n(s)_{+1}} \sum_{i=1}^{n(s)_{+1}} \mathbb{E}_{s \sim \pi_{rollout}} \left[v^i \right] \qquad (82) \\
&= V^{\pi_{rollout}}(s). \qquad (83)
\end{aligned}$$

Hence, it is unbiased.

Inductive step 1: Value estimates $\hat{V}$ are unbiased.

Induction hypothesis: Assume that $\hat{Q}(s,a^i)$ is unbiased, i.e. $\mathbb{E}\left[\hat{Q}(s,a^i)\right] = Q(s,a^i)$, for all $a^i \in \mathcal{A}_s$.

The value estimate is defined as the mean of the action-value estimates weighted by the number of times each action

was selected. Therefore, for the policy representing the current visitation counters, i.e. defined by $\pi_{visits}(s) \overset{\triangle}{\sim} n(s,a)/n(s)$, the value estimate satisfies:

$$\begin{aligned}
&\mathbb{E}\left[\hat{V}(s)\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{|\mathcal{A}_s|} \frac{n(s,a^i)}{n(s)} \hat{Q}(s,a^i)\right] && (84)\\
&= \sum_{i=1}^{|\mathcal{A}_s|} \frac{n(s,a^i)}{n(s)} \mathbb{E}\left[\hat{Q}(s,a^i)\right] && (85)\\
&= \sum_{i=1}^{|\mathcal{A}_s|} p(a^i|\pi_{visits}(s)) Q(s,a^i) && (86)\\
&= \mathbb{E}_{a\sim\pi_{visits}(s)}\left[Q(s,a)\right] && (87)\\
&= V^{\pi_{visits}}(s), && (88)
\end{aligned}$$

where we used the definition of π_{visits} and the induction hypothesis.

This explains the importance of "given tree structure" in the theorem statement - when changing the visitation counters and the according action selection policy at each state node, it results in a different weighting of action-value estimates, and hence could introduce a bias when considering the action-value function of a different policy at the root node of the search tree.

Inductive step 2: Action-value estimates $\hat{Q}$ are unbiased.

Induction hypothesis: Assume that $\hat{V}(s')$ is unbiased, i.e. $\mathbb{E}\left[\hat{V}(s'^{,i})\right] = V(s'^{,i})$, for all $s'^{,i} \in \mathcal{S}_{sa}$.

From assumption (MIS-II), we have that $w_i(s') = 0$ whenever $p_T(s'|s, a^i_{\text{prop}}) = 0$. Therefore, we can ignore the subset of the integration domain in which $p_T(s'|s, a^i_{\text{prop}}) = 0$ when calculating the expectation of $\hat{V}_f(s,a)$. From assumption (MIS-I), we have that $\sum_{i=1}^{|\mathcal{S}_{sa}|} w_i(s') = 1$ whenever the target function is non-zero, i.e. whenever $p_T(s'|s,a)(r(s,a,s') + \gamma V^{\pi}_{t+1}(s')) \neq 0$. Thus, it holds that at least one of the proposal distributions $p_T(s'|s, a^i_{\text{prop}})$ is non-zero at those points, and hence the estimator covers the whole domain of the target function. Thus, by using the tower property and the induction hypothesis, the expected future-value estimate is:

$$\begin{aligned}
&\mathbb{E}\left[\hat{V}_f(s,a)\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{|\mathcal{S}_{sa}|} w_i(s'^{,i}) \rho^i_a(s'^{,i}) \hat{V}(s'^{,i})\right] && (89)\\
&\overset{(1)}{=} \mathbb{E}\left[\mathbb{E}\left[\sum_{i=1}^{|\mathcal{S}_{sa}|} w_i(s'^{,i}) \rho^i_a(s'^{,i}) \hat{V}(s'^{,i}) \;\middle|\; s'^{,i}\right]\right] && (90)\\
&\overset{(2)}{=} \mathbb{E}\left[\sum_{i=1}^{|\mathcal{S}_{sa}|} w_i(s'^{,i}) \rho^i_a(s'^{,i}) \mathbb{E}\left[\hat{V}(s'^{,i}) \;\middle|\; s'^{,i}\right]\right] && (91)\\
&\overset{(3)}{=} \mathbb{E}\left[\sum_{i=1}^{|\mathcal{S}_{sa}|} w_i(s'^{,i}) \rho^i_a(s'^{,i}) V(s'^{,i})\right], && (92)
\end{aligned}$$

where transitions (1) and (2) are due to the tower property of expectations, and transition (3) follows from conditional independence and the induction hypothesis.

The expectation is calculated over all random variables $s'^{,i}$. We enumerate them $s'^{,1}, \ldots, s'^{,n}$ and corresponding weighting functions $w_1, \ldots, w_n$. Writing the explicit integral:

$$\begin{aligned}
&(92)\\
&= \int_{s'^{,1},\ldots,s'^{,n}} p(s'^{,1},\ldots,s'^{,n}|s, a^1_{\text{prop}},\ldots,a^n_{\text{prop}}) \\
&\quad \sum_{i=1}^{|\mathcal{S}_{sa}|} w_i(s'^{,i}) \rho^i_a(s'^{,i}) V(s'^{,i}) \,\mathrm{d}s'^{,1} \cdots \mathrm{d}s'^{,n}. && (93)
\end{aligned}$$

We simplify $p(s'^{,1},\ldots,s'^{,n}|s, a^1_{\text{prop}},\ldots,a^n_{\text{prop}}) = \prod_{j=1}^n p_T(s'^{,j}|s, a^j_{\text{prop}})$, as the state samples are independent. Hence, distributing the product over the sum yields:

$$\begin{aligned}
&(93)\\
&= \int_{s'^{,1},\ldots,s'^{,n}} \sum_{i=1}^{|\mathcal{S}_{sa}|} \Bigg(\Bigg(\prod_{j=1}^{n} p_T(s'^{,j}|s, a^j_{\text{prop}}) \Bigg) w_i(s'^{,i}) \\
&\qquad \cdot \rho^i_a(s'^{,i}) V(s'^{,i}) \Bigg) \mathrm{d}s'^{,1} \cdots \mathrm{d}s'^{,n}. && (94)
\end{aligned}$$

Switching order of integral and summation (linearity of the integral), we obtain:

$$\begin{aligned}
&(94)\\
&= \sum_{i=1}^{|\mathcal{S}_{sa}|} \int_{s'^{,1},\ldots,s'^{,n}} \Bigg(\prod_{j=1}^{n} p_T(s'^{,j}|s, a^j_{\text{prop}}) \Bigg) w_i(s'^{,i}) \\
&\qquad \cdot \rho^i_a(s'^{,i}) V(s'^{,i}) \,\mathrm{d}s'^{,1} \cdots \mathrm{d}s'^{,n}. && (95)
\end{aligned}$$

Now, for the i'th summand, the integration over transitions densities $j \neq i$ cancel (they sum to 1 as proper probability densities):

$$\begin{aligned}
&(95)\\
&= \sum_{i=1}^{|\mathcal{S}_{sa}|} \int_{s'^{,i}} p_T(s'^{,i} \mid s, a^i_{\text{prop}}) w_i(s'^{,i}) \rho^i_a(s'^{,i}) V(s'^{,i}) \\
&\cancel{\Bigg(\int_{s'^{,1},\ldots,s'^{,i-1},s'^{,i+1},\ldots,s'^{,n}} \prod_{j\neq i} p_T(s'^{,j} \mid s, a^j_{\text{prop}})} \\
&\cancel{\mathrm{d}s'^{,1} \cdots \mathrm{d}s'^{,i-1} \mathrm{d}s'^{,i+1} \cdots \mathrm{d}s'^{,n} \Bigg)} \cdot \mathrm{d}s'^{,i}. && (96)
\end{aligned}$$

We substitute the definition of $\rho_a^i(s'^{,i})$ to obtain:

(96)

$$= \sum_{i=1}^{|\mathcal{S}_{sa}|} \int_{s'^{,i}} \cancel{p_T(s'^{,i} \mid s, a^i_{\text{prop}})} w_i(s'^{,i}) \cdot \cancel{\frac{p_T(s'^{,i} \mid s, a)}{p_T(s'^{,i} \mid s, a^i_{\text{prop}})}} V(s'^{,i}) ds'^{,i} \quad (97)$$

$$= \sum_{i=1}^{|\mathcal{S}_{sa}|} \int_{s'^{,i}} w_i(s'^{,i}) p_T(s'^{,i} \mid s, a) V(s'^{,i}) \, \mathrm{d}s'^{,i}. \quad (98)$$

We can now rename all $s'^{,i}$ to s' as they share the same integration domain. Hence, we can switch again the order of summation and integration. By distributive laws, we combine shared terms $p_T(s' \mid s, a)V(s')$, and we finish the proof:

(98)

$$= \int_{s'} \sum_{i=1}^{|\mathcal{S}_{sa}|} (w_i(s') p_T(s' \mid s, a) V(s')) \, ds'. \quad (99)$$

$$= \int_{s'} \underbrace{\left(\sum_{i=1}^{|\mathcal{S}_{sa}|} w_i(s') \right)}_{=1 \text{ (MIS-I)}} p_T(s' \mid s, a) V(s') ds' \quad (100)$$

$$= \int_{s'} p_T(s' \mid s, a) V(s') ds' \quad (101)$$

$$= \mathbb{E}\left[V(s')\right]. \quad (102)$$

Note regarding bias in POMDPs: when using self-normalized particle-belief approximations instead of exact belief representations, a bias in the action-value estimate arises, which increases recursively with the tree depth. This bias can be bounded with finite-sample concentration inequalities. For an in-depth analysis of this bias we refer to Lim *et al.* [2023]. □

A.5 Derivations of MIS Tree Update Equations

In this subsection we show the derivation of the MIS Tree update equation, and also show their numerically stable forms using log likelihoods for probability values. We denote the regular and the weighted log-sum-exp function as LSE, differentiated by having one or two vector inputs, and they are defined as

$$LSE(\boldsymbol{x}_{1,\dots,n}) \triangleq \log\left(\sum_{i=1}^{n} \exp(\boldsymbol{x}_i) \right), \quad (103)$$

$$LSE(\boldsymbol{x}_{1,\dots,n}, \boldsymbol{y}_{1,\dots,n}) \triangleq \log\left(\sum_{i=1}^{n} \exp(\boldsymbol{x}_i) \cdot \boldsymbol{y}_i \right). \quad (104)$$

To handle negative $\boldsymbol{y}_{1,\dots,n}$, (104) can be defined to compute the log of absolute value and sign separately of the log argument, such that the sign is multiplied back after exponentiation if required. We also denote a vector with brackets $[\cdot]$.

We slightly modify the notations to an enumerated form in this subsection to later match vector notations. In these notations, the visitation counter relationships are:

$$n(s) = \sum_{i=1}^{|\mathcal{A}_s|} n(s, a^i), \quad n(s, a) = \sum_{i=1}^{|\mathcal{S}_{sa}|} n(s'^{,i})_{+1}, \quad (105)$$

the state value estimator:

$$\hat{V}(s) \triangleq \sum_{i=1}^{|\mathcal{A}_s|} \frac{n(s, a^i)}{n(s)} \tilde{Q}(s, a^i), \quad (106)$$

and the self normalized MIS estimator:

$$\eta(s, a) \triangleq \sum_{i=1}^{|\mathcal{S}_{sa}|} n(s'^{,i})_{+1} \rho_a^i(s'^{,i}), \quad (107)$$

$$\tilde{V}_f(s, a) \triangleq \sum_{i=1}^{|\mathcal{S}_{sa}|} n(s'^{,i})_{+1} \rho_a^i(s'^{,i}) \hat{V}(s'^{,i}) / \eta(s, a), \quad (108)$$

$$\tilde{r}(s, a) \triangleq \sum_{i=1}^{|\mathcal{S}_{sa}|} n(s'^{,i})_{+1} \rho_a^i(s'^{,i}) r(s, a, s'^{,i}) / \eta(s, a), \quad (109)$$

$$\tilde{Q}(s, a) \triangleq \tilde{r}(s, a) + \gamma \tilde{V}_f(s, a). \quad (110)$$

For brevity in this section, we denote $M \triangleq |\mathcal{S}_{sa}|$. We define the following vectors for each a given action node (s, a) in the MDP case, or $(\bar{b}, a)$ in the particle-belief MDP (for POMDPs) case:

