



Trading-Off Statistical and Computational Efficiency via W -Step Markov Decision Processes: A Policy Gradient Approach

Gianmarco Tedeschi¹ · Marco Mussi¹ · Alberto Maria Metelli¹ · Marcello Restelli¹

Received: 27 October 2025 / Revised: 27 April 2026 / Accepted: 21 May 2026
© The Author(s) 2026

Abstract

In reinforcement learning, the performance of an algorithm is typically evaluated along two dimensions: computational and statistical complexity. While theoretical researchers often prioritize statistical efficiency—minimizing the number of samples needed to reach the desired accuracy—practitioners focus mainly on reducing computational costs, such as training time and resource consumption. Bridging these two perspectives requires algorithms able to deliver strong statistical guarantees while remaining computationally efficient in practice. In this paper, we introduce *MetaStep*, a meta-algorithm designed to enhance state-of-the-art RL algorithms by improving their computational efficiency while maintaining competitive sample efficiency. *MetaStep* is based on the novel notion of *W-step Markov decision process* (MDP), where, instead of performing a single action and transitioning to the next state, the agent executes a sequence of W actions before observing the resulting state and collecting the discounted W -step cumulative reward. First, we provide a theoretical analysis of the suboptimality introduced in the optimal policy performance when planning in a W -MDP, highlighting the impact of the environment stochasticity. Second, we apply *MetaStep* to GPOMDP, a well-known policy gradient method, and theoretically investigate the advantages of learning in the W -MDP in terms of variance reduction and improved sample complexity. Finally, empirical evaluations confirm that *MetaStep* reduces computational costs while preserving sample efficiency.

Keywords Reinforcement learning · Policy gradient · Open-loop · Sample complexity

Editors: Zahraa S. Abdallah, Annalisa Appice, Przemyslaw Biecek, Andrea Tagarelli.

✉ Gianmarco Tedeschi
gianmarco.tedeschi@polimi.it
Marco Mussi
marco.mussi@polimi.it
Alberto Maria Metelli
albertomaria.metelli@polimi.it
Marcello Restelli
marcello.restelli@polimi.it

¹ Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milan, Italy

1 Introduction

Reinforcement learning (RL, Sutton & Barto, 2018) has achieved remarkable success across a wide spectrum of complex real-world control tasks, such as robotic manipulation (Peters & Schaal, 2006), autonomous driving (Likmeta et al., 2020), game playing (Silver et al., 2018), and financial portfolio management (Jiang & Liang, 2017). Among RL methods, *policy gradient* (PG, Sutton et al., 1999a) approaches have gained significant popularity (Deisenroth et al., 2013) due to their simplicity (Silver et al., 2014), flexibility, and effectiveness (Peters & Schaal, 2008) in handling high-dimensional continuous control problems, which are prevalent in many real-world settings.

Despite these advances, implementing RL solutions in large-scale real-world control problems remains challenging due to two main limitations: sample inefficiency and high computational cost. Indeed, RL algorithms typically require extensive interaction with the environment to learn effective behaviors (Sutton & Barto, 2018), making them expensive and often impractical in resource-constrained scenarios. While theoretical research focuses on improving sample efficiency, practitioners often rely on simulated environments where data is abundant (Silver et al., 2016) and where the use of computationally inefficient learning algorithms to process the data may become the bottleneck.

One of the major challenges in RL is represented by the *curse of horizon* (Liu et al., 2018), which refers to the exponential growth in complexity w.r.t. the decision horizon. As the time horizon extends, the estimation of future rewards becomes increasingly uncertain due to the accumulation of errors. This uncertainty can result in suboptimal behavior, as the impact of distant future rewards diminishes compared to immediate rewards, complicating the learning process (Liu et al., 2020). This phenomenon impacts sample complexity, as longer horizons require more data to accurately estimate long-term returns and reduce variance in gradient-based policy search. From a computational complexity perspective, longer horizons demand more resources to simulate and backpropagate through lengthy decision sequences. This represents a serious challenge for time- or sample-constrained applications. To address the curse of horizon, several approaches have been explored from both perspectives. From a statistical complexity point of view, *hierarchical RL* (Sutton et al., 1999; Barto & Mahadevan, 2003; Pateria et al., 2022) methods create temporal abstractions in the action space, allowing for more structured decision-making and partitioning of the time horizon. Instead, *action-persistence* (Metelli et al., 2020; Sabbioni et al., 2023) focuses on identifying the best action to be repeated several times to reduce the effective planning horizon and consequently the sample complexity. However, forcing the same action to be repeated can sometimes oversimplify the problem and result in too much rigidity. Finally, *open-loop* approaches (Bubeck & Munos, 2010), which involve executing a sequence of actions without intermediate feedback, simplify the decision process over long horizons at the cost of poor performance in highly stochastic environments, where reactive behavior is critical. From the computational complexity perspective, approaches like *frame skipping* (Shivaram Kalyanakrishnan et al., 2021) and *action persistence* (Metelli et al., 2020), where an agent selects and repeats an action for several frames or time steps without further decisions, have been shown to reduce inference overhead by decreasing the number of decisions an agent must compute per episode. While effective in lowering computational demands, these approaches may compromise precision in tasks requiring fine-grained control.

In this paper, we tackle the curse of horizon problem in order to reduce both the *statistical* and *computational* complexity. Our goal is to bridge *action-persistence* and *open-loop* strategies to combine their complementary advantages: reducing the effective horizon while alleviating the computation burden during training. Action persistence offers various benefits, particularly in real-world control tasks where the optimal action evolves smoothly across states, as is common in many physical systems. On the other hand, open-loop strategies significantly reduce computational demands by eliminating the need for frequent state observations to influence the action selection process.

Original Contribution This paper introduces the novel framework of *W-step Markov decision processes* (*W*-MDPs), where the agent plays sequences of *W* actions without observing intermediate states. We design *MetaStep*, a meta-algorithm designed to be paired with RL approaches, shifting the agent-environment interaction in *W*-MDPs. Eventually, we adopt *MetaStep* on top of GPOMDP (Baxter & Bartlett, 2001), presenting a statistically and computationally efficient variant to learn in this setting. The contributions of this work are summarized in more detail below.

- In Sect. 2, we present *W-step Markov decision processes* (*W*-MDPs). Then, we derive a novel result upper bounding the difference in terms of optimal value functions between planning in the original MDP and planning in the *W*-MDP (Theorem 2.1), highlighting which features of the environment are responsible for the performance loss.
- In Sect. 3, we present *MetaStep*, our meta-algorithm, and we discuss how it translates into mutating the usual RL interaction protocol between agent and environment.
- In Sect. 4, we apply *MetaStep* on GPOMDP, a widely-adopted action-based policy gradient algorithm. We study and discuss its theoretical guarantees in terms of smoothness properties of the objective function and variance reduction, and show how these properties positively affect the convergence guarantees (Theorems 4.1 and 4.2).
- In Sect. 5, we numerically evaluate our solution over GPOMDP for different window sizes *W* and discuss its empirical performance compared to the base implementation of GPOMDP both in terms of learning speed and computation time.

Related works are discussed in Sect. 6. The proofs of all the statements are provided in Appendix A. Additional experiments and experimental details are discussed in Appendix B.

2 Setting

In this section, we start by introducing the background notions on MDPs (Sect. 2.1). Then, we present the *W*-MDP framework and the learning problem (Sect. 2.2). Finally, we study the optimal performance bias for *W*-MDPs (Sect. 2.3).

2.1 Preliminaries

Markov Decision Processes A Markov Decision Process (MDP, Puterman, 2014) is defined as a tuple $\mathcal{M} := (\mathcal{S}, \mathcal{A}, P, r, \gamma, H, \mu_0)$, where $\mathcal{S}$ and $\mathcal{A}$ are the (continuous) state and action spaces with $d_{\mathcal{S}}$ and $d_{\mathcal{A}}$ denoting the dimensions of the state and action vec-

tors, respectively, $P : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})^1$ is the transition model, where $P(s'|s, a)$ specifies the probability of landing in state $s' \in \mathcal{S}$ after having played action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$, $r : \mathcal{S} \times \mathcal{A} \rightarrow [-R_{\max}, R_{\max}]$ is the reward function, where $r(s, a)$ specifies the reward the agent gets by playing action a in state s and $R_{\max} \in \mathbb{R}_{>0}$, $\gamma \in [0, 1]$ is the discount factor, $H \in \mathbb{N} \cup \{+\infty\}$ is the time horizon and $\mu_0 \in \Delta(\mathcal{S})$ is the initial-state distribution.² A trajectory $\tau = (s_0, a_0, \dots, s_{H-1}, a_{H-1})$ of length H is a sequence of H state-action pairs. The discounted return of a trajectory τ is $R(\tau) := \sum_{h=0}^{H-1} \gamma^h r(s_h, a_h)$.

Policies, Value Functions, and Optimality The agent's behavior is governed by a (possibly stochastic) Markovian policy $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$ where $\pi(a|s)$ is the probability to play action $a \in \mathcal{A}$ when in state $s \in \mathcal{S}$. The goal of an agent is to find the policy that maximizes the *expected cumulative discounted reward* $J(\pi) = \mathbb{E}_{\tau \sim p(\cdot|\pi)} [R(\tau)]$, where $p(\cdot|\pi)$ is the distribution over trajectories induced by policy π . Moreover, we define the *state value function* (Sutton & Barto, 2018) as the expected discounted cumulative reward induced by policy π in state $s \in \mathcal{S}$: $V^\pi(s) = \mathbb{E}_{\tau \sim p(\cdot|\pi)} [\sum_{h=0}^{H-1} \gamma^h r(s_h, a_h) | s_0 = s]$. We define π^* as an optimal policy as a policy maximizing the expected discounted reward for each state $s \in \mathcal{S}$: $\pi^* \in \operatorname{argmax}_{\pi \in \Pi} V^\pi(s)$, where Π is the set of Markovian policies. We denote with $V^*(s) = V^{\pi^*}(s)$ the value function of an optimal policy π^* . We consider parametric policies π_θ , i.e., policies characterized by a parameter vector θ . We use the notations $J(\theta)$ and $p(\cdot|\theta)$ to express the expected return and density over trajectories of policy π_θ .

2.2 W-Step Markov Decision Processes

Building on the definition of MDPs, we introduce the new concept of *W-step Markov decision processes* (*W-MDPs*), which redefines MDPs to handle *W*-step actions.

Definition 2.1 Let $\mathcal{M}$ be an MDP. For every $W \in \mathbb{N}_{\geq 1}$, the *W-MDP* is defined as $\mathcal{M}_W = (\mathcal{S}, \mathcal{A}^W, P_W, r_W, \bar{H}, \bar{\gamma}, \mu_0)$, where $\mathcal{A}^W = \{\mathbf{a} = (a_1, \dots, a_W) : a_i \in \mathcal{A}, \forall i \in [W]\}$ is the space of *W*-step actions, $P_W : \mathcal{S} \times \mathcal{A}^W \rightarrow \Delta(\mathcal{S})$ is the *W*-step transition model recursively defined as $P_1 = P$ and for $W \in \mathbb{N}_{\geq 2}$, $s \in \mathcal{S}$, and $(a_1, \dots, a_W) \in \mathcal{A}^W$ as:

$$P_W(\cdot | s, (a_1, \dots, a_W)) = \int_{\mathcal{S}} P_{W-1}(s' | s, (a_1, \dots, a_{W-1})) P(\cdot | s', a_W) ds',$$

$r_W : \mathcal{S} \times \mathcal{A}^W \rightarrow \left[-\frac{1-\gamma^W}{1-\gamma} R_{\max}, \frac{1-\gamma^W}{1-\gamma} R_{\max}\right]$ is the *W*-step reward function defined as $r_1 = r$ and for $W \in \mathbb{N}_{\geq 2}$, $s \in \mathcal{S}$, and $(a_1, \dots, a_W) \in \mathcal{A}^W$ as:

$$r_W(s, (a_1, \dots, a_W)) = r(s, a_1) + \gamma \int_{\mathcal{S}} P(s' | s, a_1) r_{W-1}(s', (a_2, \dots, a_W)) ds',$$

$\bar{H} = \lceil H/W \rceil$ is the time horizon, and $\bar{\gamma} = \gamma^W$ is the discount factor.

Therefore, this definition of *W-MDP* generalizes standard MDPs, as for $W = 1$, we recover the base MDP, i.e., $\mathcal{M} = \mathcal{M}_1$. In a *W-MDP*, the policy is defined as function

¹We denote with $\Delta(\mathcal{X})$ the set of probability measures over the measurable set $\mathcal{X}$

²We admit $H = \infty$ only for $\gamma < 1$.

$\pi_W : \mathcal{S} \rightarrow \Delta(\mathcal{A}^W)$. At each decision step (i.e., when $h \bmod W = 0$), the agent samples a sequence of actions according to its policy and sequentially plays it for the next W steps. The parameter $W \geq 1$ can be viewed as an environmental parameter (influencing P , r , H , and γ) that can be optimized to improve the agent's learning process. Notably, the effective horizon in $\mathcal{M}_W$ is reduced to $\frac{1}{1-\gamma^W} < \frac{1}{1-\gamma}$. The effect of open-loop actions is similar to reducing the planning horizon, either by explicitly reducing the task's discount factor and the trajectory length. As we will present in Sects. 4 and 5, the reduced effective horizon provably brings improvements both in terms of statistical and computational complexity.

