Appendix

The benefit of abstractions to the decision policy

One component of MCTS is its decision policy which decides which action to take given the previously obtained search tree statistics. A common decision policy and the one we employ here is the greedy strategy where one picks the root node action with the highest Q-value (sum of all returns divided by visits).

The Q-value of each (root node) action-visits pair can be viewed as a real-valued random variable. Furthermore, these random variables are independent iff their corresponding actions are different. Let us assume that there are $n \in \mathbb{N}$ root actions in total. We denote the respective Q-value random variables by $\mathcal{Q}_1, \ldots, \mathcal{Q}_n$. For simplicity, let us assume that the optimal action is the same as $\arg\max_{1 \leq a \leq n} \mathbb{E}[\mathcal{Q}_a]$.

Furthermore, let us assume that $\mathbb{E}[\mathcal{Q}_1] = \ldots = \mathbb{E}[\mathcal{Q}_k], k < n$. Though consequently the actions $a_1, \ldots, a_k$ are value-equivalent, they suffer from an overestimation bias in the decision policy that worsens exponentially with increasing k. The decision policy is invariant under replacing $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$ by the random variable $\mathcal{Q}^{\max} := \max(\mathcal{Q}_1, \ldots, \mathcal{Q}_k)$. Trivially, $\mathbb{E}[\mathcal{Q}^{\max}] \geq \mathbb{E}[\mathcal{Q}_1]$ and more concretely, it holds that for any constant $c \in \mathbb{R}$

$$\mathbb{P}(\mathcal{Q}^{\max} \geq c) = 1 - \mathbb{P}(\mathcal{Q}^{\max} < c) = 1 - \prod_{i=1}^{k} \mathbb{P}(\mathcal{Q}_i < c). \quad (4)$$

If we managed to detect that $\mathbb{E}[\mathcal{Q}_1] = \ldots = \mathbb{E}[\mathcal{Q}_k]$ and abstract them into a single random variable $\bar{\mathcal{Q}} := \frac{\mathcal{Q}_1 + \ldots + \mathcal{Q}_k}{k}$, then not only can the previously mentioned overestimation bias be fully mitigated but we can even decrease the variance since

$$\text{Var}(\bar{\mathcal{Q}}) = \frac{1}{k^2} \sum_{i=1}^{k} \text{Var}(\mathcal{Q}_i). \quad (5)$$

AUPO soundness

First, without any further assumptions, it can be shown that the root state abstraction AUPO is sound in the iteration limit, i.e., it only groups state-action pairs with the same Q^* value.

Theorem: Assume that MCTS has been run on a state s for m iterations and let $a^{\text{left}}, a^{\text{right}} \in \mathbb{A}(s)$ be two legal actions at s with $Q^*(s, a^{\text{left}}) \neq Q^*(s, a^{\text{right}})$. It then holds that

$$\lim_{m \to \infty} \mathbb{P}[((s, a^{\text{left}}), (s, a^{\text{right}})) \text{ is abstracted by AUPO with RF} = 1] = 0. \quad (6)$$

Proof: In the iteration limit, the Q values of (s, a^{right}) and (s, a^{left}) converge in probability to their corresponding Q^* values. Furthermore, since our MDP model definition encompasses only finite state-action pair sets, the reward function and consequently the set of possible episode returns are bounded and thus the empirical standard deviation is also bounded. Therefore, the lengths of the Gaussian Q value confidence intervals converge to 0. And since the confidence intervals are centered at their respective Q values, the probability of them overlapping converges to zero. Consequently, the probability of the state-action pairs being abstracted by AUPO also converges to zero. □

AUPO convergence speed

The key innovation that makes AUPO work in practice (this will be shown empirically later) is that one does not only compare a single pair of distributions to differentiate a single action pair but rather one compares a number of distributions induced by that action pair.

Using some simplifying assumptions, one can show that with an increase in D, the order of the number of samples required to differentiate two actions, with differing layerwise reward distributions, changes. The following theorem formalizes this.

Theorem: Again, assume that MCTS has been run on a state s for and let $a^{\text{left}}, a^{\text{right}} \in \mathbb{A}(s)$ be two legal actions at s that both have been played n times. The following assumptions are made:

1. It is assumed that all MCTS trajectories prior to performing the AUPO abstraction have been generated with a uniformly random tree policy. This ensures that for a fixed depth and root action, the obtained rewards are independent samples from a stationary distribution, a necessary requirement for the following result to hold.
2. All layerwise reward distributions $\mathcal{R}_{d,a^{\text{left}}}, \mathcal{R}_{d,a^{\text{right}}}, d \in \{1, \ldots, D\}$ are assumed to be independently distributed and Gaussians with means $m^{\text{left}} = (m_1^{\text{left}}, \ldots, m_D^{\text{left}})$ and $m^{\text{right}} = (m_1^{\text{right}}, \ldots, m_D^{\text{right}})$ and standard deviations $\sigma^{\text{left}} = (\sigma_1^{\text{left}}, \ldots, \sigma_D^{\text{left}})$, $\sigma^{\text{right}} = (\sigma_1^{\text{right}}, \ldots, \sigma_D^{\text{right}})$.
3. Lastly, it is assumed that AUPO has oracle access to these standard deviations and uses them instead of the empirical standard deviation when constructing the confidence intervals for the means.

Under these assumptions, if AUPO uses neither the return, nor the std-filter then

$$\forall \varepsilon > 0 : \mathbb{P}[\text{AUPO abstracts } a^{\text{left}} \text{ and } a^{\text{right}}] \in \mathcal{O}(f(n)),$$
$$f(n) = e^{-n \cdot (\varepsilon + \sum_{k=1}^{D} w_i)} \text{ where for } 1 \leq i \leq D : \quad (7)$$
$$w_i = \begin{cases} \frac{(\mu_i^{\text{left}} - \mu_i^{\text{right}})^2}{2(\sigma_i^{\text{left}} + \sigma_i^{\text{right}})^2}, & |\mu_i^{\text{left}} - \mu_i^{\text{right}}| \geq \frac{z^*}{\sqrt{n}}(\sigma_i^{\text{left}} + \sigma_i^{\text{right}}) \\ 1, & \text{otherwise} \end{cases},$$

and z^* is the critical value of the standard normal distribution for q (e.g. $z^* \approx 1.96$ for $q = 0.95$).

Proof: Firstly, we will derive a general upper bound for the probability of confidence intervals overlapping and then use this result in the context of AUPO's abstraction mechanism.

1) Let $n \in \mathbb{N}$ and $X_1, \ldots, X_n, Y_1, \ldots, Y_n$ be i.i.d. Gaussian random variables with respective means and stds of $\mu_X \geq \mu_Y$ and σ_X, σ_Y. For any confidence level $q \in [0, 1]$, the confidence interval for μ_X (analogously μ_Y) is of the form

$$[\overline{X} \pm \frac{z^* \cdot \sigma_X}{\sqrt{n}}] \quad (8)$$

where $z^* \in \mathbb{R}$ is the z-score for the given confidence level q and $\overline{X} = \frac{1}{n} \sum_{k=1}^{n} X_i$ ($\overline{Y}$ is defined analogously). The proba-

bility that the confidence intervals for μ_X and μ_Y overlap is thus given by

$$\mathbb{P}[|\underbrace{\overline{X}-\overline{Y}}_{Z:=}| \leq \underbrace{\frac{z^*}{\sqrt{n}}(\sigma_X+\sigma_Y)}_{T:=}]. \tag{9}$$

Since Z is Gaussian and the mean of Z is $\mu_Z := \mu_X - \mu_Y$ and the std is $\sigma_Z := \frac{\sigma_X+\sigma_Y}{\sqrt{n}}$, and since $\mathbb{P}[|Z| \leq T] = \mathbb{P}[Z \leq T] - \mathbb{P}[Z \leq -T]$ one obtains

$$\mathbb{P}[|Z| \leq T] = \frac{1}{2}\left[\text{erf}\left(\frac{T+\mu_Z}{\sqrt{2}\sigma_Z}\right) + \text{erf}\left(\frac{T-\mu_Z}{\sqrt{2}\sigma_Z}\right)\right] \tag{10}$$

using the identity $\Phi(\frac{x-\mu}{\sigma}) = \frac{1}{2}(1+\text{erf}(\frac{x-\mu}{\sigma\sqrt{2}}))$ that holds for any Gaussian with mean μ and std σ where Φ is the CDF for the standard Gaussian distribution and erf is the Gauss error function. Next, using that erf is an odd function with range $(-1, 1)$, yields

$$\begin{aligned}\mathbb{P}[|Z| \leq T] &= \frac{1}{2}\left[-\text{erfc}\left(\frac{\mu_Z+T}{\sqrt{2}\sigma_Z}\right) + \text{erfc}\left(\frac{\mu_Z-T}{\sqrt{2}\sigma_Z}\right)\right] \\ &\leq \frac{1}{2}\text{erfc}\left(\frac{\mu_Z-T}{\sqrt{2}\sigma_Z}\right) \quad \text{with erfc} := 1-\text{erf}.\end{aligned} \tag{11}$$