1. Child state visitation counters.
   MDP:
   $$\boldsymbol{n} \triangleq (n(s'^{,i})_{+1})_{i=1}^{M}, \quad (111)$$
   particle-belief MDP:
   $$\boldsymbol{n} \triangleq (n(\bar{b}'^{,i})_{+1})_{i=1}^{M}. \quad (112)$$
2. Target transition likelihoods.
   MDP:
   $$\boldsymbol{p} = (p_T(s'^{,i}|s, a))_{i=1}^{M}, \quad (113)$$
   particle-belief MDP:
   $$\boldsymbol{p} = (p(\bar{b}'^{-,i}|\bar{b}, a))_{i=1}^{M}. \quad (114)$$
3. Proposal transition likelihoods.
   MDP:
   $$\boldsymbol{q} = (p_T(s'^{,i}|s, a^i_{\text{prop}}))_{i=1}^{M}, \quad (115)$$
   particle-belief MDP:
   $$\boldsymbol{q} = (p(\bar{b}'^{-,i}|\bar{b}, a^i_{\text{prop}}))_{i=1}^{M}. \quad (116)$$
4. Immediate rewards.
   MDP:
   $$\boldsymbol{r} = (r(s, a, s'^{,i}))_{i=1}^{M} \quad (117)$$
   particle-belief MDP:
   $$\boldsymbol{r} = (r(\bar{b}, a, \bar{b}'^{,i}))_{i=1}^{M}. \quad (118)$$

5. Future value estimates.
   MDP:
$$\boldsymbol{V} = (\hat{V}(s'^{,i}))_{i=1}^{M} \tag{119}$$
   particle-belief MDP:
$$\boldsymbol{V} = (\hat{V}(\bar{b}'^{,i}))_{i=1}^{M} \tag{120}$$

From here on, the particle-belief MDP case is the same up to different notations as the MDP case, and we will show only the latter. Using these notations the immediate reward and future value estimates can be compactly written:

$$\eta(s,a) = \mathbf{1}^T(\boldsymbol{n} \odot \boldsymbol{p} \oslash \boldsymbol{q}), \tag{121}$$
$$\tilde{r}(s,a) = \eta(s,a)^{-1}\mathbf{1}^T(\boldsymbol{n} \odot \boldsymbol{p} \odot \boldsymbol{r} \oslash \boldsymbol{q}), \tag{122}$$
$$\tilde{V}_f(s,a) = \eta(s,a)^{-1}\mathbf{1}^T(\boldsymbol{n} \odot \boldsymbol{p} \odot \boldsymbol{V} \oslash \boldsymbol{q}), \tag{123}$$

where $\odot$ and $\oslash$ are the Hadamard (element-wise) product and division operations respectively. Similarly, we denote Hadamard addition and subtraction as $\oplus$ and $\ominus$. We use a shortened notation $l(\cdot) \triangleq \log$, which can be kept in any base.

We store the following values:

1. $l(\eta)$.
2. Log likelihoods $l(\boldsymbol{p})$ and $l(\boldsymbol{q})$.
3. $\boldsymbol{n}'$ and previous visitation counters $\boldsymbol{n}$.
4. $\boldsymbol{V}'$ and previous future value estimates $\boldsymbol{V}$.

The equations for $\tilde{r}(s,a)$ are analogous to $\tilde{V}_f(s,a)$, and we will not show their derivation explicitly.

Action Backpropagation. Let $s'^{,j}$ be an updated node. Hence, we updated $\boldsymbol{n}_j$ to $\boldsymbol{n}'_j$ and $\boldsymbol{V}_j$ to $\boldsymbol{V}'_j$. We update $n(s,a)$ by (105) and perform

$$\eta'(s,a) = \eta(s,a) + \frac{\boldsymbol{p}_j}{\boldsymbol{q}_j}\left(\boldsymbol{n}'_j - \boldsymbol{n}_j\right), \tag{124}$$
$$\tilde{V}'_f(s,a) = \frac{\left(\eta(s,a)\tilde{V}_f(s,a) + \frac{\boldsymbol{p}_j}{\boldsymbol{q}_j}(\boldsymbol{n}'_j\boldsymbol{V}'_j - \boldsymbol{n}_j\boldsymbol{V}_j)\right)}{\eta'(s,a)}. \tag{125}$$

Therefore, the log-form update equations are:

$$l(\eta'(s,a)) = LSE\Big([l(\eta(s,a)), l(\boldsymbol{p}_j) - l(\boldsymbol{q}_j) + l(\boldsymbol{n}'_j - \boldsymbol{n}_j)]\Big) \tag{126}$$
$$\tilde{V}'_f(s,a) = \exp\Big(LSE\big([l(\eta(s,a)), l(\boldsymbol{p}_j) - l(\boldsymbol{q}_j)], [\tilde{V}_f(s,a), \boldsymbol{n}'_j\boldsymbol{V}'_j - \boldsymbol{n}_j\boldsymbol{V}_j]\big) - l(\eta'(s,a))\Big). \tag{127}$$

Proof. We start with the update for $\eta'(s,a)$:

$$\eta'(s,a) - \eta(s,a) = \sum_{i=1}^{M}\frac{\boldsymbol{n}'_i\boldsymbol{p}_i}{\boldsymbol{q}_i} - \sum_{i=1}^{M}\frac{\boldsymbol{n}_i\boldsymbol{p}_i}{\boldsymbol{q}_i} \tag{128}$$
$$= \frac{\boldsymbol{p}_j}{\boldsymbol{q}_j}(\boldsymbol{n}'_j - \boldsymbol{n}_j), \tag{129}$$

due to all terms being equal except for $\boldsymbol{n}'_j \neq \boldsymbol{n}_j$. Similarly, for $\tilde{V}_f(s,a)$:

$$\tilde{V}'_f(s,a) - \frac{\eta(s,a)}{\eta'(s,a)}\tilde{V}_f(s,a) = \frac{1}{\eta'(s,a)}\sum_{i=1}^{M}\frac{\boldsymbol{n}'_i\boldsymbol{p}_i\boldsymbol{V}'_i}{\boldsymbol{q}_i} - \frac{\eta(s,a)}{\eta'(s,a)}\frac{1}{\eta(s,a)}\sum_{i=1}^{M}\frac{\boldsymbol{n}_i\boldsymbol{p}_i\boldsymbol{V}_i}{\boldsymbol{q}_i} \tag{130}$$
$$= \frac{1}{\eta'(s,a)}\left(\sum_{i=1}^{M}\frac{\boldsymbol{n}'_i\boldsymbol{p}_i\boldsymbol{V}'_i}{\boldsymbol{q}_i} - \sum_{i=1}^{M}\frac{\boldsymbol{n}_i\boldsymbol{p}_i\boldsymbol{V}_i}{\boldsymbol{q}_i}\right) \tag{131}$$
$$= \frac{1}{\eta'(s,a)}\frac{\boldsymbol{p}_j}{\boldsymbol{q}_j}(\boldsymbol{n}'_j\boldsymbol{V}'_j - \boldsymbol{n}_j\boldsymbol{V}_j), \tag{132}$$

due to all terms being equal except for $\boldsymbol{V}'_j \neq \boldsymbol{V}_j$, $\boldsymbol{n}'_j \neq \boldsymbol{n}_j$. □

State Expansion. This case is the same as *action backpropagation*, where the updated index is a new summand at index $j = M+1$. The same equations apply with $n(s'^{,j}) = 0$.

Proof. We append new terms to all vectors: $\boldsymbol{n}_{M+1}$, $\boldsymbol{p}_{M+1}$, $\boldsymbol{V}_{M+1}$, $\boldsymbol{q}_{M+1}$. Update to $\eta(s,a)$:

$$\eta'(s,a) - \eta(s,a) = \sum_{i=1}^{M+1}\frac{\boldsymbol{n}'_i\boldsymbol{p}'_i}{\boldsymbol{q}'_i} - \sum_{i=1}^{M}\frac{\boldsymbol{n}_i\boldsymbol{p}_i}{\boldsymbol{q}_i} \tag{133}$$
$$= \frac{\boldsymbol{n}'_{M+1}\boldsymbol{p}'_{M+1}}{\boldsymbol{q}'_{M+1}}, \tag{134}$$

due to all terms being equal except for index $M+1$. This corresponds to (129) with $\boldsymbol{n}_{M+1} = 0$, after initializing $\boldsymbol{p}'_{M+1} = \boldsymbol{p}_{M+1}$ and $\boldsymbol{q}'_{M+1} = \boldsymbol{q}_{M+1}$ Similarly, for $\hat{V}_f(s,a)$:

$$\tilde{V}'_f(s,a) - \frac{\eta(s,a)}{\eta'(s,a)}\tilde{V}_f(s,a) = \frac{1}{\eta'(s,a)}\sum_{i=1}^{M+1}\frac{\boldsymbol{n}'_i\boldsymbol{p}_i\boldsymbol{V}'_i}{\boldsymbol{q}_i} - \frac{\eta(s,a)}{\eta'(s,a)}\frac{1}{\eta(s,a)}\sum_{i=1}^{M}\frac{\boldsymbol{n}_i\boldsymbol{p}_i\boldsymbol{V}_i}{\boldsymbol{q}_i} \tag{135}$$
$$= \frac{1}{\eta'(s,a)}\left(\sum_{i=1}^{M+1}\frac{\boldsymbol{n}'_i\boldsymbol{p}_i\boldsymbol{V}'_i}{\boldsymbol{q}_i} - \sum_{i=1}^{M}\frac{\boldsymbol{n}_i\boldsymbol{p}_i\boldsymbol{V}_i}{\boldsymbol{q}_i}\right) \tag{136}$$
$$= \frac{1}{\eta'(s,a)}\frac{\boldsymbol{p}'_{M+1}}{\boldsymbol{q}'_{M+1}}\boldsymbol{n}'_{M+1}\boldsymbol{V}'_{M+1}, \tag{137}$$

which corresponds to the expression in (132) with $\boldsymbol{n}_{M+1} = 0$. □

Action Update. Let $\breve{a}$ be the new action. Here, we have to calculate $\rho^i_{\breve{a}}(s')$ for all $s'^{,i} \in \mathcal{S}_{sa}$ in $O(M)$ time. We update

$$\rho^i_{\breve{a}}(s') = p_T(s'|s,\breve{a})/p_T(s'|s,a^i_{\text{prop}}), \tag{138}$$

and recalculate (108) and (107) with the new $\rho^i_{\breve{a}}(s')$, which are also $O(M)$ time operations. For the immediate reward,

new rewards must be computed for the new triplets $(s, \breve{a}, s')$, and the update is analogous.

Therefore, the log-form equations are:

$$l(\eta'(s,\breve{a})) = LSE(l(\boldsymbol{n}) \oplus l(\boldsymbol{p}') \ominus l(\boldsymbol{q})) \tag{139}$$

$$\tilde{V}'_f(s,\breve{a}) = \exp\left(LSE(l(\boldsymbol{p}') \ominus l(\boldsymbol{q}), \boldsymbol{n} \odot \boldsymbol{V}) - l(\eta'(s,\breve{a}))\right) \tag{140}$$

In case we linearize the weight updates (as in Equation (14)), instead of re-evaluating the density, we update the stored log-likelihoods $l(\boldsymbol{p})$ using the gradients. Let $\delta a = \breve{a} - a$. The updated log-target likelihoods are approximated by:

$$l(\boldsymbol{p}'_i) \approx l(\boldsymbol{p}_i) + \nabla_a \log p_T(s'^{,i}|s,a)^T \delta a. \tag{141}$$

This approximation avoids the expensive re-evaluation of $p_T(s'|s,a)$ (or $p(\bar{b}'^{-}|\bar{b},a)$ in POMDPs). The LSE aggregation steps for η' and $\tilde{V}'_f$ remain identical using the approximated $l(\boldsymbol{p}')$.

Proof. The update of $\boldsymbol{p}'$ corresponds to recomputing it either via exact re-evaluation or via the linearized approximation. The linearized approximation is derived from a first-order Taylor expansion of $l(\boldsymbol{p}_i)$ around a:

$$l(\boldsymbol{p}'_i) = \log(p_T(s'^{,i}|s,\breve{a})) \tag{142}$$

$$\approx \log(p_T(s'^{,i}|s,a)) + \nabla_a \log p_T(s'^{,i}|s,a)^T(\breve{a} - a) \tag{143}$$

$$= l(\boldsymbol{p}_i) + \nabla_a \log p_T(s'^{,i}|s,a)^T \delta a. \tag{144}$$

Therefore, the log-form update equations that follow match exactly the definitions:

$$\eta'(s,\breve{a}) = \sum_{i=1}^{M} \frac{\boldsymbol{n}_i \boldsymbol{p}'_i}{\boldsymbol{q}_i}, \tag{145}$$

$$\tilde{V}'_f(s,\breve{a}) = \frac{1}{\eta'(s,\breve{a})} \sum_{i=1}^{M} \frac{\boldsymbol{n}_i \boldsymbol{p}'_i \boldsymbol{V}_i}{\boldsymbol{q}_i}. \tag{146}$$

□

Terminal State Backpropagation. Let s be a terminal node. Its value estimate is based only on rollouts. Hence, for a new rollout value v', we perform a running average update

$$\hat{V}'(s) = \hat{V}(s) + \frac{n'(s) - n(s)}{n'(s)_{+1}}(v' - \hat{V}(s)), \tag{147}$$

where $n'(s)$ is $n(s)$ plus the number of rollouts (usually 1).

