Trading-off Statistical and Computational Efficiency via W-step MDPs: A Policy Gradient Approach

Anonymous authors
Paper under double-blind review

Abstract

In reinforcement learning, algorithm performance 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 a desired accuracy—practitioners focus on reducing computational costs, such as training time and resource consumption. Closing this gap requires algorithms that balance both aspects effectively. In this paper, we introduce `MetaStep`, a meta-algorithm designed to enhance state-of-the-art RL algorithms by improving their computational performance 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-step 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-step MDP in terms of variance reduction and improved sample complexity. Finally, empirical evaluations confirm that `MetaStep` reduces computational costs while preserving—and, in certain scenarios, improving—sample efficiency.

1 Introduction

Reinforcement learning (RL, Sutton and Barto, 2018) has achieved remarkable success across a wide spectrum of complex real-world control tasks, such as robotic manipulation (Peters and Schaal, 2006), autonomous driving (Likmeta et al., 2020), game playing (Silver et al., 2018), and financial portfolio management (Jiang and 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 and 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 and 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 (both statistical and computational) 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 budget-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., 1999b; Barto and 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 and 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 stochastic environments where reactive behavior is critical. Differently, from the computational complexity perspective, approaches like *frame skipping* (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 fit ourselves between theorists and practitioners, tackling 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 Process* (W-MDP), where the agent plays sequences of W actions without observing intermediate states. We design `MetaStep`, a meta-algorithm designed to be paired with other RL approaches, shifting the agent-environment interaction in W-MDPs. Eventually, we adopt `MetaStep` on top of GPOMDP (Baxter and 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 Section 2, we present W-step Markov Decision Processes (W-MDPs). Then, we derive a novel result upper bounding the performance difference in terms of optimal value functions between planning in the original MDP and planning in the W-step MDP (Theorem 2.1), highlighting which features of the environment are responsible for the performance loss.
- In Section 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 Section 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 4.3 and 4.4).
- In Section 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 Section 6. The proofs of all the statements are provided in Appendix A. Additional experimental details are discussed in Appendix B.

2 Setting

In this section, we start by introducing the background notions on MDPs. Then, we present the W-MDP framework and the learning problem (Section 2.2). Finally, we propose a novel result on the optimal performance bias for W-MDPs (Section 2.3).

2.1 Preliminaries

Markov Decision Processes. A Markov Decision Process (MDP, Puterman, 1990) 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 state and action spaces, respectively, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ 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 $s \in \mathcal{S}$, $r : \mathcal{S} \times \mathcal{A} \to [-R_{\max}, R_{\max}]$ is the reward function, where $r(s, a)$ specifies the reward the agent gets by playing action a in state s, $\gamma \in [0, 1]$ 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.[1] A trajectory $\tau = (s_0, a_0, ..., s_{H-1}, a_{H-1})$ of length H is a sequence of H state-action pairs. The *discounted return* of a trajectory τ is $R(\tau) := \sum_{h=0}^{H-1} \gamma^t r(s_h, a_h)$.

Policies, Value Functions, and Optimality. The agent's behavior by a (possibly stochastic) Markovian policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ 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* $J(\pi) = \mathbb{E}_{\tau \sim d^\pi} [R(\tau)]$, where d^π is the distribution over trajectories induced by policy π. Moreover, we define the *state value function* (Sutton and Barto, 2018) as the expected discounted cumulative reward induced by policy π in state $s \in \mathcal{S}$: $V^\pi(s) = \mathbb{E}_\pi \left[\sum_{h=0}^{H-1} \gamma^h r(s_h, a_h) \middle| s_0 = s \right]$. 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^*.