Next, two cases are differentiated. If $\mu_Z - T < 0$, we simply bound $\mathbb{P}[|Z| \leq T]$ by 1. In the other case, $\mu_Z - T \geq 0$, one can use an upper bound derived by Giuseppe Abreu (Abreu 2012) to further estimate this expression in terms of the exponential function. Concretely this yields,

$$\begin{aligned}\frac{1}{2}\text{erfc}\left(\frac{\mu_Z-T}{\sqrt{2}\sigma_Z}\right) &\leq \frac{1}{50}e^{-x^2} + \frac{1}{2(x+1)}e^{-x^2/2} \leq e^{-x^2/2}, \\ x &= \frac{\mu_Z-T}{\sigma_Z}.\end{aligned} \tag{12}$$

which is a function of the form

$$\begin{aligned}\frac{1}{2}\text{erfc}\left(\frac{\mu_Z-T}{\sqrt{2}\sigma_Z}\right) &\leq \frac{1}{50}e^{-x^2} + \frac{1}{2(x+1)}e^{-x^2/2} \leq e^{-x^2/2}, \\ x &= \frac{\mu_Z-T}{\sigma_Z}.\end{aligned} \tag{13}$$

2) By definition, AUPO using no return or std filter with a distribution tracking depth D only abstracts a^{left} and a^{right} iff their mean confidence intervals up to depth D all overlap. Since all reward distributions are independent by assumption, the probability of all confidence intervals overlapping, is given as the product of the individual ones overlapping. Hence, we can use the previously obtained results about a single pair of confidence intervals to obtain the following for every $\varepsilon > 0$

$$\begin{aligned}\mathbb{P}[\text{AUPO abstracts } a^{\text{left}} \text{ and } a^{\text{right}}] &\leq e^{-\lambda_1+\sqrt{n}\,\lambda_2 - n\sum_{k=1}^{D} w_i} \\ &\in \mathcal{O}(f(n)), \lambda_1, \lambda_2 \in \mathbb{R}^+.\end{aligned} \tag{14}$$

where $f(n) = e^{-n\cdot(\varepsilon+\sum_{k=1}^{D} w_i)}$ and for $1 \leq i \leq D$:

$$w_i = \begin{cases}\frac{(\mu_i^{\text{left}}-\mu_i^{\text{right}})^2}{2(\sigma_i^{\text{left}}+\sigma_i^{\text{right}})^2}, & |\mu_i^{\text{left}} - \mu_i^{\text{right}}| \geq \frac{z^*}{\sqrt{n}}(\sigma_i^{\text{left}}+\sigma_i^{\text{right}}) \\ 1, & \text{otherwise}\end{cases} \tag{15}$$

.

This proves the original statement. □

Proof of abstraction probability theorem

In this section, Equation 7 from the main paper is proven which is the following.

Theorem: Again, assume that MCTS has been run on a state s for and let $a^{\text{left}}, a^{\text{right}} \in \mathbb{A}(s)$ be two legal actions at s that both have been played n times. The following assumptions are made:

1. It is assumed that all MCTS trajectories prior to performing the AUPO abstraction have been generated with a uniformly random tree policy. This ensures that for a fixed depth and root action, the obtained rewards are independent samples from a stationary distribution, a necessary requirement for the following result to hold.
2. All layerwise reward distributions $\mathcal{R}_{d,a^{\text{left}}}$, $\mathcal{R}_{d,a^{\text{right}}}, d \in \{1, \ldots, D\}$ are assumed to be independently distributed and Gaussians with means $m^{\text{left}} = (m_1^{\text{left}}, \ldots, m_D^{\text{left}})$ and $m^{\text{right}} = (m_1^{\text{right}}, \ldots, m_D^{\text{right}})$ and standard deviations $\sigma^{\text{left}} = (\sigma_1^{\text{left}}, \ldots, \sigma_D^{\text{left}})$, $\sigma^{\text{right}} = (\sigma_1^{\text{right}}, \ldots, \sigma_D^{\text{right}})$.
3. Lastly, it is assumed that AUPO has oracle access to these standard deviations and uses them instead of the empirical standard deviation when constructing the confidence intervals for the means.

Under these assumptions, if AUPO uses neither the return, nor the std-filter then

$$\begin{aligned}\forall \varepsilon > 0: \ &\mathbb{P}[\text{AUPO abstracts } a^{\text{left}} \text{ and } a^{\text{right}}] \in \mathcal{O}(f(n)), \\ &f(n) = e^{-n\cdot(\varepsilon+\sum_{k=1}^{D} w_i)}\end{aligned} \tag{16}$$

where for $1 \leq i \leq D$: $w_i = \begin{cases}\frac{(\mu_i^{\text{left}}-\mu_i^{\text{right}})^2}{2(\sigma_i^{\text{left}}+\sigma_i^{\text{right}})^2}, & |\mu_i^{\text{left}} - \mu_i^{\text{right}}| \geq \frac{z^*}{\sqrt{n}}(\sigma_i^{\text{left}}+\sigma_i^{\text{right}}) \\ 1, & \text{otherwise}\end{cases}$, and z^* is the critical value of the standard normal distribution for q (e.g. $z^* \approx 1.96$ for $q = 0.95$).

Ablation: Distribution tracking depth D

The performances in dependence of the distribution tracking depth D for each environment are visualized by Fig. 4.

Performances in dependence of the confidence level q

The performances in dependence of the confidence level q for each environment are visualized by Fig. 5.

Performances when using different filter combinations

The performances in dependence of different filter combinations for each environment are visualized by Fig. 6.

Performances on sparse reward environments

The AUPO performances on the sparse-reward environments Crossing Traffic, Navigation and Racetrack are visualized in Fig. 7.

Performances of Different Decision Policies

The MCTS performances with different decision policies are visualized in Fig. 8.

Reward distribution confidence intervals for SysAdmin

The average confidence intervals constructed by AUPO on the SysAdmin state described in Fig. 1a are provided by Tab. 2 and Tab. 1.

Problem models

In the following, we provide a brief description of each domain/environment that was used in this paper. Some of these environments can be parametrized (e.g., choosing a concrete map size for Sailing Wind). The concrete parameter settings can be found in the ExperimentConfigs folder in our publicly available GitHub repository (Schmöcker, Dockhorn, and Rosenhahn 2025b) as well as in Tab. 6. All environments considered here are described in depth by Schmöcker et al. (Schmöcker, Dockhorn, and Rosenhahn 2025a). For a detailed description of these environments, we refer to our implementation. In the following, descriptions for the environments that are not contained in the previous two papers are given.

Academic Advising: Though this problem is described by a survey paper (Schmöcker and Dockhorn 2025), we use a modified version in this paper. Originally, the agent would also always receive a negative reward as long as there is one mandatory course that has not been passed. We increased the reward density, by letting this negative reward be dependent on the number of missing mandatory courses. Furthermore, we also added a reward for every course passed.

Runtime measurements

We validate the claim that AUPO adds only a minor runtime overhead over vanilla MCTS for high iteration budgets, the following table, Tab. 3 lists the average decision-making times for each environment of AUPO compared to MCTS for 100 and 2000 iterations on states sampled from a distribution induced by random walks. This shows that while AUPO adds a significant overhead for low iteration budgets, the impact of the decision policy and therefore AUPO's runtime overhead vanishes. Note, though, that this runtime is both heavily implementation and hardware-dependent, and more efficient implementations might reduce this overhead. In particular, we are using highly optimized environment implementations that could be the runtime bottleneck in more complex environments.