2.3 Performance Bias on W -MDPs

The W -MDP allows for a lower effective horizon by aggregating steps. This aggregation is convenient in environments with deterministic transition probabilities but may weaken the performance when the environment is highly stochastic, as we operate partially open-loop, losing controllability. In this part, we study how much we lose in terms of the optimal value function because we are not fully able to control what happens in the intermediate steps and fully manage the environment's stochasticity.

Before that, we introduce the W -step value function $V_W^{\pi_W}$, which evaluates the performance of policy π_W in the W -MDP starting from any state $s \in \mathcal{S}$:

$$V_W^{\pi_W}(s) = \mathbb{E}_{\tau \sim p(\cdot | \pi_W)} \left[\sum_{h=0}^{\bar{H}-1} \bar{\gamma}^h r_W(s_{Wh}, (a_{Wh}, \dots, a_{Wh+W-1})) \middle| s_0 = s \right]. \quad (1)$$

An optimal policy π_W^* is a policy maximizing the W -step value function from every state $s \in \mathcal{S}$, i.e., $\pi_W^* \in \operatorname{argmax}_{\pi_W \in \Pi_W} V_W^{\pi_W}(s)$, where Π_W is the set of Markovian policies for the W -MDP. We define the *optimal W -value function* as $V_W^*(s) = V_W^{\pi_W^*}(s)$. We are now ready to bound the difference in the optimal value function when moving from the original MDP to the W -MDP.

Theorem 2.1 *Let $\mathcal{M}$ be an MDP, let $W \in \mathbb{N}_{\geq 1}$, and let $\mathcal{M}_W$ be the corresponding W -MDP. Then, for every $s \in \mathcal{S}$, it holds that:*

$$V^*(s) - V_W^*(s) \leq 4R_{\max} D(P) \frac{\gamma(1 - \gamma^{W-1})}{(1 - \gamma)^2(1 - \gamma^W)},$$

where:

$$D(P) := \max_{(s,a) \in \mathcal{S} \times \mathcal{A}} \left\{ \min_{s' \in \mathcal{S}} \{1 - P(s'|s, a)\} \right\}.$$

Some comments are in order. As expected, for $W = 1$, the performance difference between the W -MDP and the normal MDP is equal to 0, i.e., $V^*(s) = V_1^*(s)$. Furthermore, we notice that the difference gap increases *linearly* with $D(P)$, which represents the index of stochasticity of the original transition model P . When P is deterministic, then $D(P) = 0$ and, consequently, no suboptimality is introduced, i.e., $V^*(s) = V_W^*(s)$. This is compli-

ant with the known fact that open-loop control is optimal for deterministic environments. Instead, when P is “maximally stochastic”, $D(P)$ is maximized.³

3 MetaStep

In this section, we present `MetaStep`, providing an overview of the meta-algorithm, how it interacts with the environment, and showing the differences w.r.t. classic agent-environment interaction.

The classic RL, agent-environment interaction follows a continuous loop. Initially, the agent observes the current state s of the environment. Based on this observation, the agent selects a (single-step) action a following policy π , i.e., $a \sim \pi(\cdot|s)$. Subsequently, the agent plays action a , and the environment responds by transitioning to a new state s' and providing the agent with feedback in the form of a reward r , reflecting the immediate benefit or cost of the action. This process repeats for a number of steps H , as the agent attempts to adapt its behavior based on the environment’s responses.

Differently, in `MetaStep`, whose interaction protocol is presented in Algorithm 1, the agent interacts with an W -MDP (as defined in Definition 2.1). At each iteration, a sequence of $W \in \mathbb{N}_{\geq 1}$ actions is sampled from a policy π_W observing only the state at time step h when $h \bmod W = 0$. During the W sequence of steps, we collect the cumulative discounted reward for W steps and evaluate it as a single-step reward signal. Compared to the classic interaction protocol, `MetaStep` uses an inner loop of W iterations to manage the usage of actions and the collection of associated rewards, the next state s' is observed only after W steps and represents the consequence of the W actions played in the open-loop sequence. This behavior implicitly brings a trade-off in our approach. From one side, we do not have to observe the state at each interaction, reducing the effective horizon of the task and lightening the computational burden during the training phase, as we will see in Sect. 5 (the effective horizon of the task is inversely proportional to the parameter W). On the other hand, without observing the current configuration of the environment, we could end up obtaining suboptimal performance, especially in highly stochastic environments, as visible from Theorem 2.1.

Input: Horizon H , parameter W , policy π_W
for $h \in \llbracket [H/W] \rrbracket$ **do**
 Observe state s_h
 Sample action $\mathbf{a}_h = (a_{h,1}, \dots, a_{h,W}) \sim \pi_W(\cdot|s_h)$
 for $w \in \llbracket W \rrbracket$ **do**
 Play action $a_{h,w}$ and collect reward $r_{h,w}$
 end
end

Algorithm 1 `MetaStep`—interaction protocol

³For clarity, consider the finite state and action setting with $|S| = S$. If, for every (s, a) , the transition kernel is uniform over next states, i.e., $P(s' | s, a) = 1/S$ for all $s' \in S$, then the stochasticity index attains its maximum value, namely $D(P) = 1 - 1/S$.

4 MetaStep on GPOMDP

In this section, we investigate the performance of MetaStep when applied to GPOMDP (Baxter & Bartlett, 2001), which serves as a representative case due to its well-understood theoretical properties. We begin by illustrating how our meta-algorithm (Algorithm 1) can be instantiated within the GPOMDP framework, and then proceed to analyze its sample complexity.

Preliminaries on Policy Gradient and GPOMDP Policy gradients are a class of RL algorithms for searching the best-performing policy over a class of parametrized differentiable stochastic policies $\Pi_{\Theta} = \{\pi_{\theta} : \mathcal{S} \rightarrow \Delta(\mathcal{A}) : \theta \in \Theta \subseteq \mathbb{R}^d\}$, where Θ is the parameter space. At each iteration $k \in \llbracket K \rrbracket := \{1, \dots, K\}$, a dataset of N trajectories $\mathcal{T}_N^k = \{\tau_i\}_{i=1}^N$, where $N \in \mathbb{N}$ is the batch size, is collected using policy π_{θ_k} . Then, the policy parameter is iteratively updated via gradient ascent:

$$\theta_{k+1} = \theta_k + \eta \widehat{\nabla}_{\theta} J(\theta_k), \quad (2)$$

where $\eta > 0$ is the step size and $\widehat{\nabla}_{\theta} J(\theta_k)$ is an estimate of the policy gradient using the batch $\mathcal{T}_N^k$. The most common policy gradient estimators are REINFORCE (Williams, 1992) and GPOMDP (Baxter & Bartlett, 2001) that can be expressed as $\widehat{\nabla}_{\theta} J(\theta_k) = \frac{1}{N} \sum_{i=1}^N g(\tau_i | \theta_k)$ where $\tau_i \in \mathcal{T}_N^k$ for every $i \in \llbracket N \rrbracket$ and $g(\tau_i | \theta_k) = \nabla_{\theta} \log p(\tau_i | \theta_k) R(\tau_i)$. In this work, g will refer to the GPOMDP estimator, which is preferred over REINFORCE due to its smaller variance and is given by:

$$\widehat{\nabla}_{\theta} J(\theta_k) = \frac{1}{N} \sum_{i=1}^N \sum_{h=0}^{H-1} \gamma^h r(s_{i,h}, a_{i,h}) \sum_{u=0}^h \nabla_{\theta} \log \pi_{\theta_k}(a_{i,u} | s_{i,u}), \quad \tau_i \in \mathcal{T}_N^k, \quad i \in \llbracket N \rrbracket. \quad (3)$$

Specifically, the GPOMDP estimator exploits the concept of *rewards-to-go*, giving value to the rewards that come after taking an action, since the action performed before that, has no bearing on how good that action was. This enforces causality, which significantly reduces the variance of the estimator. Algorithm 2 reports the pseudocode of GPOMDP in its base implementation.

Input: Number of iterations K , batch size N , initial parameter vector θ_0 , horizon H , discount factor γ , learning rate η

Initialize $\theta \leftarrow \theta_0$

for $i \in \llbracket K \rrbracket$ **do**

 Set the stochastic policy parameters: π_θ

for $l \in \llbracket N \rrbracket$ **do**

 Initialize trajectory τ_l as an empty tuple

for $h \in \llbracket H \rrbracket$ **do**

 Observe state $s_{l,h}$

 Play action $a_{l,h} \sim \pi_\theta(\cdot | s_{l,h})$ and observe reward $r_{l,h}$

 Add to τ_l the tuple $(s_{l,h}, a_{l,h}, r_{l,h})$

end

end

 Compute the gradient estimator:

$$\hat{\nabla}_\theta J(\theta) \leftarrow \frac{1}{N} \sum_{l=1}^N \sum_{h=0}^{H-1} \left(\sum_{u=0}^h \nabla_\theta \log \pi_\theta(a_{l,u} | s_{l,u}) \right) \gamma^h r_{l,h}$$

 Update the policy parameter vector: $\theta \leftarrow \theta + \eta \hat{\nabla}_\theta J(\theta)$

end

Return θ

Algorithm 2 GPOMDP

4.1 Running MetaStep on GPOMDP

We now proceed to integrate GPOMDP (Algorithm 2) in the MetaStep procedure (Algorithm 1). The result is presented in Algorithm 3. Here, each episode is divided into H/W steps. For each step, we observe the state s_h and we sample a sequence of W actions from the parametric policy $\pi_{W,\theta}$. Each action in the sequence is played, and the corresponding reward r_w is observed. These rewards are aggregated into a so-called W -reward, which sums the rewards weighted by discount factors γ^{w-1} . The tuple (s_h, a_W, r_W) is then added to the trajectory τ_l . After data collection, the algorithm estimates the gradient of the objective function $J(\theta)$ with respect to θ , using an estimator based on the observed trajectories and discounted rewards, following the standard GPOMDP estimator, applied to the W -MDP. This new structure has several effects from both the computational and the statistical perspectives.

Input: Number of iterations K , batch size N , initial parameter vector θ_0 , parameter W , horizon H , original discount factor γ , discount factor γ , learning rate η

Initialize $\theta \leftarrow \theta_0$

for $i \in \llbracket K \rrbracket$ **do**

 Set the stochastic policy parameters: $\pi_{W,\theta}$

for $l \in \llbracket N \rrbracket$ **do**

 Initialize trajectory τ_l as an empty tuple

for $h \in \llbracket \lceil H/W \rceil \rrbracket$ **do**

 Observe state $s_{l,h}$

 Sample action sequence $\mathbf{a}_{l,h} = (a_{l,h,1}, \dots, a_{l,h,W}) \sim \pi_{W,\theta}(\cdot | s_{l,h})$

 Initialize W -reward $\rho_{l,h} \leftarrow 0$

for $w \in \llbracket W \rrbracket$ **do**

 Play action $a_{l,h,w}$ and observe reward $r_{l,h,w}$

 Update the W -reward $\rho_{l,h} \leftarrow \rho_{l,h} + \gamma^{w-1} r_{l,h,w}$

end

 Add to τ_l the tuple $(s_{l,h}, \mathbf{a}_{l,h}, \rho_{l,h})$

end

end

 Compute the gradient estimator:

$$\hat{\nabla}_{\theta} J(\theta; W) \leftarrow \frac{1}{N} \sum_{l=1}^N \sum_{h=0}^{\lceil H/W \rceil - 1} \left(\sum_{u=0}^h \nabla_{\theta} \log \pi_{W,\theta}(\mathbf{a}_{l,u} | s_{l,u}) \right) \gamma^{Wh} \rho_{l,h}$$

 Update the policy parameter vector: $\theta \leftarrow \theta + \eta \hat{\nabla}_{\theta} J(\theta; W)$

end

Return θ

Algorithm 3 MetaStep on GPOMDP

Computational Complexity From a computational perspective, the proposed approach offers two key advantages. First, recalling the GPOMDP estimator (Eq. 3), adopting MetaStep makes the estimation faster as one of the summations is over H/W terms instead of H . Second, we reduce the number of times we query the policy by a factor W . While the single queries may be more costly due to the extended temporal abstraction of the action, the overall reduction in query frequency leads to an improvement in the training phase, especially in settings where the policy evaluation might be a bottleneck. An extensive empirical evaluation of this phenomenon is provided in Sect. 5, where we examine the computational aspects across tasks of increasing complexity.

Statistical Complexity Here, we analyze the statistical complexity of MetaStep over GPOMDP. The smaller discount factor γ^W and the reduced effective horizon H/W lower the variance caused by the long horizon, even if the single rewards are larger and the problem's scale does not change.