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 (W-MDP)**.** *Let $\mathcal{M}$ be an MDP. For any $W \in \mathbb{N}_{\geq 1}$, the W-MDP is defined as $\mathcal{M}_W = (\mathcal{S}, \mathcal{A}^W, P_W, r_W, \overline{H}, \overline{\gamma}, \mu_0)$, where $\mathcal{A}^W = \{\mathbf{a} = (a_1, \dots, a_W) : a_i \in \mathcal{A}\ \forall i \in \{1, \dots, W\}\}$ is the space of W-step actions, $P_W : \mathcal{S} \times \mathcal{A}^W \to \Delta(\mathcal{S})$ 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:*

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

$r_W : \mathcal{S} \times \mathcal{A}^W \to \left[-\frac{1-\gamma^W}{1-\gamma} R_{max}, \frac{1-\gamma^W}{1-\gamma} R_{max} \right]$ 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:

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

$\overline{H} = \lceil H/W \rceil$ is the time horizon, and $\overline{\gamma} = \gamma^W$ 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-step MDP, the policy is defined as function $\pi^W : \mathcal{S} \to \Delta(\mathcal{A}^W)$. 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 presented in Sections 4 and 5, the reduced effective horizon provably brings improvements both in terms of sample and computational complexity.

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

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-step MDP starting from any state $s \in \mathcal{S}$:

$$V_W^{\pi^W}(s) = \mathbb{E}_{\pi^W}\left[\sum_{h=0}^{\overline{H}-1} \overline{\gamma}^h r_W(s_{Wh}, (a_{Wh}, \dots, a_{Wh+W-1}))\middle| s_0 = s\right]. \tag{1}$$

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

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

where:

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

Some comments are in order. As expected, for $W = 1$, the performance difference between the W-step 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, than $D(P) = 0$ and, consequently, no suboptimality is introduced, i.e., $V^*(s) = V_W^*(s)$. This is compliant with the known fact that open-loop control is optimal for deterministic environments. Instead, when P is "maximally stochastic", i.e., $P(s'|s,a) = 1/S$ for every $s' \in \mathcal{S}$, we have that $D(P) = 1 - 1/S$ taking its maximum value.

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 the 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, we redefine the communication protocol between the agent and the environment, as 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 an 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

Algorithm 1: `MetaStep` — Interaction Protocol.

Input : Horizon H, parameter W, policy π^W
for $h \in [\![\lceil H/W \rceil]\!]$ **do**
  Observe state s
  Sample action $\boldsymbol{a} = (a_1, \dots, a_W) \sim \pi^W(\cdot|s)$
  for $w \in [\![W]\!]$ **do**
    Play action a_w and collect reward r_w
  end
end

horizon of the task and lightening the computational burden during the training phase, as we will see in Section 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 detailed in Theorem 2.1.

4 `MetaStep` on `GPOMDP`

In this section, we investigate the performance of `MetaStep` when applied to `GPOMDP` (Baxter and 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 $\Pi_\Theta = \{\pi_\theta : \mathcal{S} \to \Delta(\mathcal{A}) : \theta \in \Theta \subseteq \mathbb{R}^d\}$, where Θ is the parameter space. At each iteration $k \in [\![K]\!]$, 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:

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

where $\eta > 0$ is the step size and $\widehat{\nabla}_{\theta_k} J(\theta_k)$ 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 and 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 $i \in [\![N]\!]$ and $g(\tau_i|\theta_k) = \nabla_\theta \log p(\tau_i|\theta_k) R(\tau_i)$, being $p(\cdot|\theta)$ the density function over trajetories induced by π_θ and $R(\tau)$ the trajectory return. In this work, g will refer to the `GPOMDP` estimator, which is preferred over `REINFORCE` due to its smaller variance and is given by:

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

Algorithm 2 reports the pseudocode of the `GPOMDP` in its base implementation.

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 π_θ^W. 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 $(s_h, \mathbf{a}_W, r_W)$ 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.

Algorithm 2: `GPOMDP`.