Monte Carlo Tree Search

AUPO heavily relies on Monte Carlo Tree Search (MCTS) which we are going to describe now. Let M be a finite horizon MDP. On a high level, MCTS repeatedly samples trajectories starting at some state $s_0 \in S$ where a decision has to be made until a stopping criterion is met. The final decision is then chosen as the action at s_0 with the highest average return. In contrast to a pure Monte Carlo search, MCTS improves subsequent trajectories by building a tree from a subset of the states encountered in the last iterations which is then exploited. In contrast to pure Monte Carlo search, MCTS is guaranteed to converge to the optimal action.

An MCTS search tree is made of two components. Firstly, the state nodes, that represent states and Q nodes that represent state action pairs. Each state node, saves only its children which are a set of Q nodes. Q nodes save both its children which are state nodes and the number of and the sum of the returns of all trajectories that were sampled starting at the Q node.

Initially, the MCTS search tree consists only of a single state node representing s_0. Until some stopping criterion is met, the following steps are repeated.

1. **Selection phase**: Starting at the root node, MCTS first selects a Q node according to the so-called tree policy, which may use the nodes' statistics, and then samples one of the Q node's successor states. If either a terminal state node, a state node with at least one non-visited action (partially expanded), or a new Q node successor state is sampled, the selection phase ends.

   A commonly used tree policy (**and the one we used**) that is synonymously used with MCTS is Upper Confidence Trees (UCT) (Kocsis and Szepesvári 2006) which selects an action that maximizes the Upper Confidence Bound (UCB) value. Let $s \in S$ and V_a, N_a with $a \in \mathbb{N}$ be the return sum and visits and of the Q nodes of the node representing s. The UCB value of any action a is then given by

$$\mathrm{UCB}(a) = \underbrace{\frac{V_a}{N_a}}_{\text{Q term}} + \underbrace{\lambda \sqrt{\frac{\log\left(\sum_{a' \in \mathbb{A}(s)} N_{a'}\right)}{N_a}}}_{\text{Exploration term}}. \tag{17}$$

   The exploration term quantifies how much the Q term could be improved if this Q node was fully exploited and is controlled by the exploration constant $\lambda \in \mathbb{R} \cup \{\infty\}$. If one chose $\lambda = 0$, the UCT selection policy becomes the greedy policy and for $\lambda = \infty$, the selection policy becomes a uniform policy over the visits. In case of equality, some tiebreak rule has to be selected, which is typically a random tiebreak. From here, will use MCTS and UCT (MCTS with UCB selection formula) synonymously.

2. **Expansion**: Unless the selection phases ended in a terminal state node, the search tree is expanded by a single node. In case the selection phase ended in a partially expanded state node, then one unexpanded action is selected (e.g. randomly, or according to some rule), the corresponding Q node is created and added as a child and one successor state of that Q node is sampled and added as a child to the new Q node. If the selection phase ended

Action	1000 iterations	2000 iterations	3000 iterations	4000 iterations
Hub (0)	(0.89, 1.08)	(0.91, 1.04)	(0.92, 1.03)	(0.92, 1.02)
1	(1.14, 1.38)	(1.15, 1.32)	(1.18, 1.31)	(1.21, 1.33)
2	(1.13, 1.37)	(1.16, 1.32)	(1.19, 1.32)	(1.19, 1.31)
3	(1.16, 1.40)	(1.16, 1.33)	(1.17, 1.31)	(1.21, 1.33)
4	(1.11, 1.34)	(1.15, 1.32)	(1.18, 1.32)	(1.19, 1.30)
5	(1.14, 1.38)	(1.17, 1.34)	(1.17, 1.31)	(1.19, 1.31)
6	(1.15, 1.39)	(1.17, 1.34)	(1.19, 1.32)	(1.18, 1.30)
7	(1.12, 1.35)	(1.16, 1.33)	(1.19, 1.33)	(1.19, 1.31)
8	(1.12, 1.36)	(1.16, 1.33)	(1.19, 1.33)	(1.20, 1.32)
9	(1.11, 1.35)	(1.17, 1.34)	(1.18, 1.32)	(1.19, 1.31)
Idle	(1.12, 1.36)	(1.19, 1.36)	(1.19, 1.33)	(1.21, 1.33)

Table 1: 3-step standard deviation 95% confidence intervals for the reward distribution after a different number of MCTS iterations on the SysAdmin state of Fig. 1a. Note that with higher iteration counts, rebooting the hub can be separated from the remaining actions

Action	200 iterations	500 iterations	1000 iterations	2000 iterations
Hub (0)	(7.72, 8.16)	(7.83, 8.10)	(7.88, 8.06)	(7.90, 8.03)
1	(7.67, 8.11)	(7.76, 8.03)	(7.81, 8.00)	(7.85, 7.98)
2	(7.71, 8.14)	(7.76, 8.04)	(7.81, 8.00)	(7.84, 7.97)
3	(8.62, 9.04)	(8.70, 8.97)	(8.73, 8.92)	(8.74, 8.88)
4	(7.68, 8.13)	(7.77, 8.04)	(7.82, 8.01)	(7.85, 7.98)
5	(7.72, 8.16)	(7.78, 8.05)	(7.83, 8.02)	(7.85, 7.99)
6	(7.67, 8.12)	(7.76, 8.04)	(7.81, 8.00)	(7.84, 7.97)
7	(7.69, 8.14)	(7.78, 8.05)	(7.83, 8.02)	(7.84, 7.98)
8	(7.68, 8.13)	(7.78, 8.05)	(7.81, 8.01)	(7.84, 7.98)
9	(7.68, 8.12)	(7.76, 8.04)	(7.82, 8.01)	(7.83, 7.97)
Idle	(7.64, 8.10)	(7.74, 8.02)	(7.77, 7.97)	(7.80, 7.94)

Table 2: 2-step mean 95% confidence intervals for the reward distribution after different numbers of MCTS iterations on the SysAdmin state of Fig. 1a. Note that even with very low iteration counts, rebooting machine 3 can easily be separated from the other actions.

because a new successor of a Q node was sampled, then a state node representing this new state is added as a child to that Q node.

3. **Rollout/Simulation phase**: Starting at the state $s_{rollout}$ of the newly added state node of the expansion phase (or at a terminal state node reached by the selection phase), actions according to the rollout policy are repeatedly selected and applied to $s_{rollout}$ until a terminal state is reached. All states encountered during this phase are not added to the search tree.

4. **Backpropagation**: In this phase, the statistics of all Q nodes that were part of the last sampled trajectory that corresponds to a path in the search tree are updated by incrementing their visit count and adding the trajectory's return (of the trajectory starting at the respective Q node) to their return sum statistic.

Once the MCTS search tree has been built (by reaching an iteration limit in our case) and statistics have been gathered, the final decision is made by the decision policy that in our MCTS version simply chooses the action with the highest final Q value.

Definition of relative improvement and pairings score

In the main experimental section, we evaluated AUPO with respect to the relative improvement and pairings score, which are formalized here. This pairings score was also used in (Schmöcker, Schnell, and Dockhorn 2025). While the pairings score is calculated by summing over the number of tasks where some agent performed better than another, the relative improvement score also takes the percentage of the improvement into account; however, it is prone to outliers. Hence, we considered both scores to paint the full picture.

Definition: Concretely, let $\{\pi_1, \dots, \pi_n\}$ be n agents (e.g., concrete parameter settings for possibly different base algorithms such as AUPO or MCTS) where each agent was evaluated on m tasks (in this paper, a task will always be a given MCTS iteration budget and an environment) where $p_{i,k} \in \mathbb{R}$ denotes the performance of agent π_i on the k-th task. The pairings score matrix $M^{\text{pairings}} \in \mathbb{R}^{n \times n}$ is defined as

$$M^{\text{pairings}}_{i,j} = \frac{1}{m-1} \sum_{1 \leq k \leq m} \text{sgn}(p_{i,k} - p_{j,k}) \qquad (18)$$

where sgn is the signum function. The pairings score

Domain	AUPO-100	MCTS-100	AUPO-2000	MCTS-2000
Academic Advising	1.15	1.59	23.71	25.36
Cooperative Recon	2.41	2.57	52.25	54.21
Earth Observation	7.03	6.95	130.57	136.73
Game of Life	3.93	4.31	65.72	65.18
Manufacturer	10.31	9.82	185.76	186.81
Multi-armed bandit	0.16	1.16	3.13	4.33
Push Your Luck	2.26	2.47	43.21	45.38
Sailing Wind	1.90	2.01	34.34	35.15
Saving	0.82	0.87	17.41	18.24
Skills Teaching	2.42	2.61	52.20	53.91
SysAdmin	1.20	1.58	22.97	24.29
Tamarisk	2.78	2.88	47.84	48.95
Traffic	3.05	3.85	61.23	63.74
Wildfire	1.48	2.20	34.22	35.73

Table 3: Average decision-making times of AUPO and MCTS in milliseconds for 100 and 2000 iterations. For AUPO the most computational heavy version has been used, which uses $p = 0.8, D = 4$, the return- and std filter. This data was obtained using an Intel(R) Core(TM) i5-9600K CPU @ 3.70GHz. The data shows a median runtime overhead of ≈8% for 100 iterations and ≈4% for 2000 iterations.