Proof. This is a simple running average update for the mean estimator:

$$\hat{V}(s) = \frac{1}{n(s)_{+1}} \sum_{i=1}^{n(s)_{+1}} v^i, \tag{148}$$

for all v^i rollout values from the terminal node s. □

Algorithm 2 AGMCTS - Full Pseudocode

procedure SIMULATE(s, d)
if ISTERMINAL(s) OR $d = 0$ **then**
  $v \leftarrow$ ROLLOUT(s) {*Rollout until terminal state*}
  UPDATETERMINAL(s, v) {*Eq.* (17)}
  return v
$a \leftarrow$ ACTIONPROGWIDEN(s) {*Vanilla or VPW*}
$addBranch \leftarrow$ ACTIONOPT (s, a, d)
if $|\mathcal{S}_{sa}| \leq k_o \cdot n(s,a)^{\alpha_o}$ OR $addBranch$ **then**
  $s', r \sim G(s,a)$ {$\bar{b}'^{-}, \bar{b}', r \sim G(\bar{b}, a)$ *in POMDPs*}
  $\mathcal{S}_{sa} \leftarrow \mathcal{S}_{sa} \cup \{s'\}$
  $v \leftarrow$ ROLLOUT$(s', d-1)$
else
  $s' \sim$ Unif$(\mathcal{S}_{sa})$, $r \leftarrow$ REWARD(s, a, s')
  $v \leftarrow$ SIMULATE$(s', d-1)$
UPDATEMIS(s, a, s', r) {*Eqs.* (15), (16), (18)}

procedure ACTIONOPT(s, a, d)
$addBranch \leftarrow$ FALSE
if ACTIONUPDATERULE(s, a, d) **then**
  for all $k = 1, \ldots, K_{\text{opt}}$ **do**
    $g_a^q \leftarrow \hat{\nabla}_a \tilde{Q}(s,a)$ {*Either of Eqs.* (157)-(164)}
    $\breve{a} \leftarrow$ CLIPNORM$(\breve{a}, T_{d_a}^{\max})$
    ACTIONUPDATE$(s, a, \breve{a})$ {*Eq.* (11)-(14), (18)}
    $addBranch \leftarrow addBranch \vee (\forall s'^{,i} \in \mathcal{S}_{s\breve{a}} : \rho^i_{\breve{a}}(s'^{,i}) \leq T^{add}_{\rho})$
return $addBranch$

Non-Terminal State Backpropagation. Let (s, a^j) be an updated node. Hence, we updated $n(s,a^j)$ to $n'(s,a^j)$ and $\tilde{Q}(s,a^j)$ to $\tilde{Q}'(s,a^j)$. We update $n'(s)$ by (105) and perform

$$\hat{V}'(s) = \frac{n(s)\hat{V}(s) + n'(s,a^j)\tilde{Q}'(s,a^j) - n(s,a^j)\tilde{Q}(s,a^j)}{n'(s)}. \tag{149}$$

Proof. Following the definition of the state value estimator (106), the updated estimator is:

$$\hat{V}'(s) = \sum_{i=1}^{|\mathcal{A}_s|} \frac{n'(s,a^i)}{n'(s)} \tilde{Q}'(s,a^i). \tag{150}$$

Therefore:

$$\hat{V}'(s) - \frac{n(s)}{n'(s)}\hat{V}(s)$$

$$= \sum_{i=1}^{|\mathcal{A}_s|} \frac{n'(s,a^i)}{n'(s)} \tilde{Q}'(s,a^i) - \frac{n(s)}{n'(s)} \sum_{i=1}^{|\mathcal{A}_s|} \frac{n(s,a^i)}{n(s)} \tilde{Q}(s,a^i) \tag{151}$$

$$= \frac{1}{n'(s)} \sum_{i=1}^{|\mathcal{A}_s|} (n'(s,a^i)\tilde{Q}'(s,a^i) - n(s,a^i)\tilde{Q}(s,a^i)). \tag{152}$$

All terms are equal except for the updated node j. Hence, we obtain

$$\hat{V}'(s) - \frac{n(s)}{n'(s)}\hat{V}(s)$$

$$= \frac{1}{n'(s)} \left(n'(s,a^j)\tilde{Q}'(s,a^j) - n(s,a^j)\tilde{Q}(s,a^j)\right). \tag{153}$$

□

A.6 Area Formula Assumptions and Discussion

Theorem 4.1. *[Negro, 2022] If f is locally Lipschitz, and it holds that $\operatorname{rank}(D_\xi f) = n_\xi$ a.e., then the induced probability measure over s' is absolutely continuous w.r.t. the Hausdorff measure $\mathcal{H}^{n_\xi}$ on $\mathbb{R}^{n_s}$, and its Radon-Nikodym derivative is*

$$p_T(s'|s,a) = \textstyle\sum_{\Xi^*} p_\xi(\xi^*|s,a)(J_{n_\xi} f(s,a,\xi^*))^{-1}, \quad (154)$$

for $\Xi^* = \{\xi^* : s' = f(s,a,\xi^*)\}$, and 0 when $\Xi^* = \emptyset$. The n_ξ-dimensional Jacobian is given by the Cauchy-Binet formula: $J_{n_\xi} f(s,a,\xi) = \sqrt{\det((D_\xi f)^\intercal \cdot (D_\xi f))}$.

We provide here a more detailed discussion of the assumptions in the Theorem and their implications for the transition density of the simulator.

f is locally Lipschitz. For every $\xi_0 \in \mathbb{R}^{n_\xi}$ there exist a neighborhood U_{ξ_0} and a constant $L_{\xi_0} < \infty$ such that

$$\|f(s,a,\xi_1) - f(s,a,\xi_2)\| \le L_{\xi_0}\|\xi_1 - \xi_2\|, \ \forall \xi_1, \xi_2 \in U_{\xi_0}. \quad (155)$$

This is only a local regularity assumption in the noise variable, and is weaker than requiring a single global Lipschitz constant for all ξ. Example: the map $f(s,a,\xi) = s + \xi^3$ for $s, a, \xi \in \mathbb{R}$ is locally Lipschitz in ξ but not globally Lipschitz.

Full rank condition: $\operatorname{rank}(D_\xi f) = n_\xi$ **a.e.** For fixed (s,a), the condition $\operatorname{rank}(D_\xi f) = n_\xi$ a.e. means that the noise coordinates used by the simulator generate n_ξ independent infinitesimal directions in the successor state, except on a null set. Thus the assumption is not a restriction that the ambient transition be full-dimensional in $\mathbb{R}^{n_s}$, and it is compatible with $n_\xi < n_s$; it rules out redundant or inactive noise coordinates in the chosen generative representation. This makes the condition a property of the simulator parameterization rather than of the intrinsic MDP transition law. Indeed, let $\mathcal{M}_{s,a} = \operatorname{supp}(p_T(\cdot|s,a))$ be the supported transition manifold, with intrinsic dimension m, and consider the identity generative model that samples $s' \sim p_T(\cdot|s,a)$ on $\mathcal{M}_{s,a}$ and returns $f^{\mathrm{id}}_{s,a}(s') = s'$. In any smooth local chart of $\mathcal{M}_{s,a}$, this model has rank m by construction, so a full-rank description exists on the supported manifold. The practical issue is whether the simulator exposes such non-redundant coordinates, together with the corresponding density w.r.t. $\mathcal{H}^m$, in a computable analytic form.

B AGMCTS Algorithm

The full AGMCTS algorithm description is outlined in Algorithm 2. It is based on MCTS with DPW, with new components highlighted in blue. We discuss several of the heuristics and optimizations that we found to be crucial for the performance of AGMCTS.

B.1 Monte Carlo Gradient Estimation

When estimating the action score gradient of the action-value function, we subsample the successor branches, or states of particle-beliefs when computing gradient estimates, greatly reducing computational cost.

Algorithm 3 Action Progressive Widening (APW)

procedure ACTIONPROGWIDEN(s)
if $|\mathcal{A}_s| \le k_a \cdot n(s)^{\alpha_a}$ **then**
  $a \sim \text{Unif}(\mathcal{A})$
  $\mathcal{A}_s \leftarrow \mathcal{A}_s \cup \{a\}$
  Initialize $Q(s,a)$ and $n(s,a)$
  return a
else
  {Select best action using UCB}
  return $\arg\max_{a \in \mathcal{A}_s} Q(s,a) + c\sqrt{\frac{\log n(s)}{n(s,a)}}$

Algorithm 4 Voronoi Progressive Widening (VPW)

procedure VORONOIPROGWIDEN(s)
if $|\mathcal{A}_s| \le k_a \cdot n(s)^{\alpha_a}$ **then**
  $a \leftarrow \text{VOO}(\mathcal{A}_s, \{Q(s,a')\}_{a' \in \mathcal{A}_s})$
  $\mathcal{A}_s \leftarrow \mathcal{A}_s \cup \{a\}$
  Initialize $Q(s,a)$ and $n(s,a)$
  return a
else
  {Select best action using UCB}
  return $\arg\max_{a \in \mathcal{A}_s} Q(s,a) + c\sqrt{\frac{\log n(s)}{n(s,a)}}$

procedure VOO($\mathcal{A}_s, Q_{vals}$)
$u \sim \text{Unif}[0,1]$
if $u \le \omega_{\text{VOO}}$ **or** $|\mathcal{A}_s| = 0$ **then**
  return $a \sim \text{Unif}(\mathcal{A})$ {Global Exploration}
else
  return BESTVORONOICELL($\mathcal{A}_s, Q_{vals}$) {Local Refinement}

procedure BESTVORONOICELL($\mathcal{A}_s, Q_{vals}$)
$a^* \leftarrow \arg\max_{a' \in \mathcal{A}_s} Q_{vals}(a')$
while true **do**
  $a \sim \mathcal{N}(a^*, \Sigma_{\text{VOO}})$ {Gaussian Rejection Sampling}
  if $\forall a_i \in \mathcal{A}_s \setminus \{a^*\}, D(a,a^*) \le D(a,a_i)$ **then**
    return a

Transition Model Log-Probability

Via automatic differentiation and the chain rule, we assume we can compute $\nabla_a \log p_T(s'|s,a)$ for any given transition triplet (s,a,s').

In POMDPs, we do not compute exact propagated belief probabilities $p(\bar{b}'^-|\bar{b},a)$, but rather approximate them via subsampling their gradient. We estimate (3) with K_b^∇ state particles via

$$\hat{\nabla}_a \log p(\bar{b}'^-|\bar{b},a) = \frac{J}{K_b^\nabla} \sum_{l=1}^{K_b^\nabla} \nabla_a \log p_T(s'^{-,j_l}|s^{j_l},a), \quad (156)$$

where indices j_l are sampled uniformly from $[1,\dots,J]$, where J is the particle count of $\bar{b}$ and $\bar{b}'^-$. Since this is a Monte Carlo approximation of the sum (23), the estimator (156) is unbiased.

Table 1: Hyperparameters optimized by CE in the evaluations of each algorithm in the D-Continuous Light-Dark, Mountain Car, Hill Car and Lunar Lander domains. The different subsets of algorithms to which each parameter applies are indicated in the "Algorithms" column. The "Tuning" column indicates whether the parameter was optimized via Cross-Entropy (CE) or set manually. Algorithm categories: AGMCTS includes AG-DPW, AG-VPW, AG-PFT-DPW, AG-PFT-VPW; PFT includes PFT-DPW, PFT-VPW, AG-PFT-DPW, AG-PFT-VPW; VPW includes VPW, AG-VPW, PFT-VPW, AG-PFT-VPW, VOMCPOW.

Param. Notation	Explanation	Algorithms	Tuning
c	UCT exploration bonus	All	CE
k_a	Action prog. widening factor	All	CE
α_a	Action prog. widening power	All	CE
k_o	Obs. prog. widening factor	All	CE
α_o	Obs. prog. widening power	All	CE
η_{Adam}	Adam step size (learning rate)	AGMCTS	CE
J	No. particles	PFT	Manual
$K_{rollout}$	No. states for rollout	PFT	Manual
ω_{VOO}	VOO global exploration prob.	VPW	Manual
Σ_{VOO}	VOO local refinement cov.	VPW	Manual
K_{opt}	No. grad. iterations	AGMCTS	Manual
$T_{d_a}^{\max}$	Max. *action update* step	AGMCTS	Manual
T_{ρ}^{add}	Imp. ratio add threshold	AGMCTS	Manual
T_{ρ}^{del}	Imp. ratio delete threshold	AGMCTS	Manual
K_b^{∇}	No. states for $\hat{\nabla}_a \log p_T$	AGMCTS	Manual
$n_{\max}^{\text{sims}}$	Max. simulation budget	All	Manual

Table 2: Hyperparameters used in the evaluations of each algorithm in the 2D-Continuous Light-Dark domain.