Input : Number of iterations K, batch size N, initial parameter vector θ_0, environment $\mathcal{M}$, horizon H, discount factor γ, learning rate η

Initialize $\theta \leftarrow \theta_0$
for $i \in [\![K]\!]$ **do**
  Set the stochastic policy parameters: π_θ
  for $l \in [\![N]\!]$ **do**
    Initialize trajectory τ_l as an empty tuple
    for $h \in [\![H]\!]$ **do**
      Observe state s_h
      Play action $a_h \sim \pi_\theta(\cdot|s_h)$ and observe reward r_h
      Add to τ_l the tuple (s_h, a_h, r_h)
    end
  end
  Estimate the gradient according to Equation (3)
  Update the policy parameter vector: $\theta \leftarrow \theta + \eta \widehat{\nabla}_\theta J(\theta)$
end
Return θ

Computational Complexity. From a computational perspective, the proposed approach offers two key advantages. First, recalling the `GPOMDP` estimator function (Equation 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 Section 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., $J^*(W) = \sup_{\theta \in \Theta_W} J(\theta; W)$, and with $J(\theta_k; W)$ the performance obtained by the policy parametrization θ_k using parameter W.

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

Assumption 4.1 (Bounded gradient)**.** *There exist $G > 0$ such that $\forall W \in [\![H]\!]$ the expected norm squared of the gradient of $log\ \pi_\theta^W(\cdot|s)$ satisfies:*

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

Assumption 4.2 (Bounded Hessian)**.** *There exist $F > 0$ such that $\forall W \in [\![H]\!]$ the expected spectral norm of the Hessian of $log\ \pi_\theta^W(\cdot|s)$ satisfies:*

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

Intuitively, these assumptions ensure that the effect on the environment dynamics and reward (and on their gradient), after playing an action, is controllable. 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-step MDPs.

Algorithm 3: `MetaStep` on `GPOMDP`.

Input : Number of iterations K, batch size N, initial parameter vector θ_0, parameter W, environment $\mathcal{M}_W$, horizon H/W, original discount factor γ, discount factor γ^W, learning rate η

Initialize $\theta \leftarrow \theta_0$
for $i \in [\![K]\!]$ **do**
  Set the stochastic policy parameters: π_θ
  for $l \in [\![N]\!]$ **do**
    Initialize trajectory τ_l as an empty tuple
    for $h \in [\![H/W]\!]$ **do**
      Observe state s_h
      Sample action sequence $\mathbf{a}_W \sim \pi_\theta^W(\cdot|s_h)$
      for $w \in [\![W]\!]$ **do**
        Play action a_w and observe reward r_w
        Update the W-reward $r_W \leftarrow r_W + \gamma^{w-1} r(s, a_w)$
      end
      Add to τ_l the tuple $(s_h, \mathbf{a}_W, r_W)$
    end
  end
  Compute the gradient estimator:

$$\widehat{\nabla}_\theta J(\theta) \leftarrow \frac{1}{N} \sum_{i=1}^{N} \sum_{h=0}^{\frac{H}{W}-1} \left(\sum_{u=0}^{h} \nabla_\theta \log \pi_\theta(\mathbf{a}_{\tau_i,u}|s_{\tau_i,u}) \right) \gamma^{Wh} r(s_{\tau_i,h}, \mathbf{a}_{\tau_i,h})$$

  Update the policy parameter vector: $\theta \leftarrow \theta + \eta \widehat{\nabla}_\theta J(\theta)$
end
Return θ

Lemma 4.1. *Under Assumption 4.1, the variance of the* `GPOMDP` *estimator with batch size N is bounded for every $W \in [\![H]\!]$ as:*

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

Lemma 4.2. *Under Assumptions 4.1 and 4.2, $\forall W \in [\![H]\!]$, it holds that:*

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

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 refer to Appendix A for the proof of Lemmas 4.1 and 4.2.