$s_i^{\text{pairings}}, i \leq n$ is given by

$$s_i^{\text{pairings}} = \frac{1}{n-1} \sum_{1 \leq l \leq n, l \neq i} M_{i,l}^{\text{pairings}}. \tag{19}$$

The relative improvement matrix $M^{\text{rel}} \in \mathbb{R}^{n \times n}$ is defined as

$$M_{i,j}^{\text{rel}} = \frac{1}{m-1} \sum_{1 \leq k \leq m} \frac{p_{i,k} - p_{j,k}}{\max(|p_{i,j}|, |p_{j,k}|)} \tag{20}$$

and the relative improvement score $s_i^{\text{rel}}, i \leq n$ is given by

$$s_i^{\text{rel}} = \frac{1}{n-1} \sum_{1 \leq l \leq n, l \neq i} M_{i,l}^{\text{rel}}. \tag{21}$$

Pairings and Relative Improvement Scores

The full list of pairings and relative improvement scores for each individual iteration budget is provided by Tab. 4 and Tab. 5.

Comparison with other distribution discrimination techniques

This section discusses alternatives to the Gaussian confidence intervals that were ultimately chosen for the experiments. Those alternatives are using a distribution distance measure, such as the Kolmogorov-Smirnov (KS) distance or an asymptotic confidence interval. This section uses the same symbols that are introduced in the Method section of the main paper.

- To include the former with AUPO, one would have to save each reward of each trajectory up to depth D for each root action. The KS distance D_{a_i,a_j}^{KS} for any root action pair a_i, a_j at depth $d \leq D$ is then calculated as

$$D_{d,a_i,a_j}^{\text{KS}} = \sup_{x \in \mathbb{R}} |U_{d,a_i}(x) - U_{d,a_j}(x)| \tag{22}$$

  where U_{d,a_i}, U_{d,a_j} are the cumulative distribution functions of $\mathcal{R}_{d,a_i}$ and $\mathcal{R}_{d,a_j}$. AUPO then abstracts a_i, a_j if and only iff $D_{d,a_i,a_j}^{\text{KS}} \leq \mu_{\text{KS}}$ for all $1 \leq d \leq D$ where $\mu_{\text{KS}} \in \mathbb{R}^+$ is a parameter.
- The chosen Gaussian confidence interval for the standard deviation of AUPO is not asymptotic, as the probability of the confidence interval containing the true standard deviation of $\mathcal{R}_{d,a_i}$ for action a_i at root state s does not necessarily approach the confidence level q. This is only the case for Gaussian distributions. One could, however, employ an asymptotic confidence interval that does possess this property even when the reward distributions are not Gaussian. The asymptotic confidence interval for the standard deviation is given as

$$\text{CI}_{d,a_i}^{\text{std,asymp}} = \left[\sqrt{s_{d,i}} \pm z^* \sqrt{\frac{(\kappa_{d,i} - 1) s_{d,i}}{4 m_i}} \right] \tag{23}$$

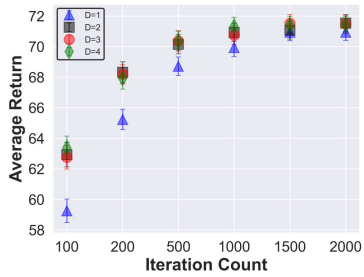
  where κ_{d,a_i} is the empirical kurtosis, which is defined as the empirical fourth moment divided by the square of the empirical second moment of $\mathcal{R}_{d,a_i}$, z^* is the critical value of the Gaussian distribution (e.g. $z^* \approx 1.96$ for $q = 0.95$), and $s_{d,i}$ is the empirical standard deviation of $\mathcal{R}_{d,a_i}$.

The small-scale experiments conducted on these used $C = 4$, $D \in \{1, 2, 3, 4\}$, RF = No. For both standard AUPO and the asymptotic confidence intervals version, SF $\in$ {No,Yes} and $q \in \{0.8, 0.9, 0.95, 0.99\}$ was used and for the KS-distance version, $\mu_{\text{KS}} \in \{0.1, 0.2, \ldots, 0.9\}$ was tested. Figure 9 shows the parameter-optimized performances given this experiment setup. In most environments, all three methods are relatively similar in performance. However, since there are two environments, namely Push Your Luck and Cooperative Recon in which standard AUPO decisively beats asymptotic confidence intervals, we discarded that method. Similarly, for the KS-distance-based AUPO version, standard AUPO also significantly outperforms it in Push Your Luck, Cooperative Recon, and Multi-

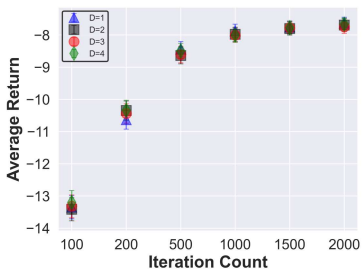
armed bandit. On the contrary, KS-based AUPO is only significantly better than standard AUPO in Academic Advising and Skills Teaching for low iteration budgets; hence, we also decided to discard this method.

Default experimental parameters

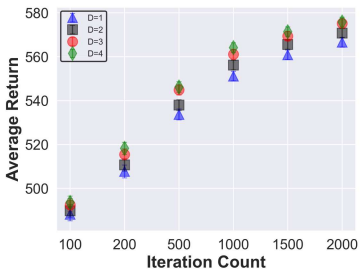
An overview of the default experimental parameters for this paper is provided in Tab. 6.