	DPW Variants		**AG Variants (Ours)**		**POMCP Variants**	
Param.	PFT-DPW	PFT-VPW	AG-PFT-DPW	AG-PFT-VPW	POMCPOW	VOMCPOW
c	1.01	1.02	1.96	3.16	0.86	1.45
k_a	7.68	7.16	5.70	4.29	0.46	0.36
α_a	0.52	0.48	0.47	0.48	0.77	0.75
k_o	8.90	9.03	8.05	4.51	0.16	$9.6\text{e}{-2}$
α_o	0.30	0.35	0.77	0.67	0.25	0.30
η_{Adam}	-	-	$5.3\text{e}{-8}$	$4.28\text{e}{-7}$	-	-

Action-Gradient Estimates of the Action-Value

In the MDP case, we use the baseline function $B(s) = \hat{V}(s)$ when estimating the action-gradient of the action-value function, essentially estimating advantage gradients. In the POMDP case, we use $B(\bar{b}) = \hat{V}(\bar{b})$ similarly.

When linearizing the importance weights update in Equation (14), we have to compute $\nabla_a \log p_T(s'^{,i}|s,a)$ (or $\hat{\nabla}_a \log p(\bar{b}^{-\prime,i}|\bar{b},a)$) for all $i = 1,\ldots,|\mathcal{S}_{sa}|$ to compute their updated importance weight. Since this is the most expensive term when computing $\hat{\nabla}\tilde{Q}(s,a)$, it makes sense to cache the transition log-gradient terms, and use all successor branches for gradient estimation. Therefore, we sum over all successor states $s' \in \mathcal{S}_{sa}$, weighted by the visitation count and the importance ratio:

$$\hat{\nabla}_a \tilde{Q}(s,a)$$

$$= \frac{1}{\eta(s,a)} \sum_{i=1}^{|\mathcal{S}_{sa}|} n(s'^{,i})_{+1}\rho_a^i(s'^{,i}) \cdot \Big(\nabla_a \log p_T(s'^{,i}|s,a) \cdot (r(s,a,s'^{,i}) + \gamma \hat{V}(s'^{,i}) - B(s)) + \nabla_a r(s,a,s'^{,i}) \Big), \tag{157}$$

and for POMDPs:

$$\hat{\nabla}_a \tilde{Q}(\bar{b},a) = \frac{1}{\eta(\bar{b},a)} \sum_{i=1}^{|\mathcal{S}_{sa}|} n(\bar{b}'^{,i})_{+1}\rho_a^i(\bar{b}^{-\prime,i}) \cdot \Big(\hat{\nabla}_a \log p(\bar{b}^{-\prime,i}|\bar{b},a) \cdot (r(\bar{b},a,\bar{b}'^{,i}) + \gamma \hat{V}(\bar{b}'^{,i}) - B(\bar{b})) + \nabla_a r(\bar{b},a,\bar{b}'^{,i}) \Big). \tag{158}$$

When using direct state sampling based on Theorem 1 Extended (Immediate Reward Gradient), we sample K_r^{∇} new successor states $s'^{,j_m}_{new} \sim p_T(\cdot|s,a)$, for $m = 1,\ldots,K_r^{\nabla}$, and

Table 3: Hyperparameters used in the evaluations of each algorithm in the 3D-Continuous Light-Dark domain.

	DPW Variants		AG Variants (Ours)		POMCP Variants	
Param.	PFT-DPW	PFT-VPW	AG-PFT-DPW	AG-PFT-VPW	POMCPOW	VOMCPOW
c	1.70	1.28	2.52	1.60	1.04	1.28
k_a	11.76	7.29	5.71	4.57	0.85	0.45
α_a	0.26	0.54	0.44	0.62	0.49	0.77
k_o	7.49	8.58	6.11	8.93	0.40	0.16
α_o	0.33	0.63	0.63	0.22	0.33	0.28
η_{Adam}	-	-	3.8e−8	2.9e−7	-	-

Table 4: Hyperparameters used in the evaluations of each algorithm in the 4D-Continuous Light-Dark domain.

	DPW Variants		AG Variants (Ours)		POMCP Variants	
Param.	PFT-DPW	PFT-VPW	AG-PFT-DPW	AG-PFT-VPW	POMCPOW	VOMCPOW
c	1.45	1.12	3.59	0.70	1.18	1.55
k_a	10.85	8.39	7.82	5.12	0.79	0.31
α_a	0.29	0.46	0.22	0.62	0.47	0.82
k_o	7.00	6.74	0.96	6.21	0.27	0.14
α_o	0.14	0.62	0.24	0.33	0.41	0.14
η_{Adam}	-	-	4.2e−8	2.8e−8	-	-

we compute:

$$\begin{aligned}
&\hat{\nabla}_a \hat{Q}(s,a) \\
&= \frac{1}{K_r^\nabla}\Big(\sum_{m=1}^{K_r^\nabla} \nabla_a \log p_T(s'^{,j_m}_{new}|s,a) \cdot r(s,a,s'^{,j_m}_{new}) \\
&\qquad\qquad + \nabla_a r(s,a,s'^{,j_m}_{new})\Big) \\
&+ \frac{1}{\eta(s,a)} \sum_{i=1}^{|\mathcal{S}_{sa}|} n(s'^{,i})_{+1} \rho_a^i(s'^{,i}) \cdot \nabla_a \log p_T(s'^{,i}|s,a) \\
&\qquad\qquad \cdot (\gamma \hat{V}(s'^{,i}) - B(s)),
\end{aligned} \tag{159}$$

In the POMDP case, we generate K_r^∇ posterior states by sampling from the belief $s^{j_m} \sim \bar{b}$, and sampling new states $s'^{,j_m}_{new} \sim p_T(\cdot|s^{j_m},a)$, for $m = 1,\dots,K_r^\nabla$, where J is the number of particles of $\bar{b}$:

$$\begin{aligned}
&\hat{\nabla}_a \hat{Q}(s,a) \\
&= \frac{1}{K_r^\nabla}\Big(\sum_{m=1}^{K_r^\nabla} \nabla_a \log p_T(s'^{,j_m}_{new}|s^{j_m},a) \cdot r(s^{j_m},a,s'^{,j_m}_{new}) \\
&\qquad\qquad + \nabla_a r(s^{j_m},a,s'^{,j_m}_{new})\Big) \\
&+ \frac{1}{\eta(s,a)} \sum_{i=1}^{|\mathcal{S}_{sa}|} n(\bar{b}'^{,i})_{+1} \rho_a^i(\bar{b}^{-\prime,i}) \cdot \hat{\nabla}_a \log p(\bar{b}^{-\prime,i}|\bar{b},a) \\
&\qquad\qquad \cdot (\gamma \hat{V}(\bar{b}'^{,i}) - B(\bar{b})).
\end{aligned} \tag{160}$$

Note: The following estimators describe exact importance weight updates. In practice, we have not used these in any of the reported experiments in Section C. We bring these estimators for completeness.

When performing exact importance weight updates (138), we sample successor branches to compute the gradient, rather than computing over all successor states. We sample K_O^∇ successor state indices i_k from $[1,\dots,|\mathcal{S}_{sa}|]$, weighted proportionally to the visitation count times the importance ratio, i.e. from the discrete distribution on $[1,\dots,|\mathcal{S}_{sa}|]$ with weights $\propto n(s'^{,i})_{+1} \cdot \rho_a^i(s'^{,i})$. Thus, the MC approximation of (28) becomes

$$\begin{aligned}
\hat{\nabla}_a \tilde{Q}(s,a) = \frac{1}{K_O^\nabla} \sum_{k=1}^{K_O^\nabla} &\nabla_a \log p_T(s'^{,i_k}|s,a) \\
&\cdot (r(s,a,s'^{,i_k}) + \gamma \hat{V}(s'^{,i_k}) - B(s)) + \nabla_a r(s,a,s'^{,i_k}),
\end{aligned} \tag{161}$$

and for POMDPs:

$$\begin{aligned}
\hat{\nabla}_a \tilde{Q}(\bar{b},a) = \frac{1}{K_O^\nabla} \sum_{k=1}^{K_O^\nabla} &\hat{\nabla}_a \log p(\bar{b}^{-\prime,i_k}|\bar{b},a) \\
&\cdot (r(\bar{b},a,\bar{b}'^{,i_k}) + \gamma \hat{V}(\bar{b}'^{,i_k}) - B(\bar{b})) + \nabla_a r(\bar{b},a,\bar{b}'^{,i_k}).
\end{aligned} \tag{162}$$

When using direct state sampling based on Theorem 1 Extended (Immediate Reward Gradient), we similarly sample K_r^∇ new successor states $s'^{,j_m}_{new} \sim p_T(\cdot|s,a)$, for $m = 1,\dots,K_r^\nabla$, and we compute:

$$\begin{aligned}
&\hat{\nabla}_a \hat{Q}(s,a) \\
&= \frac{1}{K_r^\nabla}\Big(\sum_{m=1}^{K_r^\nabla} \nabla_a \log p_T(s'^{,j_m}_{new}|s,a) \cdot r(s,a,s'^{,j_m}_{new}) \\
&\qquad\qquad + \nabla_a r(s,a,s'^{,j_m}_{new})\Big) \\
&+ \frac{1}{K_O^\nabla} \sum_{k=1}^{K_O^\nabla} \nabla_a \log p_T(s'^{,i_k}|s,a) \cdot (\gamma \cdot (\hat{V}(s'^{,i_k})) - B(s)).
\end{aligned} \tag{163}$$

Table 5: Hyperparameters used in the evaluations of each algorithm in the Two-Agent 2D-Continuous Light-Dark domain.

	DPW Variants		AG Variants (Ours)		POMCP Variants	
Param.	PFT-DPW	PFT-VPW	AG-PFT-DPW	AG-PFT-VPW	POMCPOW	VOMCPOW
c	3.72	0.93	9.33	6.7e−3	4.12	1.76
k_a	9.64	9.24	0.14	7.5e−4	4.06	4.80
α_a	0.08	0.49	0.26	1.00	0.30	0.52
k_o	2.89	8.14	10.81	18.99	3.97	1.70
α_o	0.40	0.43	0.50	1.00	0.37	0.52
η_{Adam}	-	-	1.2e−2	0.90	-	-

Table 6: Performance (Mean ± SEM) and Runtimes in 2D-Continuous Light-Dark.

Simulations	**DPW Variants**		**AG Variants (Ours)**		**POMCP Variants**	
n^{sims} / $n^{\text{sims}}_{\text{POMCPOW}}$	PFT-DPW	PFT-VPW	AG-PFT-DPW	AG-PFT-VPW	POMCPOW	VOMCPOW
50 / 400	1.87 ± 0.11 0.017s	2.65 ± 0.11 0.030s	3.33 ± 0.10 4.726s	$\mathbf{4.03} \pm 0.09$ 5.439s	2.99 ± 0.11 0.095s	$3.69^{\dagger} \pm 0.10$ 0.228s
89 / 712	3.08 ± 0.11 0.006s	$4.30^{\dagger} \pm 0.09$ 0.006s	4.00 ± 0.09 0.012s	$4.35^{\dagger} \pm 0.09$ 0.013s	3.84 ± 0.10 0.006s	$\mathbf{4.63} \pm 0.10$ 0.006s
158 / 1264	4.11 ± 0.10 0.011s	5.10 ± 0.09 0.011s	4.40 ± 0.08 0.032s	4.43 ± 0.09 0.029s	3.15 ± 0.11 0.018s	$\mathbf{5.56} \pm 0.08$ 0.016s
281 / 2248	4.69 ± 0.09 0.017s	5.34 ± 0.09 0.018s	4.59 ± 0.08 0.103s	4.77 ± 0.08 0.075s	5.16 ± 0.09 0.019s	$\mathbf{6.05} \pm 0.08$ 0.020s
500 / 4000	5.29 ± 0.08 0.027s	$\mathbf{5.98} \pm 0.08$ 0.027s	4.86 ± 0.08 0.316s	5.07 ± 0.08 0.191s	$5.73^{\dagger} \pm 0.08$ 0.033s	$5.81^{\dagger} \pm 0.08$ 0.033s

And again in the POMDP case, we generate K_r^{∇} posterior states by sampling from the belief $s^{j_m} \sim \bar{b}$, and sampling new states $s'^{,j_m}_{new} \sim p_T(\cdot|s^{j_m}, a)$, for $m = 1, \ldots, K_r^{\nabla}$, where J is the number of particles of $\bar{b}$:

$$\begin{aligned}
&\hat{\nabla}_a \hat{Q}(\bar{b}, a) \\
&= \frac{1}{K_r^{\nabla}} \Big(\sum_{m=1}^{K_r^{\nabla}} \nabla_a \log p_T(s'^{,j_m}_{new}|s^{j_m}, a) \cdot r(s^{j_m}, a, s'^{,j_m}_{new}) \\
&\qquad\qquad + \nabla_a r(s^{j_m}, a, s'^{,j_m}_{new}) \Big) \\
&+ \frac{1}{K_O^{\nabla}} \sum_{k=1}^{K_O^{\nabla}} \hat{\nabla}_a \log p(\bar{b}^{-\prime,i_k}|\bar{b}, a) \cdot (\gamma \hat{V}(\bar{b}'^{,i_k}) - B(\bar{b})).
\end{aligned} \tag{164}$$

B.2 Action Update Rule

Since we compute advantage gradients, we only perform *action update* when we have more than one successor state in order to have a meaningful estimate of the gradient. In the case of a single successor, the advantage is always zero, leading to zero gradient estimates (unless the reward is action-dependent). Hence, to save computational cost, we only perform *action update* when $|\mathcal{S}_{sa}| >= K^{\min}_{\text{child}}$, where we set $K^{\min}_{\text{child}} = 2$ in all our experiments.