We highlight that the bound described in Lemma 4.1 does not grow as W, demonstrating how the usage of W-MDPs helps reduce the effective horizon of the task and consequently also the variance of the gradient estimates. In Figure 1, we show the *bias variance* trade-off as a function of W. We notice how the bias on the performance (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 a U-shaped behavior, highlighting the trade-off between estimation accuracy and stability when we are in noisy environments. This highlights the importance of selecting an appropriate W, as for stochastic environments, it trades off bias and variance.

Below, we state the convergence on a stationary point.

Theorem 4.3. *Consider an algorithm using the update rule of Eq. (2). Under Assumptions 4.1 and 4.2, with a suitable step size to guarantee $\mathbb{E}\left[\|\nabla J(\theta; W)\|\right] \leq \epsilon$, the sample complexity is at most:*

$$NK \geq \Omega\left(\frac{R_{\max}^3 (1-\gamma^H)}{\epsilon^2 (1-\gamma)^3 (1-\gamma^W)^2} \left[1 - \frac{H}{W} \left(\gamma^H - \gamma^{H+W} \right) - \gamma^H \right]^2 \right).$$

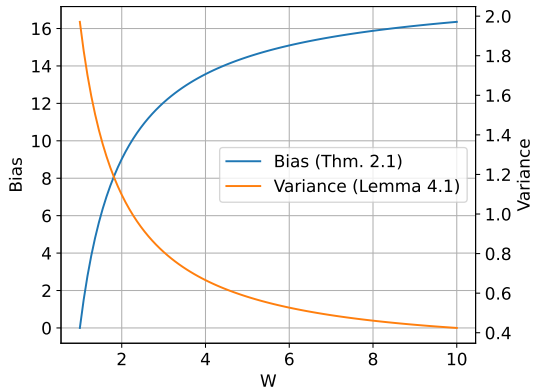


Figure 1: Trend of the variance of the gradient estimator as the parameter W changes.

The first term dominates, yielding the well-known $\Omega(\epsilon^{-2})$ rate. To provide also a global convergence result we enforce the following assumption.

Assumption 4.3 (Weak gradient domination for $J(W)$)**.** *There exists $\alpha > 0$ and $\beta > 0$ such that for every $\theta \in \Theta_W$, and for every $W \in [\![H]\!]$, it holds that:*

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

Assumption 4.3 is a standard for global convergence analysis of stochastic optimization (Yuan et al., 2022). The weak gradient domination (WGD) assumption admits that the performance function $J(\theta_k; W)$ has local maxima as long as its performance is β-close to the globally optimal one. Given Assumption 4.3, we can now state the global convergence result.

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

$$NK \geq \widetilde{\Omega}\left(\frac{R_{\max}^3}{(1-\gamma)^3\epsilon^3}\frac{(1-\gamma^H)}{(1-\gamma^W)^2}\left[1 - \frac{H}{W}\left(\gamma^H - \gamma^{H+W}\right) - \gamma^H\right]^2\right).$$

This result establishes a convergence of order $\widetilde{\Omega}(\epsilon^{-3})$ to the last-iterate global optimum convergence of stochastic policies in accordance with state-of-the-art PG analysis (Yuan et al., 2022)[2]. We notice in the sample complexity how the usage of W-step 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. The proofs of Theorems 4.3 and 4.4 are presented in Appendix A.