Let $K \in \mathbb{N}$ be the number of iterations and $N \in \mathbb{N}$ the batch size; given an accuracy threshold $\epsilon > 0$, our objective is to bound the sample complexity NK . We use $J^*(W)$ to refer to the optimal performance with parameter W , i.e., $J^*(W) = \sup_{\theta \in \Theta_W} J(\theta; W)$ where with Θ_W we intend the parameter space of π_W policies, and with $J(\theta; W)$ the performance obtained by the policy parametrization θ in the W -MDP.

To support the theoretical analysis, we present the following assumptions on the smoothness of the performance function, which are extensions of standard assumptions made in the study of sample complexity for policy gradient algorithms (Yuan et al., 2022).

Assumption 4.1 There exist $G > 0$ such that for every $W \in \llbracket H \rrbracket$ the expected norm squared Euclidean of the gradient of $\log \pi_{W,\theta}(\cdot|s)$ satisfies:

$$\mathbb{E}_{a \sim \pi_{W,\theta}(\cdot|s)} \left[\|\nabla_{\theta} \log \pi_{W,\theta}(a|s)\|^2 \right] \leq G^2, \quad \forall s \in \mathcal{S}.$$

Assumption 4.2 There exist $F > 0$ such that for every $W \in \llbracket H \rrbracket$ the expected spectral norm of the Hessian of $\log \pi_{W,\theta}(\cdot|s)$ satisfies:

$$\mathbb{E}_{a \sim \pi_{W,\theta}(\cdot|s)} \left[\|\nabla_{\theta}^2 \log \pi_{W,\theta}(a|s)\| \right] \leq F, \quad \forall s \in \mathcal{S}.$$

Intuitively, these assumptions ensure that the effect on the environment dynamics and reward (and on their gradient), after playing an action, is controllable and independent from W . These assumptions are mild in several common scenarios. Let us focus on Assumption 4.1, as a similar reasoning holds for Assumption 4.2. If we assume that the partial derivatives of the log-policy with respect to individual parameters are uniformly bounded, the squared L_2 norm of the gradient scales linearly with the dimension of the parameter vector θ (i.e., the number of parameters). Consider now a neural network policy characterized by parameter vector θ . While increasing W linearly increases the number of parameters in the output layer to $d_A W X$, where X is the number of neurons per layer, the overall dimensionality of the policy is dominated by the structure of the hidden layers. Indeed, in a standard fully connected architecture, the number of parameters scales quadratically with the network width, i.e., $Z X^2$, where Z is the number of layers. Because this quadratic internal growth outweighs the linear expansion of the output, the gradient norm is mostly determined by the hidden layers. Therefore, treating G as a constant independent of W is a reasonable approximation for the theoretical analysis. Given Assumptions 4.1 and 4.2, we can now state a bound on the variance of the gradient estimators and on the smoothness of the performance function for W -MDPs.

Lemma 4.1 Under Assumption 4.1, the variance of the GPOMDP estimator with batch size N is bounded for every $W \in \llbracket H \rrbracket$ as:

$$\text{Var} [\widehat{\nabla}_{\theta} J(\theta; W)] \leq \frac{R_{\max}^2 G^2 (1 - \gamma^H)}{N(1 - \gamma)^2 (1 - \gamma^W)} \left(1 - \frac{H}{W} (\gamma^H - \gamma^{H+W}) - \gamma^H \right) =: \frac{V(W)}{N}.$$

Lemma 4.2 Under Assumptions 4.1 and 4.2, for every $W \in \llbracket H \rrbracket$, it holds that:

$$\|\nabla_{\theta}^2 J(\theta; W)\| \leq \frac{R_{\max} (G^2 + F)}{(1 - \gamma)(1 - \gamma^W)} \left(1 - \frac{H}{W} (\gamma^H - \gamma^{H+W}) - \gamma^H \right) =: L(W).$$

Lemmas 4.1 and 4.2 are needed in the analysis of sample complexity to guarantee a structure on the MDP (Yuan et al., 2022). We highlight that the bounds described in Lem-

mas 4.1 and 4.2 do not grow as W ,⁴ demonstrating how, under the assumptions above, the use of W -MDPs helps to reduce the effective horizon of the task and consequently also the variance of the gradient estimates.

In Fig. 1, we show the *bias variance* trade-off as a function of W . We notice how the optimality bias (stated in Theorem 2.1) follows a sublinearly increasing trend with W . Differently, the variance of the gradient estimation (Lemma 4.1) shows a descending trend as W increases. These effects result in an overall U-shaped behavior, highlighting the trade-off between estimation accuracy and stability when facing noisy environments. This highlights the importance of selecting an appropriate W , as for stochastic environments, it trades off bias and variance.

We are now ready to discuss the convergence to a stationary point.

Theorem 4.1 (*Convergence to Stationary Point*) Under Assumptions 4.1 and 4.2, with a suitable step size to guarantee $\mathbb{E}[\|\nabla_{\theta} J(\theta_U; W)\|] \leq \epsilon$, where U is sampled uniformly, the sample complexity of Algorithm 3 is at most:

$$NK \leq \tilde{O}\left(\frac{R_{\max}^3(1-\gamma^H)}{\epsilon^2(1-\gamma)^3(1-\gamma^W)^2} \left(1 - \frac{H}{W}(\gamma^H - \gamma^{H+W}) - \gamma^H\right)^2\right).$$

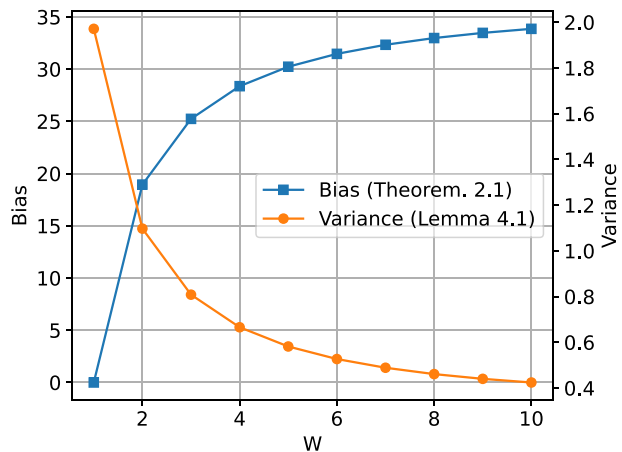
This result follows the analysis by Papini (2020). Here, the first term dominates, yielding the well-known $\mathcal{O}(\epsilon^{-2})$ rate. To also provide a global convergence result, we enforce the following assumption.

Assumption 4.3 There exists $\alpha > 0$ and $\beta \geq 0$ such that for every $\theta \in \Theta_W$, and for every $W \in \llbracket H \rrbracket$, it holds that:

$$J^*(W) - J(\theta; W) \leq \alpha \|\nabla_{\theta} J(\theta; W)\| + \beta.$$

Assumption 4.3 is a standard for global convergence analysis of stochastic optimization (Yuan et al., 2022). The *weak gradient domination* assumption admits that the per-

Fig. 1 Trend of the bias (Theorem 2.1) and variance (Lemma 4.1) bounds for different values of $W \in \{1, 2, \dots, 10\}$. Exemplification with $\gamma = 0.9$, $H = 10$, $N = 100$ and a stochasticity index $D(P) = 0.1$



⁴We provide a formal proof of this fact in Lemma A.5.

formance function $J(\theta; W)$ has local maxima as long as its performance is β -close to the globally optimal one. Given Assumption 4.3 and the results provided by Montenegro et al. (2024), we can now state the global convergence result.

Theorem 4.2 (Global Convergence) *Under Assumptions 4.1, 4.2 and 4.3, with a suitable step size to guarantee $\mathbb{E}[J^*(W) - J(\theta_K; W)] \leq \epsilon + \beta$, the sample complexity of Algorithm 3 is at most:*

$$NK \leq \tilde{O}\left(\frac{R_{\max}^3(1-\gamma^H)}{\epsilon^3(1-\gamma)^3(1-\gamma^W)^2} \left(1 - \frac{H}{W}(\gamma^H - \gamma^{H+W}) - \gamma^H\right)^2\right).$$

This result establishes a convergence of order $\tilde{O}(\epsilon^{-3})$ to the last-iterate global optimum convergence of stochastic policies in accordance with state-of-the-art PG analysis (Yuan et al., 2022).⁵ We notice in the sample complexity how the usage of W -MDPs manages to attenuate the curse of horizon by reducing the effective horizon by defining a new discount factor, $\gamma^W \geq \gamma$, which results in a lower sample complexity.

5 Experimental Validation

In this section, we show how MetaStep is effective in optimizing both the statistical and the computational complexity. We evaluate MetaStep applied to GPOMDP on three progressively difficult domains to analyze its performance under different settings. Regarding the choice of W , in principle, it can be determined, given knowledge of the environment, by optimizing the trade-off between the bias bound established in Theorem 2.1 and the sample complexities of Theorems 4.1 and 4.2. In practice, when the environment is unknown, one can either (i) estimate these quantities and empirically optimize this trade-off, although doing so sacrifices the theoretical guarantees, or (ii) treat W as an algorithmic hyperparameter and tune it experimentally. In this section, we follow approach (ii). For each environment, we test our approach with several values of $W \in \{1, 2, 3, 5, 10\}$, where for $W = 1$ the algorithm reduces to the standard version of GPOMDP. We adopt a stochastic Gaussian policy where the means for each action are parameterized using a neural network. Thus, we have a network whose input is the state s and the output is an action composed of Wd_A elements. The standard deviation has been tuned for each task. We refer the reader to Appendix B for experimental details, additional experiments and baselines. The code to used to run the experiments in this section can be found at <https://github.com/gianmteschi/metastep>.

For each task, we will present three plots. The left one—denoted with (a)—examines statistical complexity by illustrating the number of iterations (and thus the number of samples) required to achieve convergence for different values of W . The one in the middle—denoted with (b)—reports the training time as a function of the window size W . Lastly, the right one—denoted with (c)—displays the learning curves relative to training time, offering insight into both computational and statistical complexity.

Dam Control We first consider the water resource management task described in (Castelletti et al., 2010). The agent learns a water release policy balancing external demand D (e.g.,

⁵The $\tilde{O}(\cdot)$ notation hides logarithmic factors.

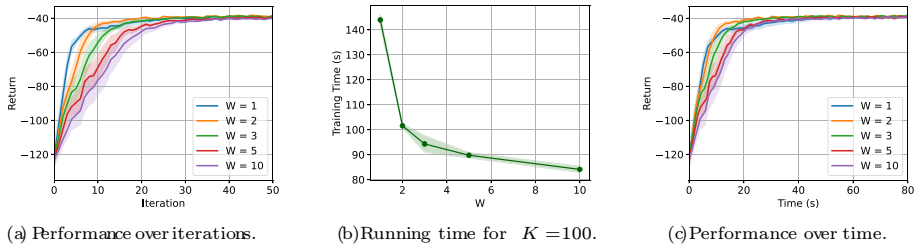


Fig. 2 Learning curves and training time for the dam control task (10 runs, mean $\pm$ 95% C.I.)

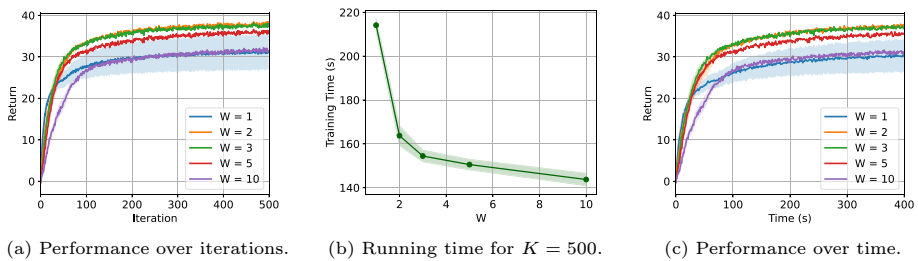


Fig. 3 Learning curves and training time for Swimmer (10 runs, mean $\pm$ 95% C.I.)

for a town) while preventing floods by keeping the water level below a threshold F . The dam experiences a stochastic daily inflow i_t , representing external factors such as rainfall, which follow a periodic yearly pattern. The demand remains constant. The system's state evolves according to the mass balance equation $s_{t+1} = \max\{s_t - a_t + i_t, 0\}$ where a_t is the water released on day t . The reward function, $R(s_t, a_t)$, is a convex combination of two objectives: flooding control, $-c_1 \max(0, s_t - F)$, and demand satisfaction, $-c_2 \max(0, D - a_t)^2$, with $c_1, c_2 > 0$ as domain-specific constants. The action space is scalar ($d_A = 1$), while the state space has dimension $d_S = 7$. The discount factor is $\gamma = 0.999$. We train our agent for $H = 1825$ steps (equivalent to 5 years of data) with batch size $N = 10$.

Results are presented in Fig. 2. Starting from Fig. 2a Instead, in Fig. 2b we can appreciate the computational advantages brought by larger values of W . The plot shows the training time needed (in terms of CPU time) for a total of $K = 100$ iterations, illustrating a decreasing trend as the value of W increases. Finally, in Fig. 2c under a different perspective, by looking at the performance over training time. We observe that the larger values of W reduce the training time needed to reach higher performances, particularly with $W \in \{2, 3\}$.