B.3 Gradient Optimization

We've found throughout our experiments that the Adam optimizer [Kingma and Ba, 2015] that normalizes the step size was crucial for consistent behavior due to the high variance of the gradient estimates. We've fixed as a hyperparameter K_{opt} the number of consecutive gradient optimization iterations before *simulation*. Together with the step size, this controls a trade-off between accuracy and computational complexity. To further ensure moderate step-sizes, we also set a max-norm threshold on the final action update $\|a - \breve{a}\| < T^{\max}_{d_a}$, and scaling down the step norm to $T^{\max}_{d_a}$ if necessary.

In some domains we used an exponential decay on the step size after the Adam step size. The calculation for the update of the accumulated action is done as:

$$\begin{aligned}
\delta_{Adam} &= \text{ADAM}(a, g_a^q), \\
\lambda &= \max\{0.999^T, 0.1\}, \\
\breve{a} &= a + \lambda \delta_{Adam},
\end{aligned}$$

where T is the number of gradient optimizations performed at this particular action node.

For the Adam optimizer, we used the standard hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$. The learning rate η_{Adam} was determined by cross entropy optimization (described in Section C), in conjunction with the optimization over the other algorithm hyperparameters.

B.4 Thresholds For Adding/Deleting State Nodes

To save computational overhead, we would like to ensure that we have relevant samples after several action updates. After *action update* to action a, we delete child state $s'^{,i}$ from $\mathcal{S}_{sa}$ if its importance ratio satisfies $\rho_a^i(s'^{,i}) < T_\rho^{del}$. Additionally, if $\rho_a^i(s'^{,i}) < T_\rho^{add}$ for all $s'^{,i} \in \mathcal{S}_{sa}$, we force sampling a

Table 7: Performance (Mean ± SEM) and Runtimes in 3D-Continuous Light-Dark.

Simulations	**DPW Variants**		**AG Variants (Ours)**		**POMCP Variants**	
n^{sims} / $n^{\text{sims}}_{\text{POMCPOW}}$	PFT-DPW	PFT-VPW	AG-PFT-DPW	AG-PFT-VPW	POMCPOW	VOMCPOW
50 / 566	$1.10^{\dagger} \pm 0.09$ 0.020s	0.16 ± 0.09 0.031s	$\mathbf{1.29} \pm 0.09$ 4.367s	0.56 ± 0.09 0.026s	$1.11^{\dagger} \pm 0.09$ 0.108s	$1.21^{\dagger} \pm 0.10$ 0.196s
89 / 1007	1.92 ± 0.09 0.013s	1.13 ± 0.09 0.015s	$1.97^{\dagger} \pm 0.09$ 0.021s	$2.01^{\dagger} \pm 0.10$ 3.673s	1.61 ± 0.09 0.011s	$\mathbf{2.33} \pm 0.10$ 0.012s
158 / 1788	2.55 ± 0.09 0.020s	2.62 ± 0.09 0.023s	2.51 ± 0.09 0.048s	$2.83^{\dagger} \pm 0.09$ 0.032s	2.39 ± 0.09 0.027s	$\mathbf{3.10} \pm 0.10$ 0.024s
281 / 3179	2.94 ± 0.09 0.035s	3.48 ± 0.09 0.040s	3.20 ± 0.08 0.125s	3.50 ± 0.08 0.063s	2.77 ± 0.09 0.031s	$\mathbf{4.26} \pm 0.09$ 0.035s
500 / 5657	3.52 ± 0.08 0.068s	3.90 ± 0.09 0.073s	3.34 ± 0.08 0.315s	4.17 ± 0.08 0.154s	3.20 ± 0.08 0.060s	$\mathbf{4.80} \pm 0.08$ 0.063s

Table 8: Performance (Mean ± SEM) and Runtimes in 4D-Continuous Light-Dark.

Simulations	**DPW Variants**		**AG Variants (Ours)**		**POMCP Variants**	
n^{sims} / $n^{\text{sims}}_{\text{POMCPOW}}$	PFT-DPW	PFT-VPW	AG-PFT-DPW	AG-PFT-VPW	POMCPOW	VOMCPOW
50 / 800	-0.28 ± 0.07 0.028s	-0.89 ± 0.06 0.038s	$-0.16^{\dagger} \pm 0.07$ 3.440s	-0.66 ± 0.06 0.031s	$\mathbf{0.07} \pm 0.06$ 0.096s	$-0.10^{\dagger} \pm 0.07$ 0.195s
89 / 1424	$0.30^{\dagger} \pm 0.08$ 0.031s	$0.34^{\dagger} \pm 0.08$ 0.029s	$\mathbf{0.41} \pm 0.07$ 0.038s	0.05 ± 0.07 3.322s	0.09 ± 0.07 0.020s	$0.27^{\dagger} \pm 0.08$ 0.020s
158 / 2528	1.03 ± 0.08 0.049s	0.86 ± 0.08 0.045s	$1.09^{\dagger} \pm 0.07$ 0.067s	$1.15^{\dagger} \pm 0.08$ 0.061s	1.01 ± 0.07 0.035s	$\mathbf{1.39} \pm 0.08$ 0.036s
281 / 4496	1.59 ± 0.08 0.078s	1.43 ± 0.08 0.079s	1.49 ± 0.07 0.143s	$\mathbf{2.43} \pm 0.08$ 0.132s	1.23 ± 0.07 0.056s	$2.34^{\dagger} \pm 0.08$ 0.064s
500 / 8000	1.98 ± 0.08 0.151s	2.28 ± 0.08 0.155s	2.10 ± 0.07 0.302s	$2.97^{\dagger} \pm 0.08$ 0.318s	1.61 ± 0.08 0.141s	$\mathbf{3.04} \pm 0.08$ 0.170s

new posterior node, overriding the progressive widening limitation. The deletion threshold avoids spending computation on states with low contribution to $\tilde{Q}(s, a)$, while the addition threshold ensures that at least one relevant sample is available after an action update.

B.5 Action Sampling Heuristics

In this section, we describe the action progressive widening strategies used to handle continuous action spaces. We detail the vanilla Action Progressive Widening (APW) used in algorithms like POMCPOW [Sunberg and Kochenderfer, 2018], and the Voronoi Progressive Widening (VPW) proposed by Lim *et al.* [Lim *et al.*, 2021].

Action Progressive Widening

Action Progressive Widening (APW) limits the branching factor of the search tree by progressively adding new action branches as the number of visits to the state node increases. New actions are sampled uniformly from the action space $\mathcal{A}$.

Voronoi Progressive Widening

VPW generalizes APW by utilizing Voronoi Optimistic Optimization (VOO) [Kim *et al.*, 2020] to guide the generation of new actions. Instead of sampling uniformly, VPW uses the existing action-value estimates to sample new actions from the Voronoi cell of the current best action with high probability, balancing exploration and exploitation during the widening phase.

The VOO procedure samples a new action by first selecting the best existing action $a^* = \arg\max_{a \in \mathcal{A}_s} Q(s, a)$, and then sampling a new candidate a from a distribution centered at a^* such that a lies within the Voronoi cell of a^*. With a probability ω_{VOO}, it samples uniformly from $\mathcal{A}$ to ensure global exploration. In our implementation, we used a Gaussian distribution centered at a^* with covariance Σ_{VOO} for sampling new actions within the Voronoi cell.

C Experiments

The highest mean result in each row is bolded. A dagger (†) indicates a mean that overlaps with a higher mean by at most 2 SEM.

C.1 Experimental Method

In a high-level overview, we conducted the experiments for each domain in the following manner:

1. Constant hyperparameters were defined for each algorithm, including maximum simulation budget $n^{\text{sims}}_{\max}$, particle count J (in POMDPs), and rollout policy.

Table 9: Performance (Mean ± SEM) and Runtimes in Two-Agent 2D-Continuous Light-Dark.

Simulations	**DPW Variants**		**AG Variants (Ours)**		**POMCP Variants**	
n^{sims} / $n^{\text{sims}}_{\text{POMCPOW}}$	PFT-DPW	PFT-VPW	AG-PFT-DPW	AG-PFT-VPW	POMCPOW	VOMCPOW
50 / 800	1.10 ± 0.11 0.035s	-0.01 ± 0.10 0.045s	$\mathbf{2.14} \pm 0.11$ 12.265s	$1.82^{\dagger} \pm 0.11$ 11.980s	$1.98^{\dagger} \pm 0.10$ 0.140s	1.66 ± 0.10 0.241s
89 / 1424	1.47 ± 0.11 0.036s	0.43 ± 0.10 0.037s	$1.81^{\dagger} \pm 0.10$ 0.057s	$1.58^{\dagger} \pm 0.10$ 0.056s	$\mathbf{1.96} \pm 0.10$ 0.020s	$1.56^{\dagger} \pm 0.10$ 0.021s
158 / 2528	1.44 ± 0.11 0.063s	1.19 ± 0.11 0.059s	$\mathbf{2.31} \pm 0.11$ 0.109s	$2.17^{\dagger} \pm 0.10$ 0.108s	$2.13^{\dagger} \pm 0.10$ 0.040s	$2.27^{\dagger} \pm 0.11$ 0.042s
281 / 4496	1.70 ± 0.10 0.088s	1.55 ± 0.11 0.096s	$\mathbf{2.72} \pm 0.11$ 0.214s	$2.48^{\dagger} \pm 0.10$ 0.208s	2.28 ± 0.10 0.065s	$2.41^{\dagger} \pm 0.11$ 0.076s
500 / 8000	1.91 ± 0.10 0.160s	1.67 ± 0.11 0.173s	$\mathbf{2.84} \pm 0.10$ 0.447s	1.18 ± 0.10 0.464s	1.83 ± 0.10 0.141s	$2.57^{\dagger} \pm 0.11$ 0.169s

Table 10: Manual hyperparameters in the Mountain/Hill Car domains.

Param.	**Mountain MDP/POMDP**	**Hill MDP/POMDP**
J	- / 30	- / 30
J^{PF}	- / 200	- / 200
$K_{rollout}$	1 / 5	1 / 5
K_{opt}	3	3 / 2
$T^{\max}_{d_a}$	σ_ξ	σ_ξ
T^{add}_{ρ}	1.0 / 0.99	1.0 / 0.99
T^{del}_{ρ}	0.5 / 1e−8	0.5 / 0.01
K^{∇}_{b}	- / 3	- / 3

Table 11: Hyperparameters used in the evaluations of each algorithm in the Mountain Car MDP domain.

	DPW Variants		**AG Variants (Ours)**	
Param.	DPW	VPW	AG-DPW	AG-VPW
c	112.20	116.80	0.0	39.90
k_a	6.13	2.09	5.02	9.08
α_a	0.60	0.72	0.67	2.3e−2
k_o	0.24	0.28	0.20	3.38
α_o	0.36	0.62	0.57	0.54
η_{Adam}	-	-	4.0e−4	0.11

2. A cross-entropy (CE) optimization was performed over the UCB and DPW parameters for each algorithm, for 50 iterations. For AGMCTS, the hyperparameters were optimized jointly with Adam's step size parameter η_{Adam}.
3. For the best performing hyperparameters of each algorithm of the 50 CE iterations, we ran 1000 simulations of the algorithm with the same seeds between algorithms, at the simulation budgets $n^{\text{sims}} = [10^{-1}, 10^{-0.75}, 10^{-0.5}, 10^{-0.25}, 1.0] \cdot n^{\text{sims}}_{\max}$, where $n^{\text{sims}}_{\max}$ is the maximum simulation budget allowed for the domain.

The simulation jobs were run on a distributed CPU cluster, using the Julia programming language with Distributed.jl multi-processing tools. Each simulation was run on a single core, without any parallelization in the solver itself. The cluster's machines feature the following processors:

1. Intel Xeon Gold 6230 Processor – up to 3.9 GHz clock speed.
2. Intel Xeon Platinum 8358 Processor – up to 3.4 GHz clock speed.
3. AMD EPYC 9654 Processor – up to 3.7 GHz clock speed.