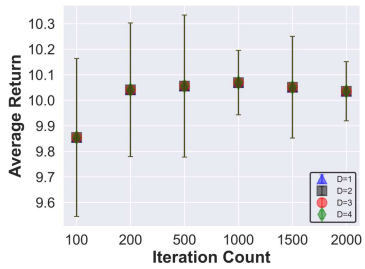
(a) Academic Advising



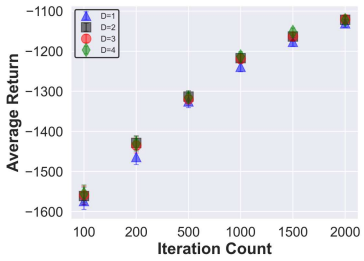
(b) Earth Observation



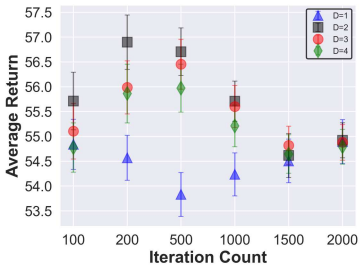
(c) Game of Life



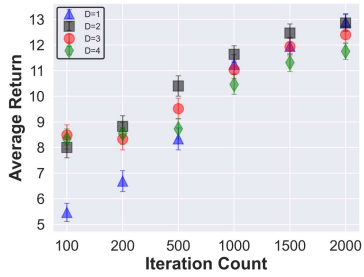
(d) Multi-armed bandit



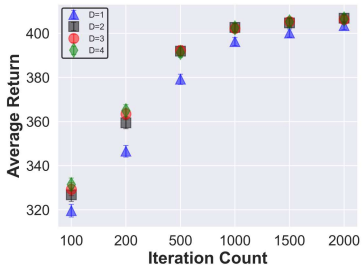
(e) Manufacturer



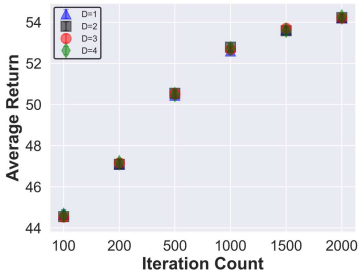
(f) Push Your Luck



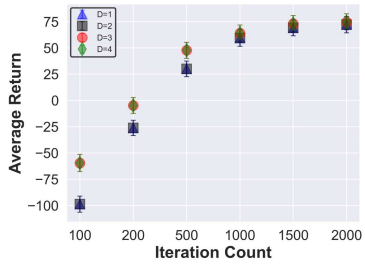
(g) Cooperative Recon



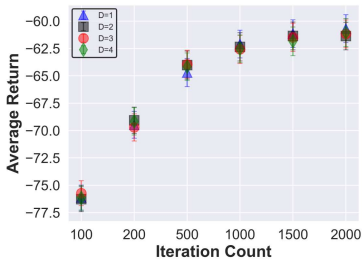
(h) SysAdmin



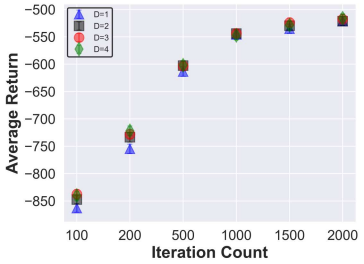
(i) Saving



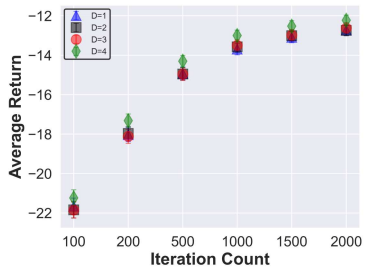
(j) Skill Teaching



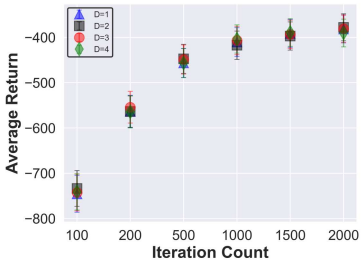
(k) Sailing Wind



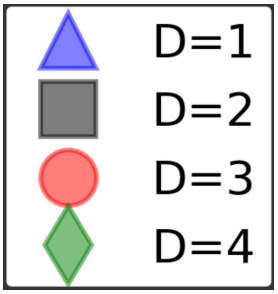
(l) Tamarisk



(m) Traffic

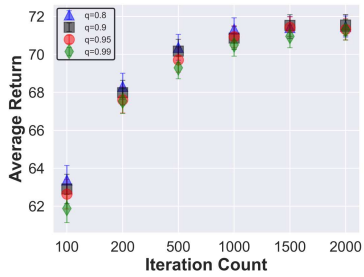


(m) Wildfire

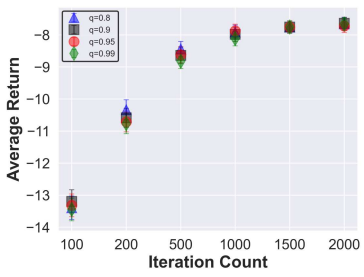


Legend

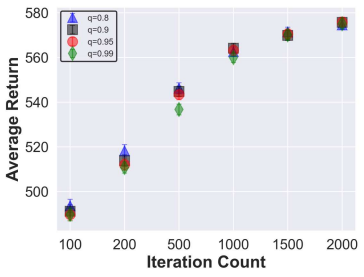
Figure 4: The performance graphs of in dependence of the MCTS iteration count of the parameter optimized versions of AUPO using different fixed values for the distribution tracking depth D.



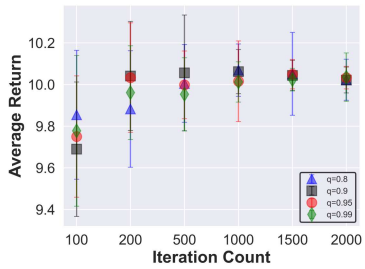
(a) Academic Advising



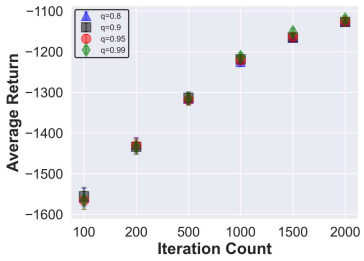
(b) Earth Observation



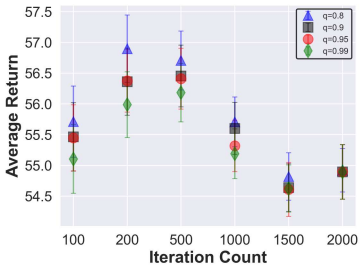
(c) Game of Life



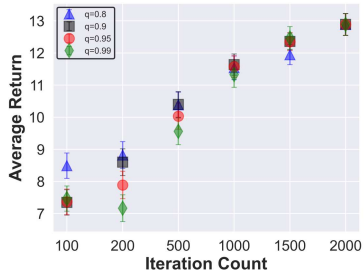
(d) Multi-armed bandit



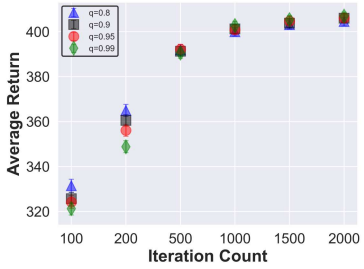
(e) Manufacturer



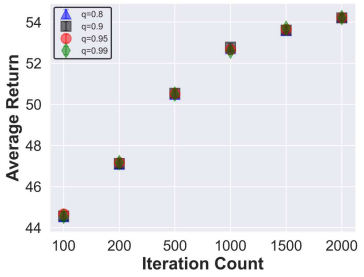
(f) Push Your Luck



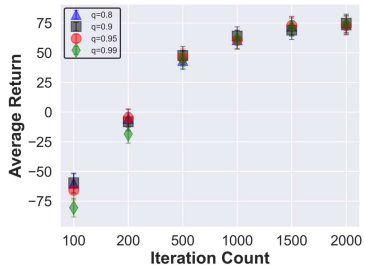
(g) Cooperative Recon



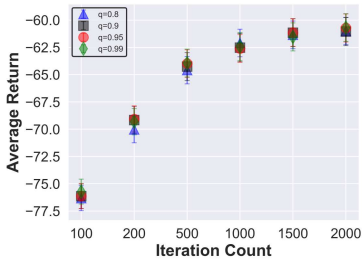
(h) SysAdmin



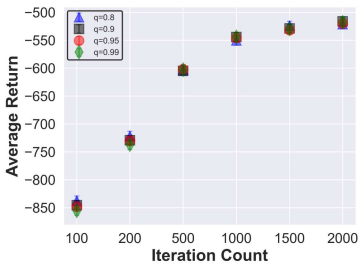
(i) Saving



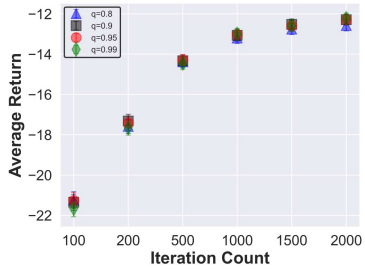
(j) Skill Teaching



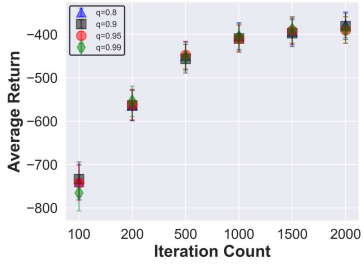
(k) Sailing Wind



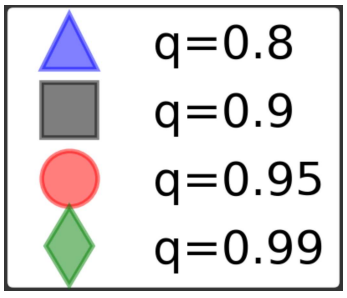
(l) Tamarisk



(m) Traffic

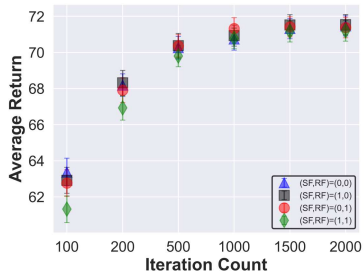


(m) Wildfire

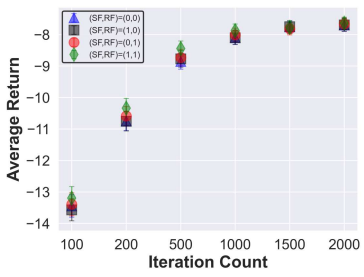


Legend

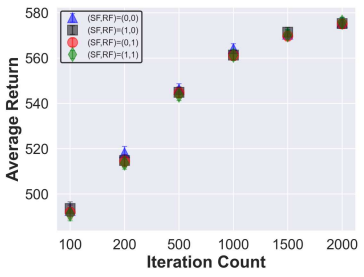
Figure 5: The performance graphs of in dependence of the MCTS iteration count of the parameter optimized versions of AUPO using different fixed values for the confidence q.