Swimmer The Swimmer environment in MuJoCo (Todorov et al., 2012) is a continuous control problem where a multi-jointed agent must move itself forward in a simulated fluid environment. The action vector dimension $d_A = 2$, while the state vector dimension $d_S = 8$. We train our agent for $H = 700$ steps, using a batch size $N = 100$ and a discount factor $\gamma = 0.995$.

Figure 3a shows the learning curve over samples. We observe that larger values of W lead to faster convergence and higher final performance. We conjecture that, this phenomenon is due to the search in the space of Π_W policies, which induces a different manifold that might avoid plateaus during optimization. In Fig. 3b.

Half-Cheetah The Half-Cheetah environment in MuJoCo (Todorov et al., 2012) is a continuous control benchmark where a bipedal robot with a torso and two legs must learn to move forward efficiently. The action vector has dimension $d_A = 6$, while the state vector has dimension $d_S = 17$. We train the agent for $H = 200$ steps, using a batch size $N = 10$ and a discount factor $\gamma = 0.999$. The lower batch size, which leads to a noisy estimate of the gradients—especially in this high-dimensional scenario—is intended to evaluate the performance of *MetaStep* in this challenging situation.

In Fig. 4a, we empirically validate Lemma 4.1, as with a smaller batch size (10 against 100 of the previous simulation on Swimmer), we manage to estimate a less noisy gradient and consequently reach better performances w.r.t. the single-step approach. Moreover, for farsighted problems (i.e., high values of γ), we enjoy a better sample complexity. From the computational perspective, Fig. 4b, also in this environment, demonstrates a clear reduction in training time as increases. Finally, in Fig. 4c, we observe how, looking at the computational complexity, $W = 2$ presents a significantly faster convergence.

Overall Results The experimental campaign demonstrates the effectiveness of the *MetaStep* approach in balancing statistical and computational efficiency. The results highlight a fundamental trade-off linked to the choice of the window parameter W . On the one hand, increasing W reduces the problem's effective horizon. This leads to a decrease in the variance of the gradient estimate, as theorized in Lemma 4.1, and a significant reduction in training time. This often results in faster convergence, both in terms of wall-clock time and, in some cases, sample efficiency, as observed in the Swimmer and Half-Cheetah experiments. On the other hand, larger values of W increase the size of the action space, making the learning problem substantially more complex. The experiments show that while moderate values of W (e.g., 2, 3) yield clear benefits, excessively high values (e.g., 10) can degrade final performance. This occurs because the difficulty of optimizing a policy in such a vast action space outweighs the advantages of a shorter horizon. The choice of W represents a critical balance. An optimal value must shorten the horizon sufficiently to gain computational and stability benefits without expanding the action space to a point where learning becomes ineffective.

6 Related Works

In this section, we discuss the relevant literature for this work. First, we present closed-loop policy gradient approaches. Then, we discuss open-loop learning solutions and action persistence methods. Finally, we present temporal abstraction methods.

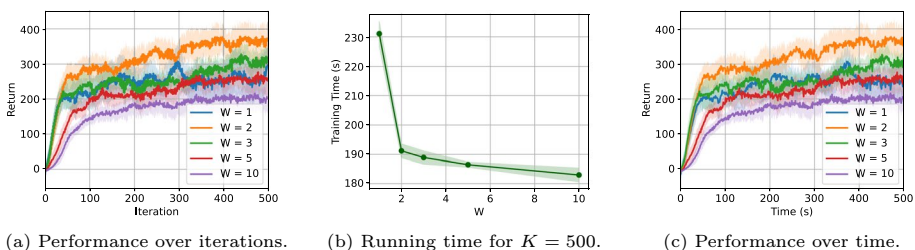


Fig. 4 Learning curves and training time for Half-Cheetah (10 runs, mean $\pm$ 95% C.I.)

Policy Gradient Methods PG methods constitute a fundamental class of RL algorithms that directly optimize a parameterized policy by following the gradient of an expected performance measure. Early work by Williams (1992) introduced the REINFORCE algorithm, which laid the foundation for policy gradient techniques. Subsequent advancements, such as the GPOMDP algorithm (Baxter & Bartlett, 2001) and *actor-critic* approaches (Sutton et al., 1999a), addressed the high variance of gradient estimates by incorporating *reward-to-go* and *baselines*. In general, the problem of variance reduction in policy gradient estimation has attracted the attention of many researchers. While earlier methods focused on the usage of *natural gradients* (Sham & Kakade, 2001), recent work moved through sample reuse to reduce estimation variance and improve sample efficiency. Among these methods, SVRPG (Papini et al., 2018) exploits the stochastic gradient at the current iteration with that of a past parametrization. More recent methods, such as SRVR-PG (Xu et al., 2020), DEF-PG (Paczolay et al., 2024) and STORM-PG (Yuan et al., 2020) integrate momentum-based gradient estimation and recursive gradient estimation to achieve better convergence rate. Eventually, recent methodologies as RT-PG (Montenegro et al., 2025) mixed importance sampling and trajectory reuse to efficiently reduce sample complexity

rates for convergence to a stationary point to $\tilde{O}(\epsilon^{-1})$, under specific assumptions. Despite these theoretical advancements, these methods aim to reduce only sample complexity and often introduce significant algorithmic complexity, making computational efficiency still a challenge. Our work introduces a novel approach that mitigates these issues by leveraging open-loop sequences of actions, effectively reducing the number of required policy updates while maintaining competitive sample efficiency.

Open-loop Approaches Open-loop planning in sequential decision problems refers to decision-making strategies where a sequence of actions is planned *entirely* in advance without incorporating immediate feedback from the environment at each step (Bubeck & Munos, 2010). This approach is particularly useful in domains where the real-time evaluation of actions is computationally expensive or in settings where the state of the environment is not (completely) observable (Chih-Han et al., 2005). Open-loop planning performs poorly in stochastic settings and non-generative models. To overcome this limitation, an approach of mixed planning has been proposed in literature (Hansen et al., 1996), mixing open-loop and closed-loop feedback planning. However, it provides only a value-based solution, which scales poorly with the dimensionality of the state and action space, with relevant limitations on the environment exploration. More recent solutions involve the mixed use of open-loop and closed-loop techniques constrained to specific operational regions of the task (Kolter et al., 2010). In our work, we propose a task-agnostic solution that embraces open-loop control methodologies but provides the agent with feedback after a fixed number of decision-making steps.

Action Persistence Action persistence in sequential decision problems is a technique where an agent commits to executing the same action over multiple time steps. This approach has been shown to reduce the complexity of decision-making and improve learning efficiency in specific environments. Prior work has explored the effects of persistent actions in both value-based (Metelli et al., 2020; Sabbioni et al., 2023) and policy-based RL (Lee et al., 2020). More complex approaches have been proposed in the literature, which involve actor-critic structures (Yu et al., 2021) or Deep RL for policy approximation (Tong et al., 2023). However, these approaches tend to be too rigid and require properly setting the

task control frequency. Furthermore, they poorly scale with the increase in dimensionality of the action space. Our work aligns with the philosophy of action persistence but provides more flexibility to the framework.

Temporal Abstraction Temporal abstraction in RL aims to improve sample efficiency and planning capabilities by structuring decision-making at multiple time scales. Options (Sutton et al., 1999) and macro-actions (McGovern & Barto, 2001) provide a hierarchical framework in which temporally extended actions facilitate long-term reasoning and efficient exploration. The MAXQ (Dietterich, 2000) is a classic example of a hierarchical solution. At its core, MAXQ proposes a value function decomposition, dividing the main MDP into smaller sub-MDP components. In this way, each sub-MDP is associated with a sub-task that can be learned independently from the others. On the Deep RL side, (Tejas et al., 2016) proposed a solution consisting of a two-level hierarchy of Deep Q-Networks (DQN, Mnih et al., 2015) that represent the high and lower level policies, allowing for a hierarchical approach in large and high-dimensional state spaces. Another class of HRL approach involves both the learning of the hierarchical policy and the semi-MDP structure (*sub-task discovery*). The skill chaining approach (Konidaris & Barto, 2009), incrementally construct options while learning an HRL agent. This process starts with one option created with its own subgoal set. The usage of a classifier is exploited to define new options based on the possibility of reaching a state in a predefined number of steps. More recently, meta-learning and learned subroutines (Bacon et al., 2017; Harb et al., 2018) have been employed to autonomously discover useful action sequences, reducing the complexity of long-horizon tasks. Specifically, Option Critic (OC, Bacon et al., 2017), let the agent use the sub-tasks (options) from the initial episode of learning. The solution proposed by OC allows to learn the options along with the entire hierarchy structure from the beginning of the learning process without the necessity of defining the hierarchical problem beforehand. In (Harb et al., 2018), a refinement of the OC, Asynchronous Advantage Option Critic (A2OC), has been proposed, introducing a regularizer cost that penalizes the high-level policy upon switching options, encouraging the retention of each option for a longer horizon. This approach proved empirically to have better performance than the frequently switching options. Other works, like (Barreto et al., 2019) adopts pseudo-rewards (e.g., cumulants) to represent option which can be combined linearly to synthesize new options without requiring additional learning.

While these methods effectively address long-term credit assignment, they often require significant computational resources and complex architectures. Our work aligns with temporal abstraction by introducing W -step action sequences that effectively function as macro-actions. However, unlike standard hierarchical RL methods, our approach does not require hierarchical structures or additional meta-learning components, making it a more lightweight and easily integrable enhancement to existing state-of-the-art algorithms. For a complete discussion on temporal abstraction, we refer the reader to (Pateria et al., 2022).

7 Discussion and Conclusions

In this paper, we introduced the novel concept of W -step Markov Decision Processes, which mixes open-loop and closed-loop interaction, and designed MetaStep a meta-algorithm exploiting the W -MDPs framework. In this framework, we observe a state and execute a sequence of W actions in an open-loop fashion. We formalized the W -MDP framework

and we define the notions of value function and optimality. Then, we showed how we may experience a suboptimality bias for highly stochastic environments. Building on this, we proposed a *meta-algorithm*, *MetaStep*, which redefines the agent-environment interaction by applying it over W -MDPs. *MetaStep* is designed to be used on top of standard RL approaches, and, in this paper, we adopted GPOMDP as a base learning algorithm. We discussed the improvement of our solution from both the computational and the statistical perspectives. Finally, we proposed an extensive experimental campaign to empirically validate our results. Such results demonstrate the capability of our approach to keep competitive statistical performances while reducing the computational complexity.

Limitations and Future Works While the use of the W -MDPs framework shows promising results both theoretically (Sects. 2 and 4) and empirically (Sect. 5), the approach still has some limitations. First, it currently supports only fixed values of W , leaving the choice of this hyperparameter to human expertise. The window size must balance several factors, such as the bias–variance trade-off (Fig. 1), which is particularly important in highly stochastic environments (or where the level of stochasticity is unknown), and the trade-off between a shorter effective horizon and an expanded action space. Future works may explore adaptive window sizes, as relying on a fixed open-loop window can be overly restrictive in tasks requiring finer control. Another direction is to study the applicability of core idea of the W -MDP to value-based RL (e.g., temporal difference methods) and of *MetaStep* to other state-of-the-art policy search methods, including actor–critic approaches. Finally, extending the framework to large-scale or real-world domains (e.g., robotics or control of complex systems) would offer valuable insights into its practical impact.

Appendix A: Omitted Proofs

In this appendix, we provide the proofs of all the theorems presented in the main paper.

Theorem 2.1 *Let $\mathcal{M}$ be an MDP, let $W \in \mathbb{N}_{\geq 1}$, and let $\mathcal{M}_W$ be the corresponding W -MDP. Then, for every $s \in \mathcal{S}$, it holds that:*

$$V^*(s) - V_W^*(s) \leq 4R_{\max}D(P) \frac{\gamma(1 - \gamma^{W-1})}{(1 - \gamma)^2(1 - \gamma^W)},$$

where:

$$D(P) := \max_{(s,a) \in \mathcal{S} \times \mathcal{A}} \left\{ \min_{s' \in \mathcal{S}} \{1 - P(s'|s, a)\} \right\}.$$

Proof Let us first introduce *state-action value function* as the expected discounted cumulative reward in state $s \in \mathcal{S}$ when we chose action $a \in \mathcal{A}$ and then follow policy π :

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{h=0}^{H-1} \gamma^h r(s_h, a_h) \mid s_0 = s, a_0 = a \right].$$

For a W -MDP, the definition of $Q_W^{\pi_W}$ follows directly.

We start the proof by observing that, for every MDP and corresponding W -MDP, we have $Q_1^*(s, a) \geq Q_W^*(s, a)$ for every $W \geq 1$.

Then, we consider the non-stationary transition model defined as follows for every h :

$$\bar{P}_W(s'|s, a, h) = \begin{cases} P(s'|s, a) & \text{if } h \bmod W = 0 \\ \mathbb{1}_{\{\operatorname{argmax}_{\bar{s} \in \mathcal{S}} P(\bar{s}|s, a)\}} & \text{otherwise} \end{cases},$$

where in the argmax ties are broken arbitrarily.