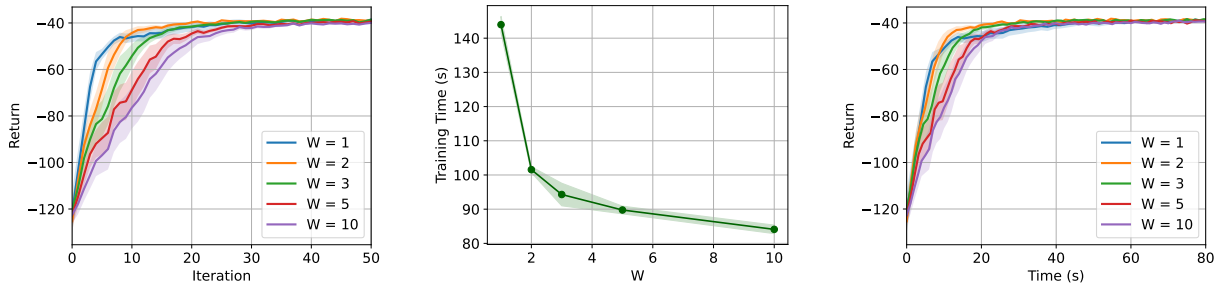
5 Experimental Validation

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 complex domains to analyze its performance under different settings. For each environment, we test our approach with several values of $W \in \{1, 2, 3, 5, 10\}$, where $W = 1$ represents 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 $Wd_{\mathcal{A}}$ elements. The standard deviation is fixed and tuned for each task.[3]

For each task, we will present three plots. The first (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 second plot (denoted with (b)) reports the training time as a function of the

[2]The $\widetilde{\Omega}(\cdot)$ notation hides logarithmic factors.

[3]Appendix B provides additional experimental details and results. The code to reproduce the experiments is available in the supplementary material.



(a) Performance over iterations. (b) Running time for $K = 100$. (c) Performance over time.

Figure 2: Learning curves and training time for the dam control task.

window size W. Lastly, the third plot (denoted with (c)) displays the learning curves relative to training time, offering insight into the computational complexity.

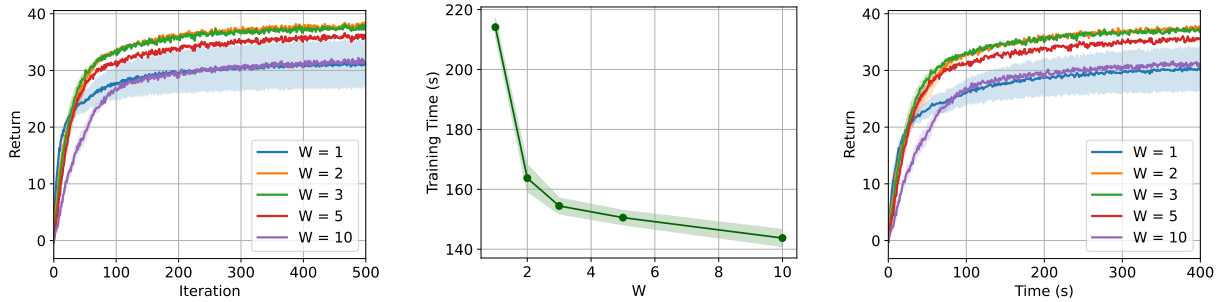
Dam Control. We first consider the water resource management task described in (Castelletti et al., 2010). The agent learns a water release policy balancing external demand D (e.g., for a town) while preventing floods by keeping the water level below a threshold F. The dam experiences a stochastic daily inflow, 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 ($d_{\mathcal{A}} = 1$), while the state space has dimension $d_{\mathcal{S}} = 7$. The discount factor is $\gamma = 0.999$. We train our agent for $H = 1825$ steps (equivalent to 5 years of data) with batch size $N = 10$.

From the learning curves in Figure 2a, we observe that all approaches converge to the optimal value, regardless of the choice of W. Notably, with $W = 2$, convergence occurs slightly faster than with the single-step approach, confirming the sample efficiency gains of a reduced effective horizon, coherently with Theorem 4.4. Furthermore, Figure 2b shows how larger values of W imply reduced training time, highlighting the computational advantages of our method. This leads us to the third view, presented in Figure 2c, where the impact of wider windows over the training time is clearer, and the improvement with $W \in \{2, 3\}$ is more evident w.r.t. the base algorithm.

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 space dimension $d_{\mathcal{A}} = 2$, while the state space dimension $d_{\mathcal{S}} = 8$. We train our agent for $H = 700$ steps, using a batch size $N = 100$ and a discount factor $\gamma = 0.995$.