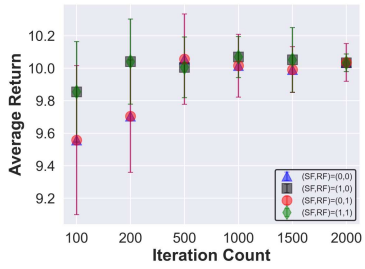
(a) Academic Advising



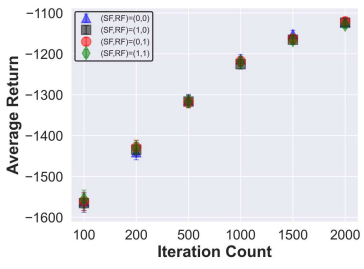
(b) Earth Observation



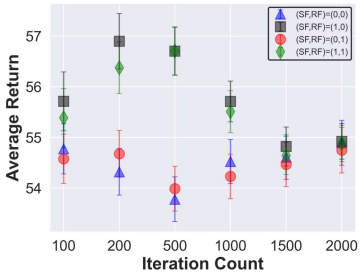
(c) Game of Life



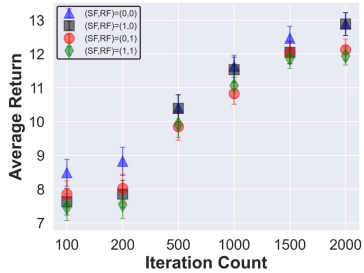
(d) Multi-armed bandit



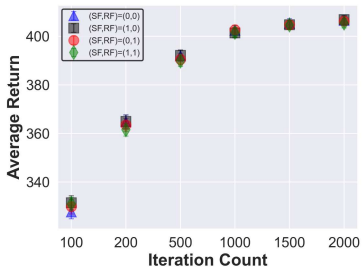
(e) Manufacturer



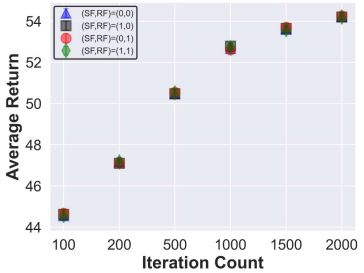
(f) Push Your Luck



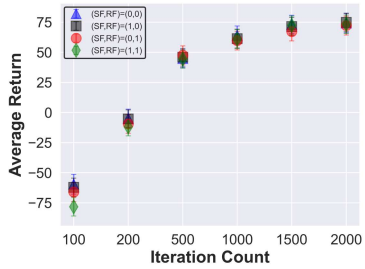
(g) Cooperative Recon



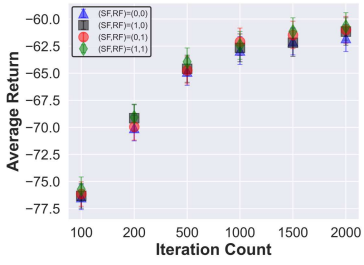
(h) SysAdmin



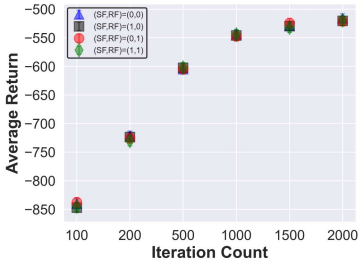
(i) Saving



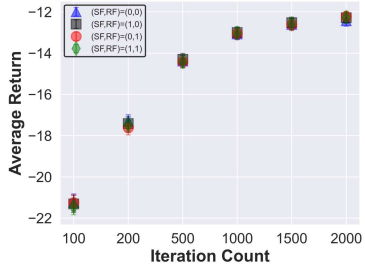
(j) Skill Teaching



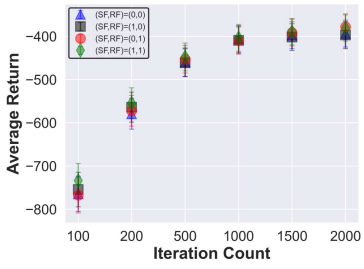
(k) Sailing Wind



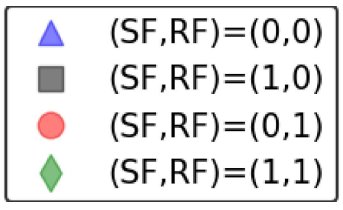
(l) Tamarisk



(m) Traffic

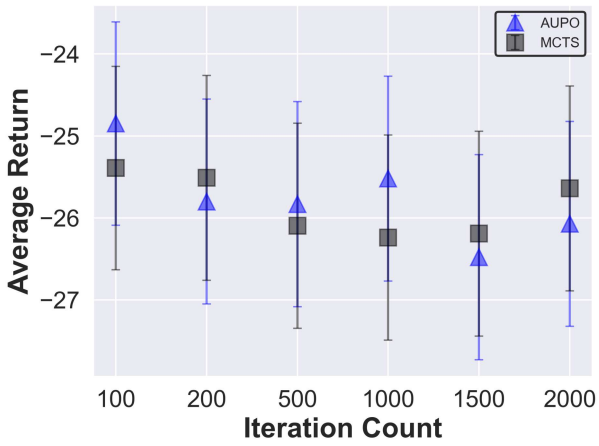


(m) Wildfire

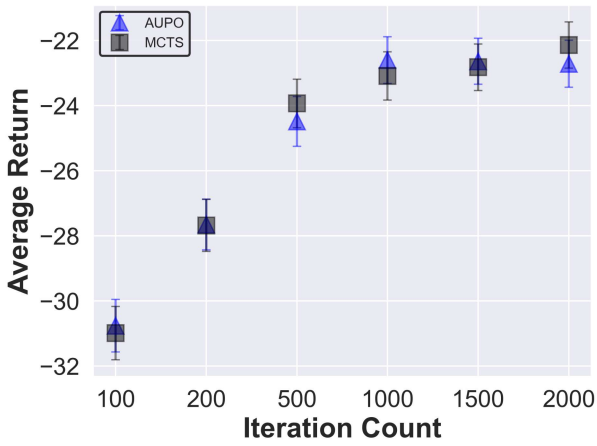


Legend

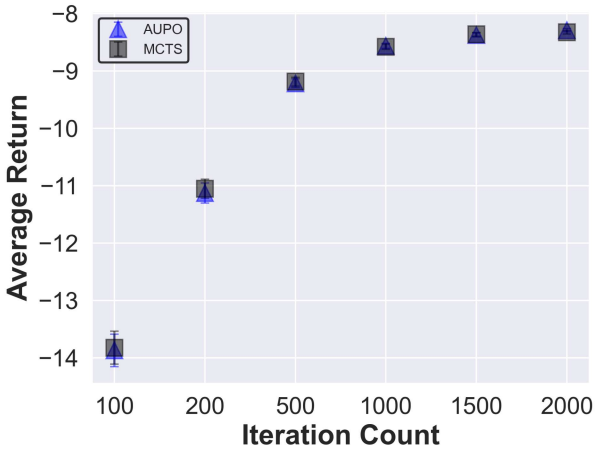
Figure 6: The performance graphs of in dependence on the MCTS iteration count of the parameter optimized versions of AUPO using different fixed filter settings. Both the return (RF) and std filter (SF) are varied.



(a) Crossing Traffic

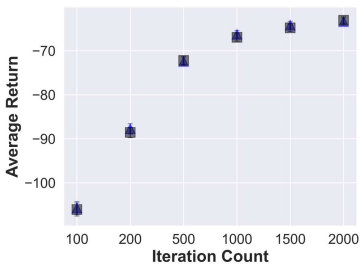


(b) Navigation

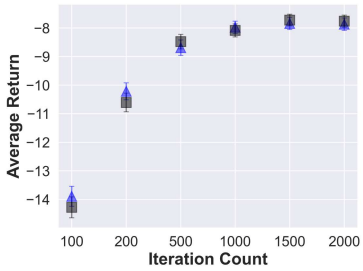


(c) Racetrack

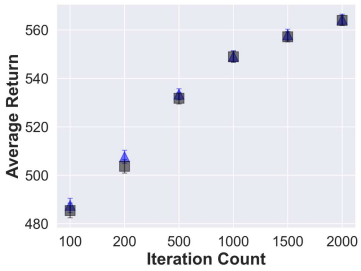
Figure 7: The performances (↑) of AUPO versus MCTS on sparse reward environments, showing that the performances do not significantly differ, as expected, since there is no information in the reward function for AUPO to separate actions. The AUPO parametrization used here is the one that achieved the highest pairings score in Figure 3, i.e $C = 2, q = 0.8, D = 4$, RF=No, SF=Yes. For MCTS, $C = 2$ was also used, as this value also achieved the highest pairings score.