We now highlight the dependence on the W and on the used transition model $\bar{P}$ in the Q as $Q_{W, \bar{P}}^*(s, a)$. It is trivial to observe that for this non-stationary deterministic transition model, we have for every (s, a) :

$$Q_{1, \bar{P}_W}^*(s, a) = Q_{W, \bar{P}_W}^*(s, a),$$

as where we are not able to take action, we have deterministic transitions. As a consequence, we observe that in the MDP with transition model $\bar{P}_W$ we have⁶:

$$\max_{\pi: \mathcal{S} \rightarrow \mathcal{A}} \mathbb{E} \left[\sum_{h=1}^{H-1} \gamma^h r_h \right] = \max_{\pi_W: \mathcal{S} \rightarrow \mathcal{A}^W} \mathbb{E} \left[\sum_{h=1}^{H-1} \gamma^h r_h \right].$$

We now aim to bound the difference between $Q_{1, P}^*$ and $Q_{W, P}^*$. This is equal to:

$$\begin{aligned} Q_{1, P}^* - Q_{W, P}^* &= Q_{1, P}^* - Q_{W, P}^* \pm Q_{W, \bar{P}_W}^* \\ &= Q_{1, P}^* - Q_{1, \bar{P}_W}^* + Q_{W, \bar{P}_W}^* - Q_{W, P}^* \\ &\leq Q_{1, P}^* - Q_{1, \bar{P}_W}^* + Q_{W, \bar{P}_W}^* - Q_{W, P}^*, \end{aligned}$$

where π_1^* denotes an optimal policy in the MDP with transition model P and $\bar{\pi}_W^*$ denotes an optimal policy in the MDP with transition model $\bar{P}_W$. Since we are to bound the difference between two Q -functions of the same policy under different transition models, we proceed by analyzing the absolute value of one of the two terms for a generic non-stationary non-Markovian policy $\pi(a_l|H_l, l)$, where H_l is the history of all observations realized up to stage l (this includes both cases $\pi = \pi_1^*$ and $\pi = \bar{\pi}_W^*$):

⁶We allow the policies to be non-stationary, but, as we shall see, this is irrelevant for the proof.

$$\begin{aligned}
& \left| Q_{1,P}^\pi(s, a) - Q_{1,\bar{P}_W}^\pi(s, a) \right| \\
&= \left| \mathbb{E}_{P,\pi} \left[\sum_{t=0}^{H-1} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a \right] - \mathbb{E}_{\bar{P}_W,\pi} \left[\sum_{t=0}^{H-1} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a \right] \right| \\
&= \left| \int \left(\prod_{l=0}^{H-1} P(s_{l+1} | s_l, a_l) \pi(a_l | H_l, l) - \prod_{l=0}^{H-1} \bar{P}_W(s_{l+1} | s_l, a_l, l) \pi(a_l | H_l, l) \right) \sum_{t=0}^{H-1} \gamma^t r(s_t, a_t) d\tau \right| \\
&= \left| \sum_{t=0}^{H-1} \gamma^t \int \left(\prod_{l=0}^t P(s_{l+1} | s_l, a_l) \pi(a_l | H_l, l) - \prod_{l=0}^t \bar{P}_W(s_{l+1} | s_l, a_l, l) \pi(a_l | H_l, l) \right) r(s_t, a_t) d\tau \right| \\
&= \left| \sum_{t=0}^{H-1} \gamma^t \int \left(\sum_{v=0}^{t-1} \prod_{l=0}^{v-1} P(s_{l+1} | s_l, a_l) \pi(a_l | H_l, l) (P(s_{v+1} | s_v, a_v) \right. \right. \\
&\quad \left. \left. - \bar{P}_W(s_{v+1} | s_v, a_v, v)) \pi(a_v | H_v, v) \cdot \prod_{l=v+1}^H \bar{P}_W(s_{l+1} | s_l, a_l, l) \pi(a_l | H_l, l) \right) r(s_t, a_t) d\tau \right| \\
&\leq R_{\max} \sum_{t=0}^{H-1} \gamma^t \underbrace{\sum_{v=0}^{t-1} \int \prod_{l=0}^{v-1} P(s_{l+1} | s_l, a_l) \pi(a_l | H_l, l) d \prod_{l=0}^{v-1} (s_l, a_l) \int \pi(a_v | H_v, v) da_v}_{=1} \\
&\quad \cdot \int |P(s_{v+1} | s_v, a_v) - \bar{P}_W(s_{v+1} | s_v, a_v, v)| ds_{v+1} \\
&\quad \cdot \underbrace{\int \prod_{l=v+1}^H \bar{P}_W(s_{l+1} | s_l, a_l, l) \pi(a_l | H_l, l) d \prod_{l=v+1}^t (s_l, a_l) ds_{H+1}}_{=1} \\
&\leq R_{\max} \sum_{t=0}^{H-1} \gamma^t \sum_{v=0}^{t-1} \max_{s \in \mathcal{S}, a \in \mathcal{A}} \|P(\cdot | s, a) - \bar{P}_W(\cdot | s, a, v)\|_1 \\
&\leq R_{\max} \sum_{t=0}^{+\infty} \gamma^t \sum_{v=0}^{t-1} \max_{s \in \mathcal{S}, a \in \mathcal{A}} \|P(\cdot | s, a) - \bar{P}_W(\cdot | s, a, v)\|_1.
\end{aligned}$$

We now observe that $\max_{s \in \mathcal{S}, a \in \mathcal{A}} \|P(\cdot | s, a) - \bar{P}_W(\cdot | s, a, v)\|_1 = 0$ when v is multiple of W . In the other case, having denoted $\bar{s} \in \operatorname{argmax}_{\bar{s} \in \mathcal{S}} P(\bar{s} | s, a)$:

$$\begin{aligned}
\max_{s \in \mathcal{S}, a \in \mathcal{A}} \|P(\cdot | s, a) - \bar{P}_W(\cdot | s, a, v)\|_1 &= \max_{s \in \mathcal{S}, a \in \mathcal{A}} \left\{ (1 - P(\bar{s} | s, a)) + \sum_{s' \neq \bar{s}} P(s' | s, a) \right\} \\
&= 2 \max_{s \in \mathcal{S}, a \in \mathcal{A}} (1 - P(\bar{s} | s, a)) \\
&= 2 \max_{s \in \mathcal{S}, a \in \mathcal{A}} \min_{s' \in \mathcal{S}} (1 - P(s' | s, a)) = 2D(P).
\end{aligned}$$

Thus, we have:

$$\begin{aligned}
\left| Q_{1,P}^\pi(s,a) - Q_{1,\bar{P}_W}^\pi(s,a) \right| &\leq 2R_{\max}D(P) \sum_{t=0}^{+\infty} \gamma^t \sum_{v=0}^{t-1} \mathbb{1}\{v \bmod W \neq 0\} \\
&= 2R_{\max}D(P) \sum_{v=0}^{+\infty} \mathbb{1}\{v \bmod W \neq 0\} \sum_{t=v+1}^{+\infty} \gamma^t \\
&= \frac{2R_{\max}D(P)}{1-\gamma} \sum_{v=0}^{+\infty} \mathbb{1}\{v \bmod W \neq 0\} \gamma^{v+1} \\
&= \frac{2R_{\max}D(P)}{1-\gamma} \sum_{d=0}^{+\infty} \sum_{m=0}^{W-1} \mathbb{1}\{(dW+m) \bmod W \neq 0\} \gamma^{dW+m+1} \\
&= \frac{2R_{\max}D(P)}{1-\gamma} \sum_{d=0}^{+\infty} \sum_{m=1}^{W-1} \gamma^{dW+m+1} \\
&= \frac{2\gamma R_{\max}D(P)}{1-\gamma} \sum_{d=0}^{+\infty} \gamma^{dW} \sum_{m=1}^{W-1} \gamma^m \\
&= 2R_{\max}D(P) \frac{\gamma(1-\gamma^{W-1})}{(1-\gamma)^2(1-\gamma^W)},
\end{aligned}$$

where we have swapped the summations in the second line, redefined index v with the pair of indexes (d, m) , and exploited multiple times the geometric sums. $\square$

Lemma 4.1 *Under Assumption 4.1, the variance of the GPOMDP estimator with batch size N is bounded for every $W \in \llbracket H \rrbracket$ as:*

$$\mathbb{V}\text{ar} [\widehat{\nabla}_\theta J(\theta; W)] \leq \frac{R_{\max}^2 G^2 (1-\gamma^H)}{N(1-\gamma)^2(1-\gamma^W)} \left(1 - \frac{H}{W} (\gamma^H - \gamma^{H+W}) - \gamma^H \right) =: \frac{V(W)}{N}.$$

Proof Starting from Lemma A.1, recalling Assumptions A.2 and A.3, we transpose the lemma in W -MDPs, applying Definition 2.1 and Assumption 4.1, thus $H \leftarrow \frac{H}{W}$, $\gamma \leftarrow \gamma^W$ and $R_{\max} \leftarrow R_{\max,W}$, and we get:

$$\mathbb{V}\text{ar} \|\nabla_\theta^2 J(\theta; W)\| \leq \frac{R_{\max}^2}{(1-\gamma)^2} \frac{G^2(1-\gamma^H)}{N(1-\gamma^W)} \left(1 - \frac{H}{W} (\gamma^H - \gamma^{H+W}) - \gamma^H \right).$$

$\square$

Lemma 4.2 *Under Assumptions 4.1 and 4.2, for every $W \in \llbracket H \rrbracket$, it holds that:*

$$\|\nabla_\theta^2 J(\theta; W)\| \leq \frac{R_{\max}(G^2 + F)}{(1-\gamma)(1-\gamma^W)} \left(1 - \frac{H}{W} (\gamma^H - \gamma^{H+W}) - \gamma^H \right) =: L(W).$$

Proof Recalling Assumption 4.1 and 4.2 and given Lemma A.2 applying Definition 2.1,

thus $H \leftarrow \frac{H}{W}$, $\gamma \leftarrow \gamma^W$ and $R_{\max} \leftarrow R_{\max, W} := \frac{1-\gamma^W}{1-\gamma} R_{\max}$, we get:

$$\|\nabla_{\theta}^2 J(\theta; W)\| \leq \frac{R_{\max}}{(1-\gamma)} \frac{(G^2 + F)}{1-\gamma^W} \left(1 - \frac{H}{W} (\gamma^H - \gamma^{H+W}) - \gamma^H\right).$$

□

Theorem 4.3 (Convergence to Stationary Point) . Under Assumptions 4.1 and 4.2, with a suitable step size to guarantee $\mathbb{E} [\|\nabla_{\theta} J(\theta_U; W)\|] \leq \epsilon$, where U is sampled uniformly he sample complexity of Algorithm 3 is at most:

$$NK \leq \tilde{O} \left(\frac{R_{\max}^3 (1-\gamma^H)}{\epsilon^2 (1-\gamma)^3 (1-\gamma^W)^2} \left(1 - \frac{H}{W} (\gamma^H - \gamma^{H+W}) - \gamma^H\right)^2 \right).$$

Proof We first apply Theorem A.3 under Assumptions A.1, A.2 and A.3.

$$NK = 2 (J^*(W) - J(\theta_0; W)) \left(\frac{L(W)}{\epsilon} + \frac{L(W)V(W)}{N\epsilon^2} \right)$$

We now apply Lemma 4.2 and 4.1 $V \leftarrow V(w)$ and $L \leftarrow L(w)$. Then, neglecting constants, we obtain:

$$\begin{aligned} NK = & \tilde{O} \left(\frac{R_{\max} N}{\epsilon (1-\gamma) (1-\gamma^W)} \left(1 - \frac{H}{W} (\gamma^H - \gamma^{H+W}) - \gamma^H\right) \right. \\ & \left. + \frac{R_{\max}^3 (1-\gamma^H)}{\epsilon^2 (1-\gamma)^3 (1-\gamma^W)^2} \left(1 - \frac{H}{W} (\gamma^H - \gamma^{H+W}) - \gamma^H\right)^2 \right) \end{aligned}$$

□

Theorem 4.4 (Global Convergence) Under Assumptions 4.1, 4.2 and 4.3, with a suitable step size to guarantee $\mathbb{E} [J^*(W) - J(\theta_K; W)] \leq \epsilon + \beta$, the sample complexity of Algorithm 3 is at most:

$$NK \leq \tilde{O} \left(\frac{R_{\max}^3 (1-\gamma^H)}{\epsilon^3 (1-\gamma)^3 (1-\gamma^W)^2} \left(1 - \frac{H}{W} (\gamma^H - \gamma^{H+W}) - \gamma^H\right)^2 \right).$$

Proof We first apply Theorem A.4, under Assumptions A.5, A.4 and A.6.

$$NK = \frac{16\alpha^4 LV}{\epsilon^3} \log \frac{\max\{0, J^*(W) - J(\theta_K; W) - \beta\}}{\epsilon}.$$

We redefine $V \leftarrow V(W)$ and $L \leftarrow L(W)$, applying Lemma 4.1 and 4.2. Then, neglecting logarithmic values and constants, we obtain:

$$NK = \tilde{O} \left(\frac{R_{\max}^3}{(1-\gamma)^3} \frac{(1-\gamma^H)}{\epsilon^3(1-\gamma^W)^2} \left(1 - \frac{H}{W} (\gamma^H - \gamma^{H+W}) - \gamma^H \right)^2 \right).$$

□

Technical Results from the Literature

In this part, we report assumptions, theorems, and lemmas from other works we used for the analysis.