The implementation of AGMCTS was written in the POMDPs.jl [Egorov *et al.*, 2017] framework. We took the double progressive widening (DPW) [Couëtoux *et al.*, 2011] implementation from the MCTS.jl [JuliaPOMDP, 2025a] package, particle belief implementations from the ParticleFilters.jl [JuliaPOMDP, 2024] package, and the POMCPOW [Sunberg and Kochenderfer, 2018] implementation from the POMCPOW.jl [JuliaPOMDP, 2025b] package. The implementation of Voronoi progressive widening (VPW) [Lim *et al.*, 2021] was adapted from the codebase of the original authors.

The following algorithms were compared in the MDP domains:

- **DPW:** MCTS with Double Progressive Widening (DPW), where the action sampling is uniform via Action Progressive Widening (APW).
- **VPW:** MCTS with Voronoi Progressive Widening (VPW), where the action sampling is via Voronoi Optimistic Optimization (VOO).
- **AG-DPW:** AGMCTS with DPW and uniform action sampling via APW.
- **AG-VPW:** AGMCTS with VPW and action sampling via VOO.

The following algorithms were compared in the POMDP domains:

- **PFT-DPW:** Particle Filter Tree (PFT) with Double Progressive Widening (DPW), where the action sampling is uniform via Action Progressive Widening (APW).

Table 12: Hyperparameters used in the evaluations of each algorithm in the Mountain Car POMDP domain.

	DPW Variants		AG Variants (Ours)		POMCP Variants	
Param.	PFT-DPW	PFT-VPW	AG-PFT-DPW	AG-PFT-VPW	POMCPOW	VOMCPOW
c	119.06	139.54	43.36	46.97	144.46	70.89
k_a	5.40	8.96	3.11	3.10	3.93	5.23
α_a	0.87	0.76	2.6e−2	2.9e−2	0.64	0.67
k_o	0.92	3.23	5.69	5.63	0.47	0.37
α_o	0.48	0.30	0.46	0.58	9.8e−2	0.48
η_{Adam}	-	-	1.7e−2	1.4e−2	-	-

Table 13: Hyperparameters used in the evaluations of each algorithm in the Hill Car MDP domain.

	DPW Variants		AG Variants (Ours)	
Param.	DPW	VPW	AG-DPW	AG-VPW
c	177.99	135.07	169.92	173.43
k_a	6.73	3.79	6.66	1.28
α_a	0.62	0.71	0.37	0.54
k_o	0.52	0.59	7.44	6.39
α_o	0.26	0.72	0.32	0.26
η_{Adam}	-	-	4.6e−6	5.8e−5

- **PFT-VPW:** Particle Filter Tree (PFT) with Voronoi Progressive Widening (VPW), where the action sampling is via Voronoi Optimistic Optimization (VOO).
- **AG-PFT-DPW:** AGMCTS with DPW and uniform action sampling via APW.
- **AG-PFT-VPW:** AGMCTS with VPW and action sampling via VOO.
- **POMCPOW:** Partially Observable Monte Carlo Planning with Observation Widening (POMCPOW), using uniform action sampling via APW.
- **VOMCPOW:** Variant of POMCPOW using action sampling via VOO.

For the action-gradient algorithms, automatic differentiation was mostly done via Enzyme.jl [Moses and Churavy, 2020], and we used ForwardDiff.jl [Revels *et al.*, 2016] overloaded with SciMLSensitivity.jl for ODE solution sensitivity [Rackauckas *et al.*, 2020].

The following reporting conventions apply to all scenarios and results presented in this appendix. Algorithm-specific hyperparameters and their notations are summarized in Table 1, while domain-specific values are listed in the respective tables. Reported numbers are rounded to two significant digits. Tabulated performance metrics reflect ± 1 standard error (SEM). The highest mean performance in each row is highlighted in **bold**, and a dagger (†) indicates results with confidence intervals (± 2 SEM) overlapping with the best result. The plots in Figure 2 depict ± 2 standard deviations. Runtimes (in seconds) are indicated in smaller font below the performance values; we note that these runtimes are artificially high for small simulation counts due to Julia's precompilation overhead.

Simulation Budget

Since POMCPOW is a state-trajectory algorithm, it is incomparable in terms of simulation budgets to belief-trajectory algorithms like PFT-DPW and AGMCTS. In the POMDP scenarios, we measured empirically the runtime of PFT-DPW at $n_{\max}^{\text{sims}}$ and J, and set a larger simulation budget for POMCPOW $n_{\max,\text{POMCPOW}}^{\text{sims}}$ to match the runtime of PFT-DPW.

Rollout Budget

In order to save computation time and not compute a partially observable rollout, the rollout computation for a new particle belief was done by drawing $K_{rollout}$ particles from the belief, and computing the rollout policy based on the mean state, while calculating the rollout estimate based on the mean of the returns of the $K_{rollout}$ particles. For POMCPOW, since every new node is expanded initially only with a single particle, the rollout was computed with that particle. This is a disadvantage for POMCPOW as it made the rollout returns less meaningful, however it seemed to have achieved consistently better results than PFT-DPW despite that. In the MDP domains, we have set $K_{rollout} = 1$ similarly to POMCPOW.

Cross-Entropy (CE) Optimization

The algorithm hyperparameters were first determined by cross entropy optimization (CE) [Rubinstein and Kroese, 2004], largely following [Botev *et al.*, 2013], using Gaussian distributions over the continuous DPW hyperparameters:

1. UCT exploration bonus c.
2. Action progressive widening parameters k_a and α_a.
3. Observation progressive widening parameters k_o and α_o.

For AGMCTS, we also optimized over the Adam learning rate η_{Adam}. To ensure that the obtained parameters result in algorithms that do not cancel out the action gradients, we constrained the CE search space to the following condition:

$$\eta_{\text{Adam}} \cdot K_{\text{opt}} \left(N - \text{ceil}\left(\left(\frac{K_{\text{child}}^{\min}}{k_o} \right)^{1/\alpha_o} \right) \right) \geq 0.01 \cdot T_{d_a}^{\max}$$

This condition states that the number of action gradient optimization steps after $K_{\min}^{\text{child}}$, assuming all of the simulation budget has been put into a single action branch at the root, multiplied by the step size η_{Adam}, should be sufficient to reach at least 1% of the maximum allowed action update distance $T_{d_a}^{\max}$. CE parameter samples that did not satisfy this condition were rejected. In the domains in which AGMCTS algorithms obtained worse performance than their non-gradient

Table 14: Hyperparameters used in the evaluations of each algorithm in the Hill Car POMDP domain.

	DPW Variants		AG Variants (Ours)		POMCP Variants	
Param.	PFT-DPW	PFT-VPW	AG-PFT-DPW	AG-PFT-VPW	POMCPOW	VOMCPOW
c	162.86	5.2e−7	131.41	136.45	101.51	70.13
k_a	10.31	4.80	5.55	1.95	9.94	7.44
α_a	0.56	0.49	0.30	0.55	0.18	0.36
k_o	0.85	8.15	9.88	13.31	8.56	6.81
α_o	0.53	0.32	0.58	0.42	0.84	0.73
η_{Adam}	-	-	8.1e−6	1.5e−5		

Table 15: Performance (Mean ± SEM) and Runtimes in Mountain Car MDP.

Simulations	**DPW Variants**		**AG Variants (Ours)**	
n^{sims}	DPW	VPW	AG-DPW	AG-VPW
50	-52.00 ± 0.17 0.007s	-50.50 ± 0.29 0.008s	-52.00 ± 0.17 0.009s	$\mathbf{-4.30} \pm 1.17$ 0.036s
89	-46.18 ± 0.68 0.004s	-50.33 ± 0.31 0.004s	-51.66 ± 0.25 0.007s	$\mathbf{7.26} \pm 1.03$ 0.015s
158	-47.46 ± 0.56 0.006s	-50.86 ± 0.27 0.006s	$12.55^{\dagger} \pm 1.03$ 0.056s	$\mathbf{16.16} \pm 0.79$ 0.027s
281	-45.71 ± 0.64 0.010s	24.42 ± 0.09 0.009s	$\mathbf{29.68} \pm 0.06$ 0.060s	21.27 ± 0.57 0.065s
500	24.24 ± 0.38 0.016s	24.33 ± 0.05 0.016s	$\mathbf{29.97} \pm 0.06$ 0.148s	25.94 ± 0.12 0.152s

counterparts, we observed that cancelling this condition led to convergence to hyperparameters that effectively disabled the action gradients, and the performance matched that of the DPW/VPW counterpart algorithm. This suggests that in these domains, the action gradients were not beneficial to the planning performance.

We used 150 parameter samples, 40 simulations for each parameter sample to determine its mean return, used 30 elite samples to fit the new distribution, and smoothing the new distribution's parameters:

$$\mu_{\text{new}} = \alpha_\mu \mu_{\text{new}} + (1 - \alpha_\mu)\mu_{\text{old}},$$
$$\Sigma_{\text{new}} = \alpha_\Sigma \Sigma_{\text{new}} + (1 - \alpha_\Sigma)\Sigma_{\text{old}},$$

where we chose $\alpha_\mu = 0.8, \alpha_\Sigma = 0.5$.

We performed 50 CE iterations for each algorithm in each dimension, with reasonable initial guesses for the starting parameters based on the literature and manual experiments. The hyperparameters chosen were those from the CE iteration achieving the maximal mean of mean returns over the 30 elite samples. After choosing the parameters, we reported the mean return over 1000 scenarios, with ± 2 standard errors around the mean.

C.2 dD-Continuous Light-Dark POMDP.

Problem Description

In this domain, $\mathcal{S} = \mathcal{O} = \mathbb{R}^d$. The agent starts at a random position on the sphere of radius r_0 centered at the origin: $S_{r_0}^{d-1}$. The goal position is located at $s_g = (\mathbf{0}_{d-1}, r_g)$. A beacon is located to the side at $s_b = (r_b, \mathbf{0}_{d-1})$. The constants we took are $r_a = 1.5$, $r_g = r_b = 2.5$, $r_0 = 0.5$. $T_{\text{goal}} = 0.2$ is the goal tolerance, and the POMDP terminates if $\|s - s_g\| < T_{\text{goal}}$, or after $L = 6$ time steps. We set $\gamma = 0.99$.

The action space is $B_{r_a}(\mathbf{0}_d)$. The transition model is a simple Gaussian with added noise: $s' \sim \mathcal{N}(s + a, \sigma_T^2 \mathbf{I})$ with $\sigma_T = 0.025$.

The observations are the agent's relative position to the beacon, with a Gaussian added noise that indreases with distance: $o \sim \mathcal{N}(s - s_b, (\sigma_O(\|s - s_b\|))^2 \mathbf{I})$, with noise function $\sigma_O(x) = \min\{\sigma_O^{\max}, k_{\sigma_O}(x + x^{\alpha_{\sigma_O}})\}$, for $\sigma_O^{\max} = 15$, $k_{\sigma_O} = 0.01$, $\alpha_{\sigma_O} = 8$.

The reward function is $r(s, a, s') = R_{goal} \exp(-\frac{1}{2}(\frac{d}{0.5T})^2) - R_{moat} \exp(-\frac{1}{2}(\frac{d-5T}{T})^2) - R_{dist} d^2$, where $d = \|s - s_g\|$. We took constants R_{goal}=10, R_{moat}=2, $R_{dist} = 0.02$. This reward function only depends on the posterior state. The agent needs to carefully target the goal as the reward decreases quickly from the center, and advancing at random towards the goal will incur a penalty due to the moat.

To simulate more difficult domains, we include a noisy rollout policy. The rollout heads directly to s_g, but the computed action has an added noise of $\mathcal{N}(\mathbf{0}, \sigma_r^2 \mathbf{I})$. In our experiments we set $\sigma_r = 0.1$.

The implementation of dD-Continuous Light-Dark was written using POMDPs.jl [Egorov *et al.*, 2017]. In this domain, it is straightforward to compute the transition density:

$$p_T(s'|s, a) = \frac{1}{(2\pi\sigma_T^2)^{d/2}} \exp\left(-\frac{\|s' - (s + a)\|^2}{2\sigma_T^2}\right).$$

Table 16: Performance (Mean ± SEM) and Runtimes in Mountain Car POMDP.