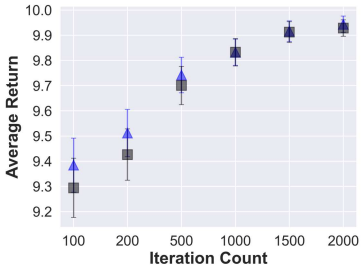
(a) Academic Advising



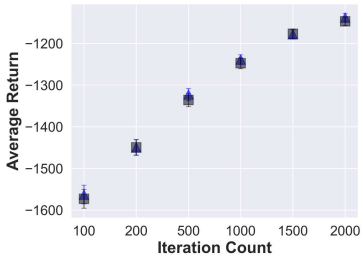
(b) Earth Observation



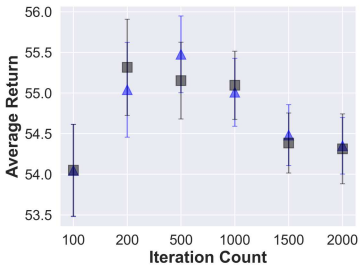
(c) Game of Life



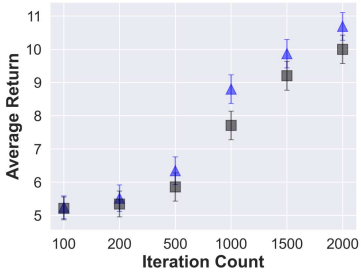
(d) Multi-armed bandit



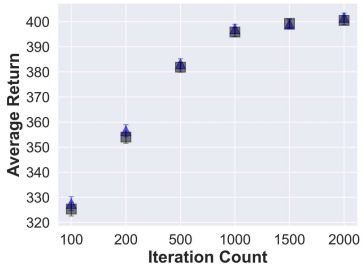
(e) Manufacturer



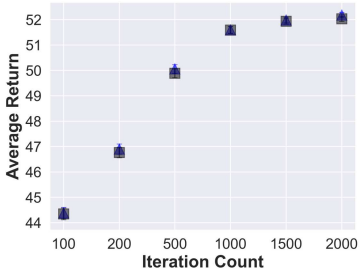
(f) Push Your Luck



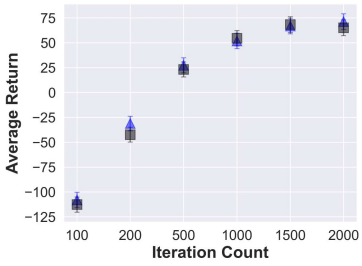
(g) Cooperative Recon



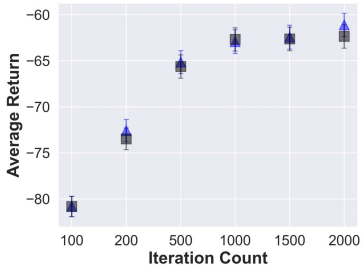
(h) SysAdmin



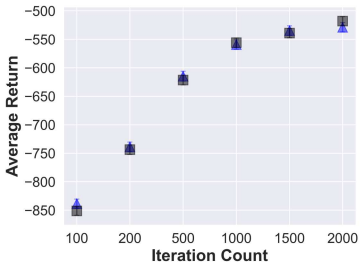
(i) Saving



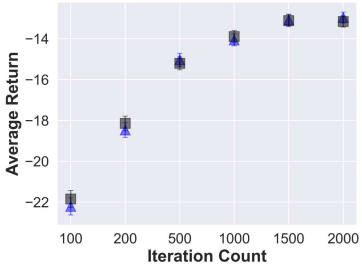
(j) Skill Teaching



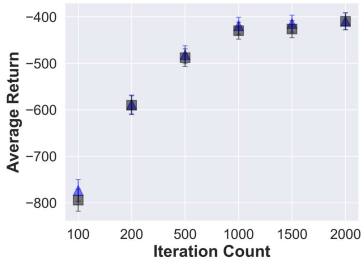
(k) Sailing Wind



(l) Tamarisk



(m) Traffic



(m) Wildfire

Legend

Figure 8: The exploration-constant optimized performance graphs of in dependence on the MCTS iteration for two MCTS decision policies; the decision policy that chooses the action with the highest Q value (By Q) and the policy that chooses the action with the most visits (By Visits).

Parameters	Score
AUPO(2,0.8,4,No,No)	0.120
U-AUPO(16,0.8,4,No,No)	0.068
RANDOM-ABS(2,0.9)	0.055
MCTS(2)	0.049
U-MCTS(4)	−0.006

100 iterations relative improvement score.

Parameters	Score
AUPO(2,0.8,3,No,Yes)	0.742
RANDOM-ABS(2,0.7)	0.360
U-AUPO(1,0.8,4,Yes,No)	0.319
MCTS(2)	0.301
U-MCTS(0.5)	−0.099

100 iterations pairings score.

Parameters	Score
AUPO(2,0.8,3,No,Yes)	0.137
RANDOM-ABS(1,0.7)	0.073
U-AUPO(0.5,0.95,4,Yes,No)	0.068
MCTS(2)	0.062
U-MCTS(1)	−0.005

200 iterations relative improvement score.

Parameters	Score
AUPO(2,0.8,3,Yes,Yes)	0.826
RANDOM-ABS(2,0.9)	0.438
MCTS(2)	0.368
U-AUPO(0.5,0.8,4,Yes,No)	0.330
U-MCTS(0.5)	−0.020

200 iterations pairings score.

Parameters	Score
AUPO(2,0.9,4,Yes,No)	0.139
RANDOM-ABS(2,0.4)	0.109
U-AUPO(0.5,0.9,3,Yes,No)	0.107
MCTS(2)	0.105
U-MCTS(0.5)	0.069

500 iterations relative improvement score.

Parameters	Score
AUPO(2,0.9,4,Yes,Yes)	0.793
U-AUPO(2,0.9,4,Yes,Yes)	0.459
MCTS(2)	0.431
RANDOM-ABS(1,0.7)	0.417
U-MCTS(0.5)	−0.007

500 iterations pairings score.

Table 4: The pairings and relative improvement scores for the **100, 200, and 500 iterations** setting for the parameters combination of AUPO, U-AUPO, RANDOM-ABS, MCTS, and U-MCTS with the highest respective scores as well as the concrete parameters used to reach that score. The parameters and environments used to obtain these scores are the same as the experiments as those described in the main experimental section. The parameter format for AUPO and U-AUPO is (C, q, D, RF, SF), the format RANDOM-ABS is (C, p_{random}), and for both MCTS and U-MCTS is (C). For RANDOM-ABS the best scores are obtained using the standard root policy.

Parameters	Score
AUPO(2,0.9,2,Yes,Yes)	0.108
RANDOM-ABS(2,0.9)	0.089
U-AUPO(1,0.9,4,Yes,Yes)	0.088
MCTS(2)	0.084
U-MCTS(1)	0.057

1000 iterations relative improvement score.

Parameters	Score
AUPO(2,0.9,4,Yes,Yes)	0.770
U-AUPO(1,0.9,4,Yes,Yes)	0.499
RANDOM-ABS(2,0.9)	0.447
MCTS(2)	0.417
U-MCTS(1)	0.036

1000 iterations pairings score.

Parameters	Score
AUPO(2,0.99,2,Yes,Yes)	0.099
MCTS(2)	0.084
RANDOM-ABS(2,0.8)	0.083
U-AUPO(1,0.8,4,Yes,Yes)	0.083
U-MCTS(1)	0.061

1500 iterations relative improvement score.

Parameters	Score
AUPO(2,0.9,4,Yes,Yes)	0.763
U-AUPO(2,0.9,4,Yes,Yes)	0.532
MCTS(2)	0.438
RANDOM-ABS(2,0.7)	0.418
U-MCTS(2)	0.026

1500 iterations pairings score.