Assumption A.1 (Papini et al., 2022), Definition 1. Let $\Pi_{\Theta} = \{\pi_{\theta} \mid \theta \in \Theta\}$ be a class of twice-differentiable parametric stochastic policies. There exists a constant $\xi_1 > 0$ such that for every state $s \in \mathcal{S}$ is defined as:

$$\mathbb{E}_{a \sim \pi_{\theta}(\cdot|s)} \left[\left\| \nabla_{\theta} \log \pi_{\theta}(a|s) \right\| \right] \leq \xi_1.$$

Assumption A.2 (Papini et al., 2022), Definition 1. Let $\Pi_{\Theta} = \{\pi_{\theta} \mid \theta \in \Theta\}$ be a class of twice-differentiable parametric stochastic policies. There exists a constant $\xi_2 > 0$ such that for every state $s \in \mathcal{S}$ is defined as:

$$\mathbb{E}_{a \sim \pi_{\theta}(\cdot|s)} \left[\left\| \nabla_{\theta} \log \pi_{\theta}(a|s) \right\|^2 \right] \leq \xi_2.$$

Assumption A.3 (Papini et al., 2022), Definition 1. Let $\Pi_{\Theta} = \{\pi_{\theta} \mid \theta \in \Theta\}$ be a class of twice-differentiable parametric stochastic policies. There exists a constant $\xi_3 > 0$ such that for every state $s \in \mathcal{S}$ is defined as:

$$\mathbb{E}_{a \sim \pi_{\theta}(\cdot|s)} \left[\left\| \nabla_{\theta}^2 \log \pi_{\theta}(a|s) \right\| \right] \leq \xi_3.$$

Assumption A.4 (Yuan et al., 2022), Assumption 3.1. There exists $L > 0$ such that, for all $\theta, \theta' \in \Theta$, it holds:

$$|J(\theta') - J(\theta) - \langle \nabla J(\theta), \theta' - \theta \rangle| \leq \frac{L}{2} \|\theta' - \theta\|^2.$$

Assumption A.5 (Montenegro et al., 2024), Assumption 6.1. There exist $\alpha > 0$ and $\beta \geq 0$ such that for every $\theta \in \Theta$ it holds that:

$$J^* - J(\theta) \leq \alpha \|\nabla_{\theta} J(\theta)\| + \beta.$$

Assumption A.6 (Montenegro et al., 2024), Assumption 6.3. For every $\theta \in \Theta$, the stochastic gradient $\widehat{\nabla}_{\theta} J(\theta)$ computed with batch size N has bounded variance, i.e.:

$$\mathbb{V}\text{ar} [\widehat{\nabla}_{\theta} J(\theta)] := \mathbb{E} [\|\widehat{\nabla}_{\theta} J(\theta) - \nabla_{\theta} J(\theta)\|^2] \leq \frac{V}{N}.$$

Lemma A.1 (Papini et al., 2022), Lemma 29. Given Assumptions A.2 and A.3 and a task horizon H , for every $\theta \in \Theta$, the variance of the GPOMDP estimator is upper-bounded as:

$$\mathbb{V}\text{ar} [\widehat{\nabla}_{\theta} J(\theta)] \leq \frac{\xi_2 R_{\max}^2 (1 - \gamma^H)}{N(1 - \gamma)^3} (1 - H(\gamma^H - \gamma^{H+1}) - \gamma^H) =: \frac{V}{N}.$$

Proof The result follows the one of (Papini et al., 2022) Lemma 29, stopping one step before on line D82. $\square$

Lemma A.2 (Yuan et al., 2022), Lemma 4.4. Under Assumptions A.2 and A.3, $J(\cdot)$ is L -smooth, namely $\|\nabla_{\theta}^2 J(\theta)\| \leq L$ with

$$L = \frac{R_{\max}(\xi_2 + \xi_3)}{(1 - \gamma)^2} (1 - H(\gamma^H - \gamma^{H+1}) - \gamma^H).$$

Proof The result follows from Lemma 4.4 of (Yuan et al., 2022) having limited the summations to $H - 1$

Theorem A.3 (Papini, 2020), Theorem 7.1. Under Assumptions A.4, A.6, running an algorithm with the update rule of Equation (3) with initial policy parameters θ_0 , batch size N , and a step size $\eta = \min \left\{ \frac{1}{L}, \frac{\epsilon N}{LV} \right\}$:

$$NK = 2(J(\theta^*) - J(\theta_0)) \left(\frac{L}{\epsilon} + \frac{LV}{\epsilon^2 N} \right)$$

guarantees $\mathbb{E} [\|\nabla_{\theta} J(\theta_U)\|] \leq \epsilon$ for U uniformly sampled from $\{0, 1, \dots, K - 1\}$

Theorem A.4 (Montenegro et al., 2024), Theorem F.1. Under Assumptions A.5, A.4 and A.6, running Algorithm 2 for $K > 0$ iterations with a batch size of $N > 0$ trajectories in each iteration with the constant learning rate η fulfilling:

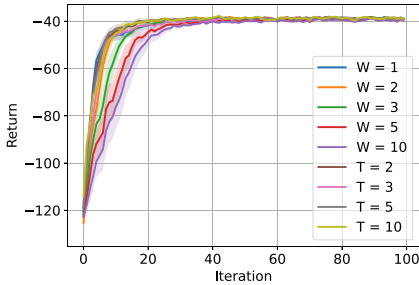
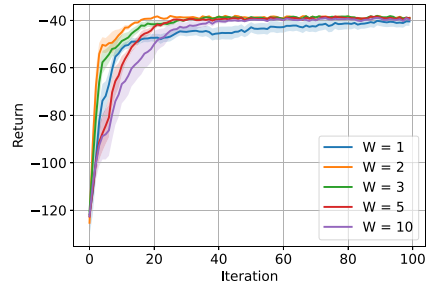
$$\eta \leq \min \left\{ \frac{1}{L}, \frac{1}{\mu \max\{0, J^* - J(\theta_0) - \beta\}}, \left(\frac{N}{LV\mu} \right)^{1/3} \right\},$$

Table 1 Parameters of the environments

Environment	Epoch	N	H	γ	$d_{\mathcal{S}}$	$d_{\mathcal{A}}$
Dam control	100	10	1825	0.999	7	1
Swimmer	500	10, 100	200, 500, 700	0.995, 0.999	8	2
Half-Cheetah	500	10, 100	200, 500, 700	0.995, 0.999	17	6

Table 2 Training parameters for the environments

Environment	σ	η	Type
Dam control	0.5	0.005	Adam
Swimmer	1	0.001	Constant
Half-Cheetah	0.1	0.001	Adam

(a) MetaStep (W) and action persistence (T), both over GPOMDP.

(b) MetaStep over REINFORCE.

Fig. 5 Learning curves for the Dam environment with $H = 1825$ (5 years), $\gamma = 0.999$ and $N = 10$ (10 runs, mean $\pm$ 95% C.I.)

where $\mu = \frac{1}{\alpha^2}$. Then, it holds that:

$$J^* - \mathbb{E}[J(\theta_K)] \leq \beta + \left(1 - \frac{1}{2} \sqrt{\frac{\mu \eta^3 LV}{N}}\right)^K \max\{0, J^* - J(\theta_0) - \beta\} + \sqrt{\frac{LV\eta}{\mu N}}.$$

In particular, for sufficiently small $\epsilon > 0$, setting $\eta = \frac{\epsilon^2 \mu N}{4LV}$, the following total number of samples is sufficient to ensure that $J^* - \mathbb{E}[J(\theta_K)] \leq \beta + \epsilon$:

$$NK \geq \frac{16LV}{\epsilon^3 \mu^2} \log \frac{\max\{0, J^* - J(\theta_0) - \beta\}}{\epsilon}.$$

Lemma A.5 The following function is non-increasing in the set $\{W \in \llbracket H \rrbracket : H \bmod W = 0\}$:

$$\frac{1}{1 - \gamma^W} \left(1 - \frac{H}{W} (\gamma^H - \gamma^{H+W}) - \gamma^H\right).$$

Proof We can rewrite the function as follows:

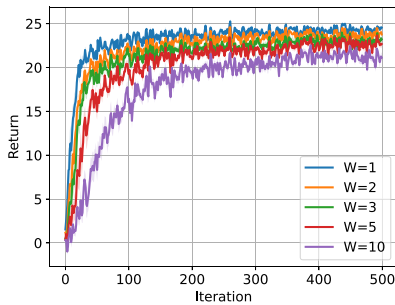
$$f(W) = \frac{1 - \gamma^H}{1 - \gamma^W} - \frac{H\gamma^H}{W}.$$

We reparametrize the function introducing $k \in \mathbb{N}$ such that $H = kW$ so that, with little abuse of notation:

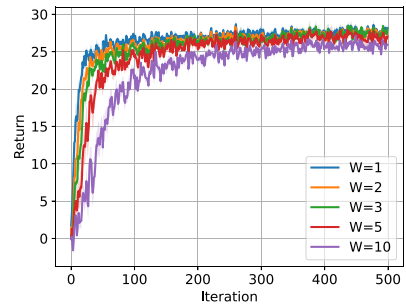
$$f(k) = \frac{1 - \gamma^H}{1 - \gamma^{H/k}} - k\gamma^H.$$

Thus, f is non-increasing in W if and only if f is non-decreasing in k . We compute the derivative w.r.t. k :

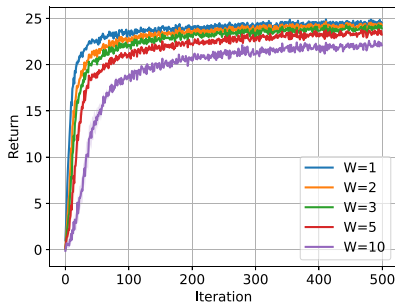
$$\begin{aligned} \frac{\partial f}{\partial k} &= -\frac{\gamma^{H/k} H \log \gamma (1 - \gamma^H)}{k^2 (1 - \gamma^{H/k})^2} - \gamma^H \\ &= \frac{\gamma^{H/k} \log \left(\frac{1}{\gamma^{H/k}} \right) (1 - \gamma^H)}{k (1 - \gamma^{H/k})^2} - \gamma^H \\ &\geq \frac{\gamma^{H/k} (1 - \gamma^H)}{k (1 - \gamma^{H/k})} - \gamma^H \\ &= \frac{\gamma^{H/k}}{k} (1 + \gamma^{H/k} + \dots + \gamma^{(k-1)H/k}) - \gamma^H \\ &\geq \frac{\gamma^{H/k}}{k} (\gamma^{(k-1)H/k} + \dots + \gamma^{(k-1)H/k}) - \gamma^H \\ &= \gamma^H - \gamma^H \\ &= 0, \end{aligned}$$



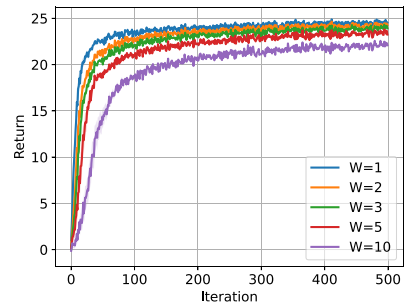
(a) $N = 10$, $\gamma = 0.995$.



(b) $N = 10$, $\gamma = 0.999$.

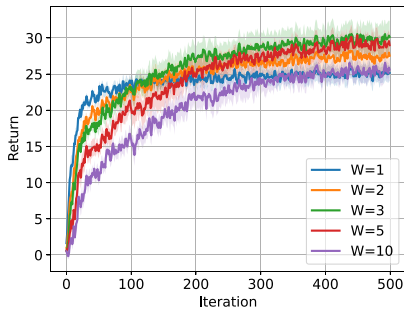
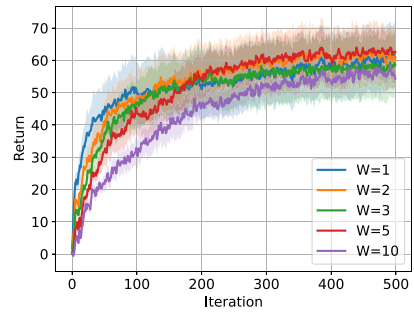
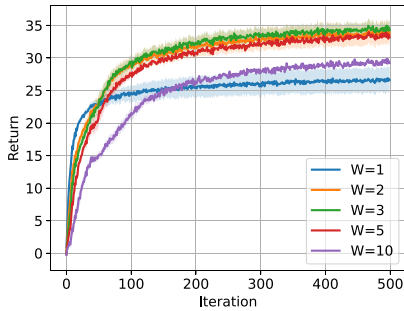
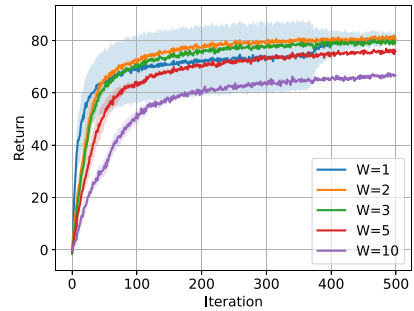


(c) $N = 100$, $\gamma = 0.995$.