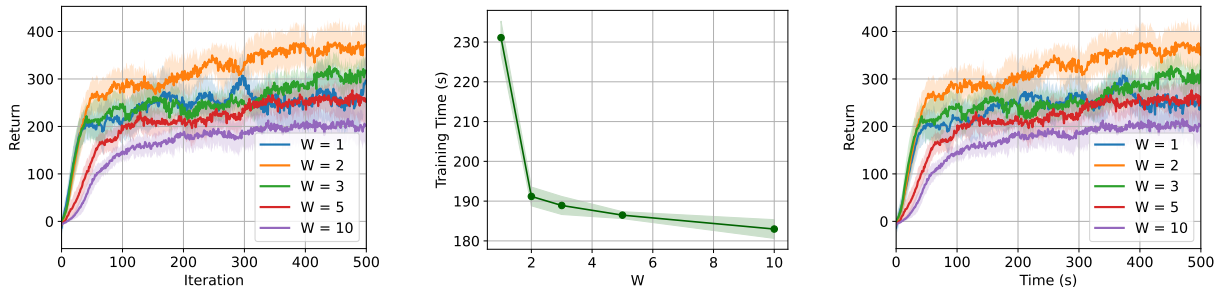
Figure 3a shows the learning curve over samples. We observe that larger values of W lead to faster convergence and higher final performance. This phenomenon is due to the search in the space of Π^W policies, which induces a different manifold that might avoid plateaus during optimization. In Figure 3b, we appreciate the analysis of the computation time needed for the training as a function of W, highlighting its descending trend w.r.t. higher values of W. Finally, in Figure 3c, we present the performance over the training time required, highlighting the benefits of using a higher value of W.

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 space has dimension $d_{\mathcal{A}} = 6$, while the state space has dimension $d_{\mathcal{S}} = 17$. 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.



(a) Performance over iterations. (b) Running time for $K = 500$. (c) Performance over time.

Figure 3: Learning curves and training time for Swimmer.



(a) Performance over iterations. (b) Running time for $K = 500$. (c) Performance over time.

Figure 4: Learning curves and training time for Half-Cheetah.

In Figure 4a, we empirically validate Lemma 4.1, as with a smaller batch size, 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 (as shown in Theorem 4.4). From the computational perspective, Figure 4b, also in this environment, demonstrates a clear reduction in training time as W increases. Finally, in Figure 4c, we observe how, looking at the computational complexity, $W = 2$ presents a significantly faster convergence.

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.

Policy Gradient Methods. Policy gradient (PG) methods constitute a fundamental class of reinforcement learning (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 and Bartlett, 2001) and actor-critic approaches (Sutton et al., 1999a), addressed the high variance of gradient estimates by incorporating reward-to-go and baseline subtraction. In general, the problem of variance reduction in policy gradient estimation has attracted the attention of many researchers (Kakade, 2001; Schulman et al., 2015; Papini et al., 2018). Despite these advancements, computational efficiency remains challenging, as policy gradient methods typically require frequent updates and high sample complexity. 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 and 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 (Yu et al., 2005). However, 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. (1999b) and macro-actions McGovern and Barto (2001) provide a hierarchical framework in which temporally extended actions facilitate long-term reasoning and efficient exploration. 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. 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.

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 the solution from both the computational and the statistical perspectives. Finally, we proposed an extensive experimental evaluation to empirically validate our results. Future works may investigate the applicability of `MetaStep` on other policy search methods and extend the analysis to value-based approaches.

References

P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13:341–379, 2003.

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

S. Bubeck and R. Munos. Open loop optimistic planning. In *COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27-29, 2010*, pages 477–489. Omnipress, 2010.

A. Castelletti, S. Galelli, M. Restelli, and R. Soncini-Sessa. Tree-based reinforcement learning for optimal water reservoir operation. *Water Resources Research*, 46(9), 2010.