Simulations	**DPW Variants**		**AG Variants (Ours)**		**POMCP Variants**	
n^{sims} / $n^{\text{sims}}_{\text{POMCPOW}}$	PFT-DPW	PFT-VPW	AG-PFT-DPW	AG-PFT-VPW	POMCPOW	VOMCPOW
50 / 274	$24.64^{\dagger} \pm 0.37$ 0.005s	$24.43^{\dagger} \pm 0.38$ 0.006s	$24.96^{\dagger} \pm 0.41$ 0.047s	$24.72^{\dagger} \pm 0.44$ 0.048s	$24.80^{\dagger} \pm 0.29$ 0.016s	$\mathbf{25.62} \pm 0.36$ 0.018s
89 / 487	$24.54^{\dagger} \pm 0.36$ 0.008s	$24.48^{\dagger} \pm 0.35$ 0.008s	$25.56^{\dagger} \pm 0.36$ 0.051s	$\mathbf{25.58} \pm 0.37$ 0.051s	$25.39^{\dagger} \pm 0.31$ 0.019s	$25.46^{\dagger} \pm 0.37$ 0.020s
158 / 865	24.17 ± 0.38 0.013s	23.82 ± 0.40 0.013s	$25.21^{\dagger} \pm 0.41$ 0.143s	$\mathbf{25.98} \pm 0.34$ 0.142s	$25.36^{\dagger} \pm 0.33$ 0.034s	$25.38^{\dagger} \pm 0.36$ 0.037s
281 / 1539	23.23 ± 0.45 0.023s	23.14 ± 0.44 0.024s	$\mathbf{26.64} \pm 0.26$ 0.376s	25.01 ± 0.45 0.366s	24.75 ± 0.40 0.063s	24.05 ± 0.45 0.069s
500 / 2739	23.20 ± 0.43 0.042s	22.69 ± 0.45 0.044s	$\mathbf{26.96} \pm 0.25$ 0.928s	$26.27^{\dagger} \pm 0.35$ 0.904s	24.95 ± 0.36 0.116s	24.11 ± 0.44 0.134s

Table 17: Performance (Mean ± SEM) and Runtimes in Hill Car MDP.

Simulations	**DPW Variants**		**AG Variants (Ours)**	
n^{sims}	DPW	VPW	AG-DPW	AG-VPW
50	$-92.87^{\dagger} \pm 1.05$ 0.339s	$-92.71^{\dagger} \pm 1.06$ 0.359s	$\mathbf{-91.90} \pm 1.12$ 0.036s	-98.05 ± 0.56 1.412s
89	-92.06 ± 1.11 0.006s	-90.93 ± 1.18 0.008s	$\mathbf{56.27} \pm 1.00$ 0.389s	25.66 ± 2.16 0.127s
158	-71.43 ± 1.96 0.023s	-69.31 ± 2.01 0.022s	-39.16 ± 2.49 0.123s	$\mathbf{39.03} \pm 1.82$ 0.221s
281	-66.67 ± 2.08 0.029s	-67.08 ± 2.07 0.026s	$\mathbf{56.68} \pm 0.97$ 0.313s	-6.36 ± 2.55 0.472s
500	-66.04 ± 2.09 0.026s	-59.66 ± 2.23 0.029s	$\mathbf{56.34} \pm 1.00$ 0.744s	44.56 ± 1.62 1.028s

The gradient function $\nabla_a r$ was equal to 0 in this domain, and $\nabla_a \log p_T$ was calculated using Enzyme.jl [Moses and Churavy, 2020]. Enzyme.jl provided efficient automatic differentiation, yielding log-gradient computation times close to the transition probability.

Evaluation

We evaluated the algorithms on the dD-Continuous Light-Dark domain for $d = 2, 3, 4$ dimensions. PFT-DPW and AGMCTS were given a budget of $n^{\text{sims}}_{\max} = 500$ simulations. The particle count during planning was set to be dependent on the dimension:

$$J_{d=2} = 64, \quad J_{d=3} = 128, \quad J_{d=4} = 256.$$

The particle filter used for inference between planning steps had a particle count J^{PF} given by:

$$J^{\text{PF}}_{d=2} = 256, \quad J^{\text{PF}}_{d=3} = 512, \quad J^{\text{PF}}_{d=4} = 1024.$$

POMCPOW was given a simulation budget according to the formula

$$n^{\text{sims}}_{\text{POMCPOW},d} = n^{\text{sims}}_{\max} \cdot \sqrt{J_d}.$$

Additional parameters that were manually set in all dimensions were:

$$T^{add}_{\rho} = 0.9, \quad T^{del}_{\rho} = 1\text{e}{-8}, \quad K_{rollout} = 10,$$
$$K_{\text{opt}} = 3, \quad T^{\max}_{d_a} = 0.05 \cdot \sigma_T, \quad K^{\nabla}_b = 4$$
$$\omega_{\text{VOO}} = 0.85, \quad \Sigma_{\text{VOO}} = 0.05 \cdot \boldsymbol{I}.$$

The hyperparameters found by CE optimization for each algorithm in each dimension are summarized in Tables 2, 3 and 4. In this domain, exponential step size decay was used. The importance weight updates were linearized according to Equation (14), along with linearized weight-updates with state-reward action-gradient estimates via Equation (160). Performance results and runtimes are reported in Tables 6, 7 and 8.

C.3 Two-Agent dD-Continuous Light-Dark POMDP.

Problem Description

This domain is a multi-agent extension of the dD-Continuous Light-Dark domain described previously, and we specifically evaluate in $d = 2$ dimensions due to computational constraints. The state space and observation space are concatenated vectors of the individual agents, such that $\mathcal{S} = \mathcal{O} = \mathbb{R}^{2d}$, and the action space is the product of balls

Table 18: Performance (Mean ± SEM) and Runtimes in Hill Car POMDP.

Simulations	**DPW Variants**		**AG Variants (Ours)**		**POMCP Variants**	
n^{sims} / $n^{\text{sims}}_{\text{POMCPOW}}$	PFT-DPW	PFT-VPW	AG-PFT-DPW	AG-PFT-VPW	POMCPOW	VOMCPOW
50 / 274	-83.22 ± 1.55 0.044s	-78.37 ± 1.73 0.054s	48.63 ± 1.34 0.438s	35.93 ± 1.84 0.497s	$\mathbf{53.83} \pm 0.98$ 0.167s	21.96 ± 2.18 0.216s
89 / 487	-75.82 ± 1.81 0.061s	-72.40 ± 1.92 0.062s	-87.58 ± 1.36 0.117s	-54.87 ± 2.29 0.120s	$\mathbf{54.69} \pm 0.94$ 0.062s	26.43 ± 2.11 0.054s
158 / 865	-71.17 ± 1.94 0.095s	-63.52 ± 2.14 0.100s	$\mathbf{55.10} \pm 0.95$ 0.331s	15.29 ± 2.30 0.263s	$54.09^{\dagger} \pm 1.00$ 0.115s	27.83 ± 2.08 0.096s
281 / 1539	-46.51 ± 2.39 0.192s	-60.41 ± 2.19 0.175s	$\mathbf{55.30} \pm 0.93$ 0.764s	41.64 ± 1.66 0.563s	$54.56^{\dagger} \pm 0.98$ 0.189s	31.17 ± 2.00 0.164s
500 / 2739	-42.19 ± 2.43 0.360s	-57.66 ± 2.25 0.347s	$\mathbf{56.37} \pm 0.86$ 1.636s	40.18 ± 1.72 1.109s	$55.18^{\dagger} \pm 0.94$ 0.348s	30.55 ± 2.02 0.297s

Table 19: Manual hyperparameters in the Lunar Lander domains.

Param.	**MDP**	**POMDP**
J	1	150
J^{PF}	-	2000
$K_{rollout}$	1	5
K_{opt}	3	3
$T_{d_a}^{\max}$	0.01	0.01
T_{ρ}^{add}	0.9	0.9
T_{ρ}^{del}	0.001	1e−8
K_b^{∇}	-	3

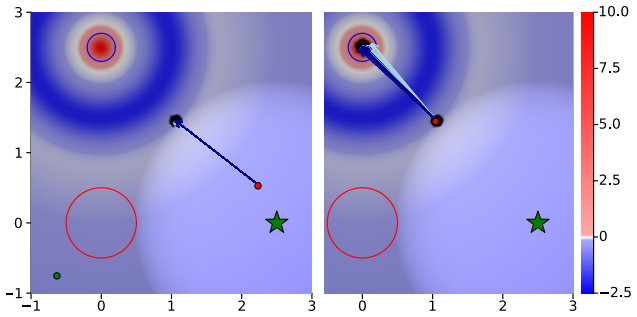
Table 20: Hyperparameters used in the evaluations of each algorithm in the Lunar Lander MDP domain.

	DPW Variants		**AG Variants (Ours)**	
Param.	DPW	VPW	AG-DPW	AG-VPW
c	60.50	58.96	61.54	61.10
k_a	1.43	1.50	1.87	1.79
α_a	0.59	0.58	0.51	0.54
k_o	0.07	0.08	0.07	0.11
α_o	0.29	0.69	0.91	0.74
η_{Adam}	-	-	5.5e−2	2.1e−3

$\mathcal{A} = B_{r_a}(\mathbf{0}_N) \times B_{r_a}(\mathbf{0}_N)$. The transition and reward models follow the same logic as the single-agent case, averaged over the number of agents. The transition dynamics are independent for each agent $i \in \{1, 2\}$, following $x^{(i)\prime} \sim \mathcal{N}(x^{(i)} + a^{(i)}, \sigma_T^2 \mathbf{I})$. The collective reward is the average of the individual rewards obtained by each agent, with the exception that the reward goal is only obtained for all agents if they are all within the goal threshold distance $T_{\text{goal}} = 0.2$.

The primary challenge in this domain lies in the coupled observation model. While the first agent observes the fixed beacon x_b, the second agent does not observe the beacon directly; instead, it observes its position relative to the first agent. Let $x = [x^{(1)}, x^{(2)}]^T$. The observation vector $z = [z^{(1)}, z^{(2)}]^T$ is distributed as:

$$z^{(1)} \sim \mathcal{N}(x^{(1)} - x_b, \Sigma(x^{(1)}, x_b))$$



(a) Middle of scenario, $t = 5$. **(b)** End of scenario, $t = 6$.

Figure 3: AGMCTS at 2D-Continuous Light-Dark. The agent's current state is the red dot, the current belief particles in orange, the next state in black, and the next belief particles in gray, the next observation is the green dot. The goal is the blue ring centered at $(0, 2.5)$, b_0 is the red ring centered at $(0, 0)$, and the beacon is the green star at $(2.5, 0)$. The reward function for the posterior state is the blue-red heatmap drawn in the background. The shaded area is where the observation noise is $\sigma_O > 5$. The chosen action branch of AGMCTS is drawn as the blue arrows, getting darker in hue with each action update. We can see that the optimized actions point closer to the goal from the current belief's mean.

$$z^{(2)} \sim \mathcal{N}(x^{(2)} - x^{(1)}, \Sigma(x^{(2)}, x^{(1)}))$$

where the covariance $\Sigma(u, v) = (\sigma_Z(\|u - v\|))^2 \mathbf{I}$ uses the same distance-dependent noise function σ_Z as the single-agent domain. This structure necessitates collaboration: Agent 1 must localize itself relative to the beacon to become a reliable reference point for Agent 2, effectively acting as a mobile beacon.

Evaluation

We evaluated the algorithms for $d = 2$ dimensions. All manually-chosen hyperparameters and particle counts followed the same as the single-agent experiments. The hyperparameters found via CE optimization for the collaborative setting are listed in Table 5. Notably, AGMCTS required a significantly higher learning rate in this domain compared to the single-agent version. Further hyperparameters are re-

Table 21: Hyperparameters used in the evaluations of each algorithm in the Lunar Lander POMDP domain.

	DPW Variants		AG Variants (Ours)		POMCP Variants	
Param.	PFT-DPW	PFT-VPW	AG-PFT-DPW	AG-PFT-VPW	POMCPOW	VOMCPOW
c	60.87	57.80	50.99	50.19	71.02	74.80
k_a	3.67	2.81	2.41	3.47	2.69	3.05
α_a	0.39	0.45	0.49	0.41	0.46	0.43
k_o	0.24	0.22	0.28	0.28	0.57	0.54
α_o	0.65	0.59	0.49	0.80	0.28	0.29
η_{Adam}	-	-	1.2e−7	3.7e−7	-	-

Table 22: Performance (Mean ± SEM) and Runtimes in Lunar Lander MDP.

Simulations	**DPW Variants**		**AG Variants (Ours)**	
n^{sims}	DPW	VPW	AG-DPW	AG-VPW
100	$44.40^{\dagger} \pm 1.03$ 0.019s	39.23 ± 1.97 0.083s	$44.22^{\dagger} \pm 1.41$ 0.022s	$\mathbf{44.63} \pm 0.55$ 0.086s
178	$50.24^{\dagger} \pm 0.99$ 0.002s	$47.46^{\dagger} \pm 1.35$ 0.002s	$\mathbf{50.69} \pm 0.48$ 0.004s	$48.79^{\dagger} \pm 1.00$ 0.203s
316	$\mathbf{55.80} \pm 0.41$ 0.008s	$54.46^{\dagger} \pm 0.41$ 0.007s	$54.00^{\dagger} \pm 1.05$ 0.239s	$54.49^{\dagger} \pm 0.42$ 0.017s
562	$\mathbf{59.80} \pm 0.37$ 0.005s	$58.45^{\dagger} \pm 0.35$ 0.005s	$59.31^{\dagger} \pm 0.39$ 0.046s	57.56 ± 0.45 0.059s
1000	$\mathbf{63.20} \pm 0.29$ 0.008s	60.94 ± 0.34 0.009s	61.28 ± 0.40 0.203s	60.28 ± 0.42 0.216s

ported in Table 5. Performance results and runtimes are reported in Table 9.