(d) $N = 100$, $\gamma = 0.999$.

Fig. 6 Learning curves for swimmer with $H = 200$ (10 runs, mean $\pm$ 95% C.I.)

(a) $N = 10, \gamma = 0.995$.(b) $N = 10, \gamma = 0.999$.(c) $N = 100, \gamma = 0.995$.(d) $N = 100, \gamma = 0.999$.**Fig. 7** Learning curves for swimmer with $H = 500$ (10 runs, mean $\pm$ 95% C.I.)

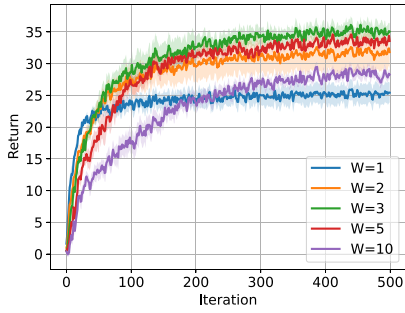
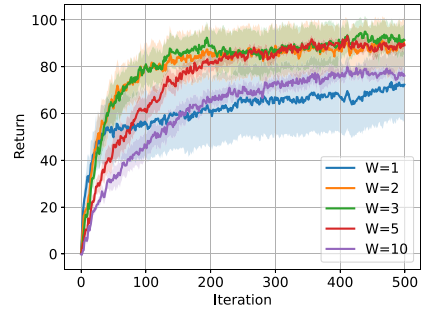
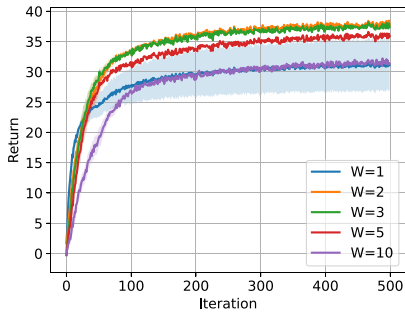
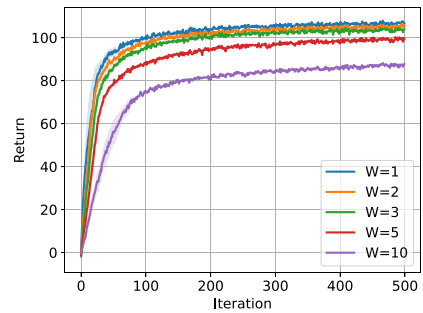
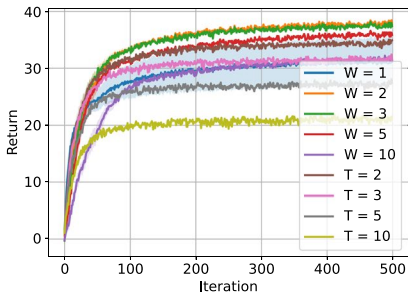
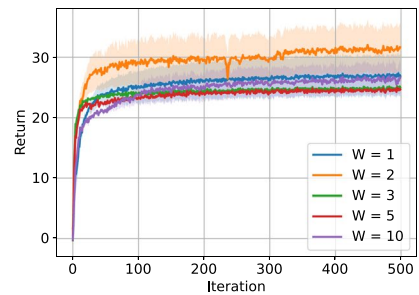
having exploited the inequality $\log(1/x) \geq 1 - x$ for $x \in [0, 1]$, applied with $x = \gamma^{H/k}$. $\square$

Detailed Experimental Settings and Additional Results

In this appendix, we discuss the experimental setting for the simulations provided in the main paper, and we provide further results.

Detailed Experimental Setting

Algorithm Settings To further reduce the variance in estimating the gradients, we adopt a version of GPOMDP with the optimal baseline (Peters & Schaal, 2008):

(a) $N = 10, \gamma = 0.995$.(b) $N = 10, \gamma = 0.999$.(c) $N = 100, \gamma = 0.995$.(d) $N = 100, \gamma = 0.999$.**Fig. 8** Learning curves for swimmer with $H = 700$ (10 runs, mean $\pm$ 95% C.I.)(a) **MetaStep** (W) and action persistence (T), both over GPOMDP.(b) **MetaStep** over REINFORCE.**Fig. 9** Learning curves for swimmer with $H = 700, \gamma = 0.995$ and $N = 100$ (10 runs, mean $\pm$ 95% C.I.)

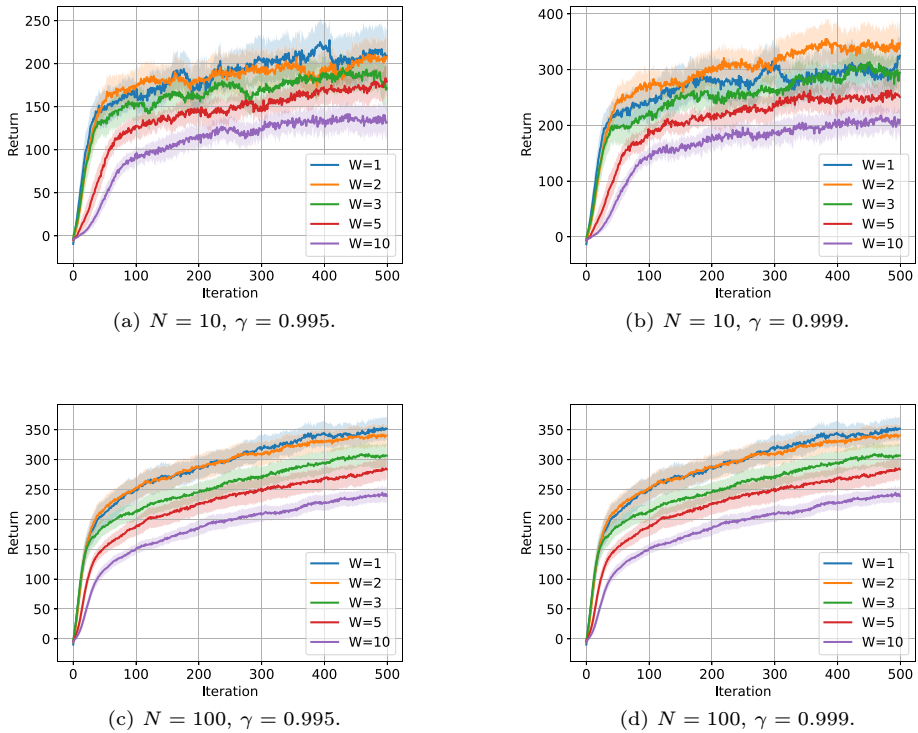


Fig. 10 Learning curves for Half-Cheetah with $H = 200$ (10 runs, mean $\pm$ 95% C.I.)

$$\hat{\nabla}_{\theta} J(\theta_t) = \mathbb{E} \left[\sum_{j=0}^{T-1} \sum_{t=0}^j \nabla_{\theta} \log \pi_{\theta} (a_t | s_t) (r_j - b_j) \right],$$

$$b_j = \frac{\mathbb{E} \left[\left(\sum_{t=0}^j \nabla_{\theta} \log \pi_{\theta} (a_t | s_t) \right)^2 r_j \right]}{\mathbb{E} \left[\left(\sum_{t=0}^j \nabla_{\theta} \log \pi_{\theta} (a_t | s_t) \right)^2 \right]},$$

The policy adopted for the experiments has been parametrized using a neural network with 2 hidden layers, composed of 50 and 25 parameters each.

We tested our algorithm on 3 environments of increasing complexity. *Dam control* (Castelletti et al., 2010); *Swimmer-v4*, *HalfCheetah-v4* from the MuJoCo suite (Todorov et al., 2012). Details on the environmental parameters are shown in Table 1. We adopted different learning rates η and exploration parameters, depending on the task we were observing, employing in some cases Adam (Diederik et al., 2015) to adaptively set the step size. The exact method adopted for each environment is shown in Table 2.

- **Computational Resources** All the experiments are run on a MacBook Pro. The machine is equipped as follows: CPU: Apple M2 Pro (10 cores, 3.4 GHz);
- RAM: 16 GB;

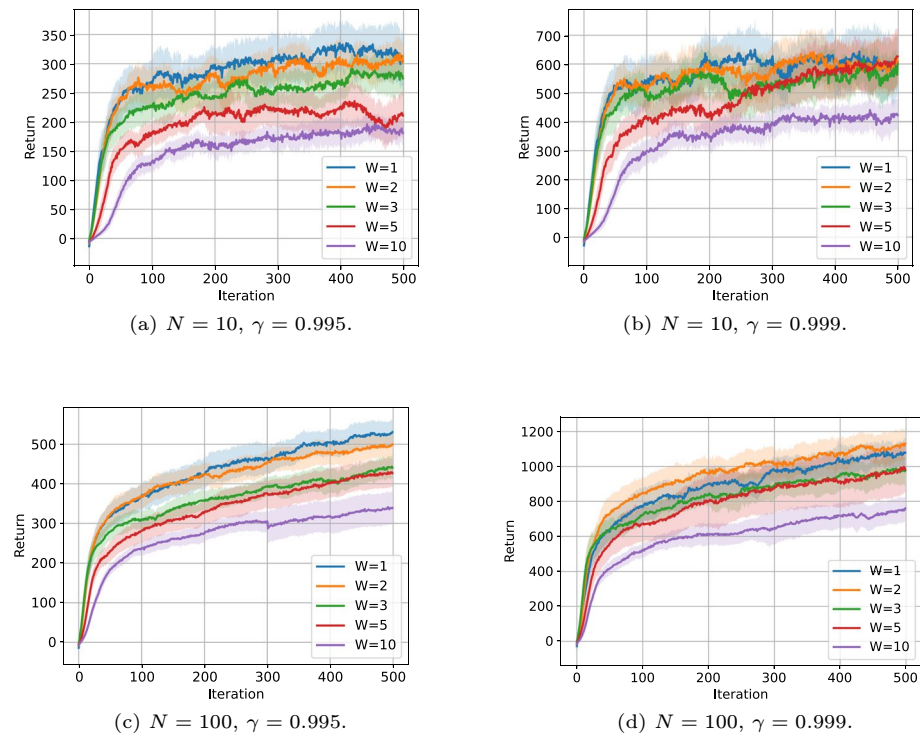


Fig. 11 Learning curves for Half-Cheetah with $H = 500$ (10 runs, mean $\pm$ 95% C.I.)

- GPU: 16-core GPU. All the performances are run over 8 CPU cores. Refer to Figs. 2b and 4b for the computational times for each environment for relevant experiments.

Additional Results

In the following, we complete the experimental campaign of the main paper with additional results.

Dam The additional results presented on the Dam environment propose a comparison between MetaStep and action persistence (Metelli et al., 2020) and an analysis over the integration of MetaStep on the REINFORCE algorithm (Williams, 1992). Starting from Fig. 5a we notice how on this task action persistence is particularly effective, matching or outperforming MetaStep. This is likely due to the specific dynamics of the Dam task, which favor stable, lower-frequency control. Moving to Fig. 5b we can appreciate how configuration with $W > 1$ manage to converge faster to the optimal solution w.r.t. the closed-loop solution. This illustrates the practical impact of Lemma 4.1, by mitigating the high variance inherent in the REINFORCE estimator, larger window sizes yield a more stable gradient and, consequently, faster optimization.

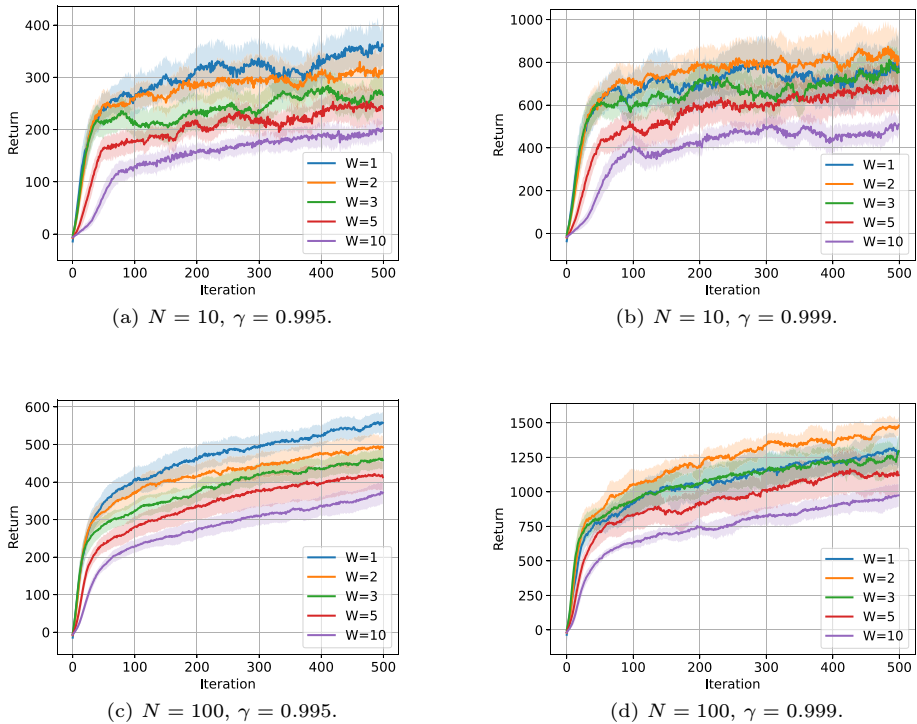


Fig. 12 Learning curves for Half-Cheetah with $H = 700$ (10 runs, mean $\pm$ 95% C.I.)