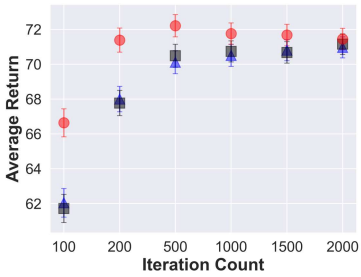
Parameters	Score
AUPO(2,0.95,4,Yes,Yes)	0.098
RANDOM-ABS(2,0.7)	0.087
MCTS(2)	0.086
U-AUPO(1,0.99,4,Yes,Yes)	0.086
U-MCTS(1)	0.059

2000 iterations relative improvement score.

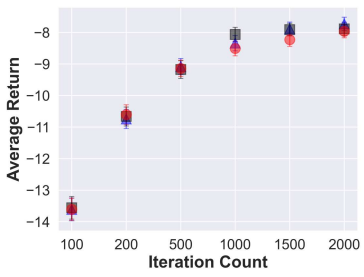
Parameters	Score
AUPO(2,0.95,4,Yes,Yes)	0.753
U-AUPO(2,0.95,4,Yes,Yes)	0.538
RANDOM-ABS(2,0.8)	0.532
MCTS(2)	0.487
U-MCTS(1)	0.028

2000 iterations pairings score.

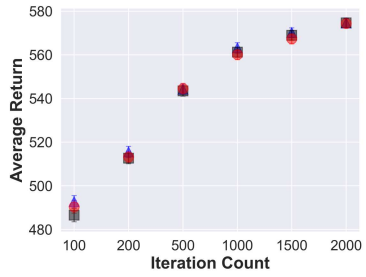
Table 5: The pairings and relative improvement score for the **1000, 1500, and 2000 iterations** setting for the parameters combination of AUPO, U-AUPO, RANDOM-ABS, MCTS, and U-MCTS with the highest respective score as well as the concrete parameters used to reach that score. The parameters and environments used to obtain these scores are the same as the experiments of the main experimental section. The parameter format for AUPO and U-AUPO is (C, q, D, RF, SF), the format RANDOM-ABS is (C, p_{random}), and for both MCTS and U-MCTS is (C). For RANDOM-ABS the best scores are obtained using the standard root policy.



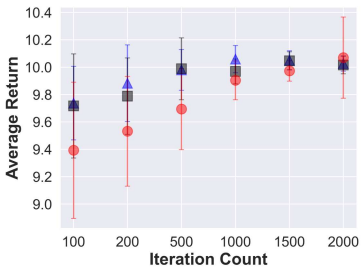
(a) Academic Advising



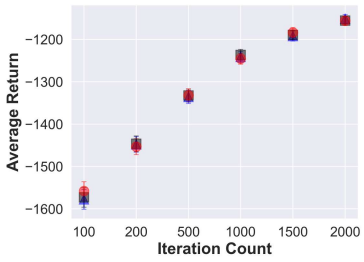
(b) Earth Observation



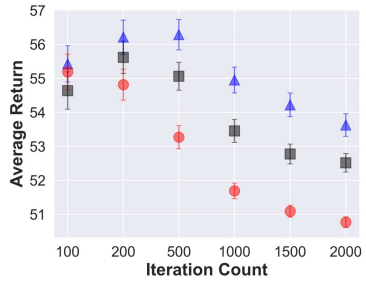
(c) Game of Life



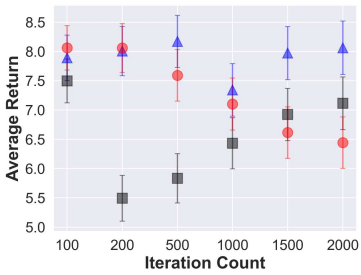
(d) Multi-armed bandit



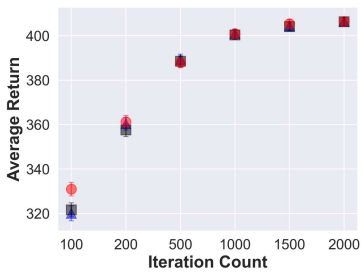
(e) Manufacturer



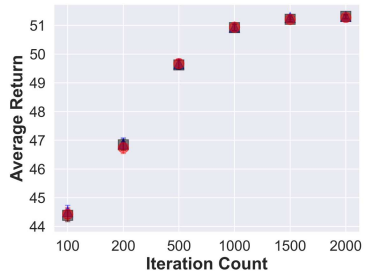
(f) Push Your Luck



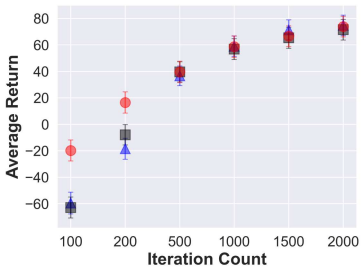
(g) Cooperative Recon



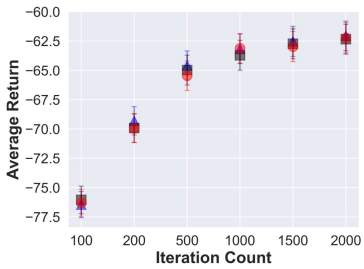
(h) SysAdmin



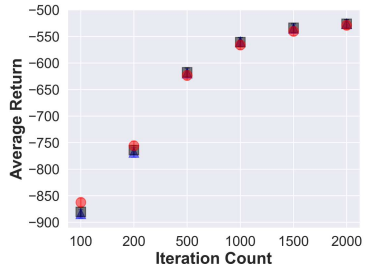
(i) Saving



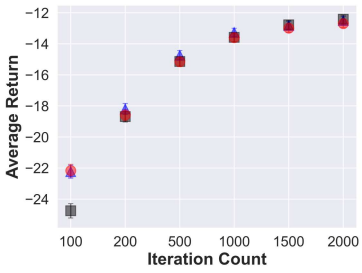
(j) Skill Teaching



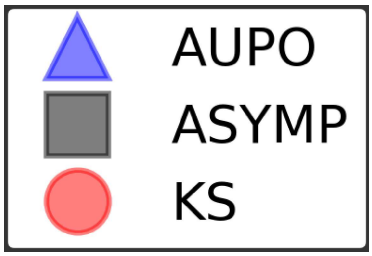
(k) Sailing Wind



(l) Tamarisk



(m) Traffic



Legend

Figure 9: The performance graphs (↑) in dependence of the MCTS iteration count of the parameter optimized versions of standard AUPO, AUPO using KS-distance (KS), and AUPO using asymptotic confidence interval (ASYMP).

Setting	Default value
Discount γ	1
Min eval episodes	2000
Episode horizon	50
Confidence interval levels	99%
RNG seed	42 (best parameter reruns with seed 4269)
Compilation	g++ version 10 with `-O3` flag
Hardware	Slurm cluster (1 CPU, 1GB RAM per job) using the following CPU types: Intel(R) Core(TM) i7-7820X CPU @ 3.60GHz, Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz, Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz, AMD EPYC 72F3 8-Core Processor, Intel(R) Xeon(R) CPU E5-2690 v2 @ 3.00GHz, Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz, Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz.
Academic Advising params	map: AcademicAdvisingCourses/2_Anand.txt, dense_rewards:True
Cooperative Recon params	map: CooperativeReconSetups/3_IPPC.txt
Crossing Traffic params	idle_action: 1; width: 4; height: 3; spawn_rate: 0.5
Earth Observation params	map: EarthObservationMaps/1_IPPC.txt
Game of Life params	map: GameOfLifeMaps/3_Anand.txt
Manufacturer params	map: ManufacturerSetups/3_IPPC.txt
Multi-armed Bandit params	means: 10.0; 9.0, stds: 1.0; 10.0, repeats: 10
Navigation params	map: NavigationMaps/3_Anand.txt
Push Your Luck params	map: DiceProbs/10_IPPC.txt
Racetrack params	map: Racetracks/ring-2.track
Sailing Wind params	size: 15
Saving params	p: 4; t: 4
Skill Teaching params	map: SkillsTeachingSkills/5_IPPC.txt
SysAdmin params	map: SysAdminTopologies/4_Anand.txt
Tamarisk params	map: TamariskMaps/2_IPPC.txt
Traffic params	map: TrafficModels/1_IPPC.txt
Wildfire params	map: WildfireSetups/1_IPPC.txt

Table 6: Overview of the default parameter setting used for the experiments of this paper.