C.4 Mountain Car and Hill Car

Problem Description

A car is placed randomly in a valley between two hills, and the goal is to reach the top of the right hill. The car's action is its acceleration left or right, and must gain momentum in order to reach the goal. In this domain, the state $s = (x, v) \in \mathbb{R}^2$ is the car's position and velocity, $\mathcal{A} = [-a_{\max}, a_{\max}] \subset \mathbb{R}$ is the action space.

In the POMDP version, $z \in \mathbb{R}$ is the observation of the car's position with noise. It is given by $z \sim \mathcal{N}(x, \sigma_O^2)$, where $\sigma_O = 0.03$.

The car's position is bounded to $[x_{\min}, x_{\max}]$, and the velocity is bounded to $[-v_{\max}, v_{\max}]$.

If the car reaches the goal $x \geq x_{\max}$, it receives a reward of 100. If $x < x_{\min}$, or $|v| \geq v_{\max}$, a penalty reward of -100 is attained and the scenario terminates. Otherwise, the car obtains a reward of -0.1 at each time step that does not terminate. The reward function is a function of the posterior state, i.e. $r(s, a, s') = r(s')$. In Mountain Car, the scenario terminates after $L = 200$ time steps, while in Hill Car it terminates after $L = 30$ time steps. The discount in both domains is $\gamma = 0.99$.

The rollout policy chooses $a_{\max}$ when $v > 0$, and $-a_{\max}$ otherwise.

In both domains, the generative transition model is given by a noise applied to the action at the input to the deterministic dynamics:

$$\begin{aligned} \xi &\sim \mathcal{N}(0, \sigma_\xi^2), \\ \tilde{a} &= \text{clip}(a + \xi, -a_{\max}, a_{\max}), \\ s' &= f(s, \tilde{a}, s'), \end{aligned}$$

where $\sigma_\xi = 0.1$ for both domains.

When calculating the transition density in this domain, we do it based on the input noise to a deterministic simulator case via the Area Formula. For each (s, a, s') tuple we calculate the generating action noise ξ, depending on the scenario. If the obtained ξ is outside of the bounds (beyond numerical errors) $[L, R]$ where $L = -a_{\max} - a$ and $R = a_{\max} - a$, then the density is 0. Otherwise, because of the clip operation, we calculate its density by the following mixture:

$$p(\xi|s, a) = \begin{cases} CDF(\mathcal{N}(0, \sigma_\xi^2), L), & \text{if } \xi \leq L \\ 1 - CDF(\mathcal{N}(0, \sigma_\xi^2), R), & \text{if } \xi \geq R \\ PDF(\mathcal{N}(0, \sigma_\xi^2), \xi), & \text{otherwise.} \end{cases}$$

Afterwards, we calculate the transition model density by the Area Formula according to the case of *Input Noise to Simulator*.

Mountain Car Transition Model

In this domain, we set $x_{\min} = -1.5$, $x_{\max} = 0.5$, $v_{\max} = 0.05$, $a_{\max} = 1.0$. The deterministic dynamics are calculated by:

$$\begin{aligned} v' &= v + 0.001 \cdot \tilde{a} - 0.0025 \cos(3x), \\ x' &= x + v'. \end{aligned}$$

Table 23: Performance (Mean ± SEM) and Runtimes in Lunar Lander POMDP.

Simulations	**DPW Variants**		**AG Variants (Ours)**		**POMCP Variants**	
n^{sims} / $n^{\text{sims}}_{\text{POMCPOW}}$	PFT-DPW	PFT-VPW	AG-PFT-DPW	AG-PFT-VPW	POMCPOW	VOMCPOW
100 / 875	21.99 ± 4.67 0.015s	12.59 ± 5.40 0.020s	20.93 ± 4.90 0.307s	22.11 ± 4.88 0.343s	$\mathbf{42.13} \pm 3.84$ 0.088s	$41.96^{\dagger} \pm 3.81$ 0.145s
178 / 1557	25.03 ± 4.70 0.018s	$30.36^{\dagger} \pm 4.23$ 0.019s	$39.39^{\dagger} \pm 3.48$ 0.043s	$37.07^{\dagger} \pm 3.77$ 0.041s	$\mathbf{45.50} \pm 3.82$ 0.025s	$44.35^{\dagger} \pm 3.86$ 0.028s
316 / 2764	36.11 ± 3.78 0.031s	$41.24^{\dagger} \pm 3.31$ 0.031s	$45.49^{\dagger} \pm 3.26$ 0.081s	$44.51^{\dagger} \pm 3.13$ 0.078s	$\mathbf{51.65} \pm 3.09$ 0.037s	$49.52^{\dagger} \pm 3.40$ 0.037s
562 / 4916	$46.23^{\dagger} \pm 2.75$ 0.053s	41.27 ± 3.59 0.053s	$51.69^{\dagger} \pm 2.63$ 0.184s	$47.34^{\dagger} \pm 3.16$ 0.186s	$52.67^{\dagger} \pm 3.15$ 0.075s	$\mathbf{56.09} \pm 2.68$ 0.081s
1000 / 8748	$44.72^{\dagger} \pm 3.33$ 0.103s	$44.05^{\dagger} \pm 3.45$ 0.102s	$51.47^{\dagger} \pm 2.97$ 0.504s	$52.00^{\dagger} \pm 2.77$ 0.508s	$\mathbf{54.89} \pm 2.96$ 0.136s	$53.67^{\dagger} \pm 3.05$ 0.144s

In order to calculate the transition density, for given s, a, s' we invert the dynamics to obtain ξ.

Hill Car Transition Model

In this domain, we set $x_{\min} = -1.0$, $x_{\max} = 1.0$, $v_{\max} = 2.5$, $a_{\max} = 4.0$. The deterministic dynamics are calculated by integrating the car's continuous-time dynamics with an ODE solver over a time period of $\Delta t = 0.1$. The equations were integrated using the adaptive Tsitouras 5/4 Runge-Kutta method (Tsit5) [Tsitouras, 2011], with a manually specified initial step size of $\delta t = 0.01$.

The continuous-time dynamics are given by:

$$\dot{v} = \left(\frac{a}{m} - g \cdot h'(x) - v^2 \cdot h'(x) \cdot h''(x)\right) \frac{1}{1 + h'(x)^2},$$
$$\dot{x} = v,$$

where $m = 1$, $g = 9.81$, and the hill-curve function $h(x)$ is given by:

$$h(x) = \begin{cases} x^2 + x, & \text{if } x < 0 \\ \frac{x}{\sqrt{1.0 + 5.0 * x^2}}, & \text{otherwise.} \end{cases}$$

Because the input noise is only based on the action, we calculate ξ by caching simulator outputs (s, a, ξ, s') tuples for which $s' = f(s, \tilde{a})$. For a new tuple (s, a, s'), we calculate $\xi' = \tilde{a} - a$ for the cached $\tilde{a}$. The jacobian of the dynamics is calculated by ForwardDiff.jl and SciMLSensitivity.jl overloaded gradient function for the ODE solver.

Evaluation

DPW, PFT-DPW and AGMCTS were given a simulation budget of $n^{\text{sims}}_{\max} = 500$ simulations. POMCPOW simulation budget was set by the formula:

$$n^{\text{sims}}_{pomcpow} = n^{\text{sims}}_{\max} \cdot 0.08 \cdot J.$$

For the VPW algorithms, we have used the following VOO hyperparameters:

$$\omega_{\text{VOO}} = 0.85, \quad \Sigma_{\text{VOO}} = 0.05.$$

In these domains, we have not used exponential step size decay. In all of the Mountain/Hill car domains, the importance weight updates were linearized according to Equation (14). We used linearized weight-updates with state-reward action-gradient estimates via Equation (159) for the MDPs and Equation (160) for the POMDPs.

Further hyperparameters are reported in Tables 10, 11, 12, 13, and 14. Performance results and runtimes are reported in Tables 15, 16, 17, and 18.

C.5 Lunar Lander

Problem Description

We used the Julia implementation provided by Mern *et al.* [2021].

In this domain, a spacecraft vehicle must land safely on a flat surface from a high altitude. The vehicle state is represented by a six dimensional tuple $(x, y, \theta, \dot{x}, \dot{y}, \omega)$, where x and y are the horizontal and vertical positions, θ is the orientation angle, $\dot{x}$ and $\dot{y}$ are the horizontal and vertical speeds, and ω is the angular rate. The action space $\mathcal{A} \subset \mathbb{R}^3$ is a three-dimensional continuous space defined by the tuple $(F_x, T, \delta) \in [0, 15] \times [-5, 5] \times [-1, 1]$. T is the main thrust which acts along the vehicle's vertical axis through its center of mass. F_x is the corrective thrust, which acts along a horizontal axis offset from the center of mass by a distance δ.

The transition model is computed as an additive Gaussian noise to the output of a deterministic transition function. The deterministic transition function is given by the following set of computations:

$$\begin{aligned}
f_x &= \cos(\theta) \cdot F_x - \sin(\theta) \cdot T, \\
f_z &= \cos(\theta) \cdot T + \sin(\theta) \cdot F_x, \\
\tau &= -\delta \cdot F_x, \\
a_x &= \frac{f_x}{m}, \quad a_z = \frac{f_z}{m}, \quad \dot{\omega} = \frac{\tau}{I}, \\
v'_x &= v_x + a_x \cdot \Delta t + \epsilon_1 \cdot Q_4, \\
v'_z &= v_z + (a_z - 9.0) \cdot \Delta t + \epsilon_2 \cdot Q_5, \\
\omega' &= \omega + \dot{\omega} \cdot \Delta t + \epsilon_3 \cdot Q_6, \\
x' &= x + v_x \cdot \Delta t, \quad z' = z + v_z \cdot \Delta t, \quad \theta' = \theta + \omega \cdot \Delta t, \\
s' &= \begin{bmatrix} x' & z' & \theta' & v'_x & v'_z & \omega' \end{bmatrix}^T,
\end{aligned}$$

where the noise variables ϵ_i are sampled from a standard normal distribution, $Q_4 = 0.1$, $Q_5 = 0.1$, $Q_6 = 0.01$. We took

the values $m = 1$, $I = 10$, and $\delta t = 0.4$. For computing the transition density, we used the Area Formula by solving the dynamics for the noise variables ϵ_i for given (s, a, s') tuples.

In the POMDP variant, the vehicle makes noisy observations of its angular rate, horizontal speed, and above-ground height as measured by a distance sensor: $o = (\tilde{\omega}, \tilde{v_x}, \tilde{h})$, where $\tilde{\omega} \sim \mathcal{N}(\omega, 1.0^2)$, $\tilde{v_x} \sim \mathcal{N}(v_x, 0.01^2)$, and $\tilde{h} \sim \mathcal{N}(z/\cos(\theta), 0.1^2)$.

The reward function is defined as:

$$r(s, a, s') = \begin{cases} -1000, & \text{if } x \geq 15 \vee \theta \geq 0.5 \\ 100 - x - \dot{y}^2, & \text{if } y \leq 1 \\ -1, & \text{otherwise.} \end{cases}$$

The discount factor is set to $\gamma = 0.99$.

The initial vehicle state is sampled from a multivariate Gaussian with mean $\mu = (x = 0, y = 50, \theta = 0, \dot{x} = 0, \dot{y} = -10, \omega = 0)$. The rollout policy was computed as a simple proportional rule: $a_{rollout} = (-0.1 \cdot \dot{x}, -0.1 \cdot \dot{y}, 0)$.

Evaluation

DPW, PFT-DPW and AGMCTS were given a simulation budget of $n^{\text{sims}}_{\max} = 1000$ simulations. POMCPOW simulation budget was set by the formula:

$$n^{\text{sims}}_{pomcpow} = n^{\text{sims}}_{\max} \cdot 0.035 \cdot J.$$

For the VPW algorithms, we have used the following VOO hyperparameters:

$$\omega_{\text{VOO}} = 0.9, \quad \Sigma_{\text{VOO}} = \text{diag}(0.2, 0.5, 0.05).$$

In these domains, we have not used exponential step size decay. In all of the Lunar Lander domains, the importance weight updates were linearized according to Equation (14). We used linearized weight-updates with state-reward action-gradient estimates, as in Equation (159) for the MDPs and Equation (160) for the POMDPs.

Further hyperparameters are reported in Tables 19, 20, and 21. Performance results and runtimes are reported in Tables 22 and 23.