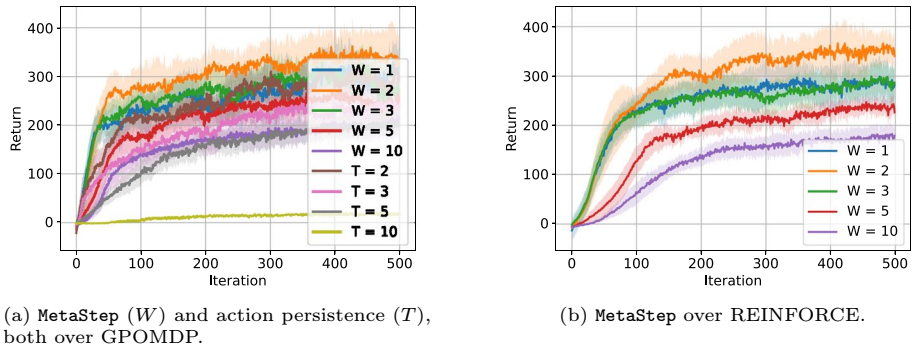


Fig. 13 Learning curves for Half-Cheetah with $H = 200$, $\gamma = 0.999$ and $N = 10$ (10 runs, mean $\pm$ 95% C.I.)

Swimmer The additional results presented on the Swimmer environment demonstrate the impact of the parameter W w.r.t. longer horizons, higher discount factors and batch size. As shown in Figs. 6, 7 and 8, for $H \in \{200, 500, 700\}$ independently from the batch size, we can appreciate how higher values of W bring an improvement in terms of sample complexity converging faster to the optimal behavior, especially with higher values of γ . This result supports our theoretical statements, giving a practical example of the benefits

of a reduced effective horizon. Moreover, we can appreciate how searching in the space of W policies induces a different manifold which is shown to help prevent plateau on the optimization phase (Figs. 7 and 8). In addition, we present a comparison between *MetaStep* and action persistence both applied over GPOMDP. In Fig. 9a we apply *MetaStep* over the REINFORCE algorithm. The results obtained are consistent with those in Figs. 7 and 8, confirming how learning in the space of W policies can prevent plateaus during the optimization process. Furthermore, since the REINFORCE suffers from high variance (Papini, 2020), larger window sizes (e.g., $W = 10$) prove particularly effective. This observation practically illustrates the variance reduction properties discussed in Lemma 4.1.

Half-Cheetah The additional results presented on the Half-Cheetah environment demonstrate the impact of the parameter W w.r.t. longer horizons, higher discount factors and batch size. We mostly appreciate how with a small batch size ($N = 10$) we are able to estimate less noisy gradient, thus providing more accurate policy updates. We see in Fig. 10, 11 and 12, how, regardless of the discount factor, values of $W > 1$ manage to obtain comparable performance to the closed-loop approach. Moreover, when the discount factor is higher ($\gamma = 0.999$), larger values of W manage also to converge faster w.r.t. $W = 1$, with the benefits of a reduced computational complexity (Fig. 4b). These results prove practically what is stated in Lemma 4.1, showing how the adoption of W -MDPs can effectively reduce the variance of the gradient estimator. Moreover, in Figs. 11 and 12, we can observe how for longer horizons and high discount factors, values of $W > 1$ not only converge faster w.r.t the closed-loop approach, but in certain setting also outperform the $W = 1$ solution, providing an empirical demonstration of the benefits of a reduced effective horizon. Additionally, we propose in Fig. 13a the learning curves of *MetaStep* and action persistence, both over GPOMDP. As we can see by the trends, *MetaStep* dominates with almost every configuration of W the performance of GPOMDP with action persistence. This validates our hypothesis: enabling the policy to vary actions within the window is significantly more effective than simply repeating the same action. Moreover, in Fig. 13b.

Author Contributions G.T. wrote the main manuscript, implemented the algorithm, wrote the theoretical contributions M.M. helped in writing the manuscript, helped in writing the theoretical contributions, reviewed the implementation A.M.M. helped in writing the manuscript, helped in writing the theoretical contributions All authors reviewed the manuscript

Funding Open access funding provided by Politecnico di Milano within the CRUI-CARE Agreement.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Bacon, P.-L., Harb, J., & Precup, D. (2017). The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, 31
- Barreto, A., Borsa, H., Comanici, G., Aygün, E., Hamel, P., Toyama, D., Mourad, S., Silver, D., & Precup, D. (2019) The option keyboard: Combining skills in reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 32
- Barto, A. G., & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13, 341–379.
- Baxter, J., & Bartlett, P. L. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15, 319–350.
- Bubeck, S., & Munos, R. (2010) Open loop optimistic planning. In *Annual conference on learning theory (COLT)*, pp. 477–489. Omnipress
- Castelletti, A., Galelli, S., Restelli, M., & Soncini-Sessa, R. (2010) Tree-based reinforcement learning for optimal water reservoir operation. *Water Resources Research*, 46(9)
- Chih-Han, Yu., Chuang, J., Gerkey, B., Gordon, G. J., & Andrew, Y. N. (2005). *Open-loop plans in multi-robot pomdps*. Technical report, Stanford CS Dept: Tech. Rep.
- Deisenroth, M. P., Neumann, G., & Peters, J. (2013). A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1–2), 1–142.
- Diederik, P., Kingma, J., & Ba. (2015). Adam: A method for stochastic optimization. International Conference on Learning Representations (ICLR)
- Dietterich, T. G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13, 227–303.
- Hansen, E., Barto, A., & Zilberstein, S. (1996). Reinforcement learning for mixed open-loop and closed-loop control. *Advances in Neural Information Processing Systems (NIPS)* (Vol. 9,
- Harb, J., Bacon, P.-L., Klissarov, M., & Precup, D. (2018) When waiting is not an option: Learning options with a deliberation cost. In *Proceedings of the AAAI conference on artificial intelligence*, 32
- Jiang, Z., & Liang, J. (2017). Cryptocurrency portfolio management with deep reinforcement learning. In *Intelligent systems conference (IntelliSys)*, pp. 905–913. IEEE
- Kolter, J. Z., Plagemann, C., Jackson, D.T., Ng, A. Y., & Thrun, S. (2010) A probabilistic approach to mixed open-loop and closed-loop control, with application to extreme autonomous driving. In *IEEE international conference on robotics and automation (ICRA)*, pp. 839–845. IEEE
- Konidaris, G., & Barto, A. G. (2009). Skill discovery in continuous reinforcement learning domains using skill chaining. *Advances in Neural Information Processing Systems (NIPS)* (Vol. 22,
- Lee, J., Lee, B.-J., & Kim, K.-E. (2020). Reinforcement learning for control with multiple frequencies. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 3254–3264.
- Likmeta, A., Metelli, A. M., Tirinzoni, A., Giol, R., Restelli, M., & Romano, D. (2020). Combining reinforcement learning with rule-based controllers for transparent and general decision-making in autonomous driving. *Robotics Autonomous Systems*, 131, Article 103568.
- Liu, Q., Li, L., Tang, Z., & Zhou, D. (2018) Breaking the curse of horizon: Infinite-horizon off-policy estimation. *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5361–5371
- Liu, Y., Bacon, P.-L., & Brunskill, E. (2020). Understanding the curse of horizon in off-policy evaluation via conditional importance sampling. In *Proceedings of the international conference on machine learning (ICML)*, vol. 119 of *Proceedings of machine learning research*, pp. 6184–6193. PMLR
- McGovern, A., & Barto, A.G. (2001) Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the international conference on machine learning (ICML)*, pp. 361–368
- Metelli, A.M., Mazzolini, F., Bisi, L., Sabbioni, L., & Restelli, M. (2020). Control frequency adaptation via action persistence in batch reinforcement learning. In *Proceedings of the international conference on machine learning (ICML)*, vol. 119 of *Proceedings of machine learning research*, pp. 6862–6873. PMLR
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.
- Montenegro, A., Mansutti, F., Mussi, M., Papini, M., & Metelli, A. M. (2025) Reusing trajectories in policy gradients enables fast convergence. *arXiv preprint arXiv:2506.06178*
- Montenegro, A., Mussi, M., Metelli, A. M., & Papini, M. (2024). Learning optimal deterministic policies with stochastic policy gradients. In *Proceedings of the international conference on machine learning (ICML)*
- Paczolay, G., Papini, M., Metelli, A. M., Harmati, I., & Restelli, M. (2024). Sample complexity of variance-reduced policy gradient: Weaker assumptions and lower bounds. *Machine Learning*, 113(9), 6475–6510.

- Papini, M. (2020). *Safe policy optimization*. Politecnico di Milano. PhD thesis.
- Papini, M., Binaghi, D., Canonaco, G., Pirota, M., & Restelli, M. (2018). Stochastic variance-reduced policy gradient. In *Proceedings of the international conference on machine learning (ICML)*, pp. 4026–4035. PMLR
- Papini, M., Pirota, M., & Restelli, M. (2022). Smoothing policies and safe policy gradients. *Machine Learning*, 111(11), 4081–4137.
- Pateria, S., Subagdja, B., Tan, A.-H., & Quek, C. (2022). Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Survey*, 54(5), 109.
- Peters, J., & Schaal, S. (2006). Policy gradient methods for robotics. In *IEEE international conference on intelligent robots and systems (IROS)*, pp. 2219–2225. IEEE
- Peters, J., & Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4), 682–697.
- Puterman, M. L. (2014). *Markov decision processes: Discrete stochastic dynamic programming*. Wiley Series in Probability and Statistics. John Wiley & Sons
- Sabbioni, L., Daire, L. A., Bisi, L., Metelli, A. M., & Restelli, M. (2023). Simultaneously updating all persistence values in reinforcement learning. In *AAAI conference on artificial intelligence*, pp. 9668–9676. AAAI Press
- Sham, M., & Kakade. (2001). A natural policy gradient. *Advances in Neural Information Processing Systems (NIPS)* (Vol. 14).
- Shivaram Kalyanakrishnan, S., Aravindan, V., Bagdawat, V., Bhatt, H., Goka, A., Gupta, K., Krishna, V., & Piratla. (2021). An analysis of frame-skipping in reinforcement learning [arXiv:2102.03718](https://arxiv.org/abs/2102.03718) arXiv preprint.
- Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, H., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419), 1140–1144.
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587), 484–489.
- Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M.A. (2014). Deterministic policy gradient algorithms. In *Proceedings of the international conference on machine learning (ICML)*, vol. 32 of *JMLR workshop and conference proceedings*, pp. 387–395. JMLR.org
- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. The MIT Press.
- Sutton, R. S., McAllester, D. A., Singh, S., & Mansour, Y. (1999a). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems (NIPS)*, pp. 1057–1063. The MIT Press
- Sutton, R. S., Precup, D., & Singh, S. (1999). Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2), 181–211.
- Tejas, D., Kulkarni, K., Narasimhan, A., Saeedi, J., & Tenenbaum. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in Neural Information Processing Systems (NIPS)* (Vol. 29).
- Todorov, E., Erez, T., & Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *IEEE international conference on intelligent robots and systems*, pp. 5026–5033. IEEE
- Tong, J., Shi, D., Liu, Y., & Fan, W. (2023). Gldap: Global dynamic action persistence adaptation for deep reinforcement learning. *ACM Transactions on Autonomous and Adaptive Systems*, 18(2), 1–18.
- Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8, 229–256.
- Xu, P., Gao, F., & Gu, Q. (2020). Sample efficient policy gradient methods with recursive variance reduction. *International Conference on Learning Representations*
- Yuan, H., Lian, X., Liu, J., & Zhou, Y. (2020). Stochastic recursive momentum for policy gradient methods. [arXiv:2003.04302](https://arxiv.org/abs/2003.04302) arXiv preprint.
- Yuan, R., Gower, R. M., & Lazaric, A. (2022). A general sample complexity analysis of vanilla policy gradient. In *Proceedings of the international conference on artificial intelligence and statistics (AISTATS)*, pp. 3332–3380. PMLR
- Yu, H., Xu, W., & Zhang, H. (2021). Taac: Temporally abstract actor-critic for continuous control. *Advances in Neural Information Processing Systems (NeurIPS)* (Vol. 34).