M. P. Deisenroth, G. Neumann, and J. Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2013.

E. Hansen, A. Barto, and S. Zilberstein. Reinforcement learning for mixed open-loop and closed-loop control. *Advances in Neural Information Processing Systems*, 9, 1996.

J. Harb, P.-L. Bacon, M. Klissarov, and D. Precup. When waiting is not an option: Learning options with a deliberation cost. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Z. Jiang and J. Liang. Cryptocurrency portfolio management with deep reinforcement learning. In *Intelligent systems conference (IntelliSys)*, pages 905–913. IEEE, 2017.

S. M. Kakade. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

S. Kalyanakrishnan, S. Aravindan, V. Bagdawat, V. Bhatt, H. Goka, A. Gupta, K. Krishna, and V. Piratla. An analysis of frame-skipping in reinforcement learning. *arXiv preprint arXiv:2102.03718*, 2021.

D. P. Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

J. Z. Kolter, C. Plagemann, D. T. Jackson, A. Y. Ng, and S. Thrun. A probabilistic approach to mixed open-loop and closed-loop control, with application to extreme autonomous driving. In *2010 IEEE International Conference on Robotics and Automation*, pages 839–845. IEEE, 2010.

J. Lee, B.-J. Lee, and K.-E. Kim. Reinforcement learning for control with multiple frequencies. *Advances in Neural Information Processing Systems*, 33:3254–3264, 2020.

A. Likmeta, A. M. Metelli, A. Tirinzoni, R. Giol, M. Restelli, and D. Romano. Combining reinforcement learning with rule-based controllers for transparent and general decision-making in autonomous driving. *Robotics Auton. Syst.*, 131:103568, 2020.

Q. Liu, L. Li, Z. Tang, and D. Zhou. Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5361–5371, 2018.

Y. Liu, P. Bacon, and E. Brunskill. Understanding the curse of horizon in off-policy evaluation via conditional importance sampling. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 6184–6193. PMLR, 2020.

A. McGovern and A. G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 361–368, 2001.

A. M. Metelli, F. Mazzolini, L. Bisi, L. Sabbioni, and M. Restelli. Control frequency adaptation via action persistence in batch reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 6862–6873. PMLR, 2020.

A. Montenegro, M. Mussi, A. M. Metelli, and M. Papini. Learning optimal deterministic policies with stochastic policy gradients. In *International Conference on Machine Learning (ICML)*, 2024.

M. Papini. Safe policy optimization. 2020.

M. Papini, D. Binaghi, G. Canonaco, M. Pirotta, and M. Restelli. Stochastic variance-reduced policy gradient. In *International conference on machine learning*, pages 4026–4035. PMLR, 2018.

S. Pateria, B. Subagdja, A. Tan, and C. Quek. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Survey*, 54(5):109:1–109:35, 2022.

J. Peters and S. Schaal. Policy gradient methods for robotics. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2219–2225. IEEE, 2006.

J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4): 682–697, 2008.

M. L. Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2: 331–434, 1990.

L. Sabbioni, L. A. Daire, L. Bisi, A. M. Metelli, and M. Restelli. Simultaneously updating all persistence values in reinforcement learning. In *AAAI Conference on Artificial Intelligence*, pages 9668–9676. AAAI Press, 2023.

J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 387–395. JMLR.org, 2014.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, H. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

R. S. Sutton and A. G. Barto. *Reinforcement learning: an introduction*. The MIT Press, Cambridge, 2018.

R. S. Sutton, D. A. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1057–1063. The MIT Press, 1999a.

R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999b.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

J. Tong, D. Shi, Y. Liu, and W. Fan. Gldap: Global dynamic action persistence adaptation for deep reinforcement learning. *ACM Transactions on Autonomous and Adaptive Systems*, 18(2):1–18, 2023.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

C. Yu, J. Chuang, B. Gerkey, G. Gordon, and A. Ng. Open-loop plans in multi-robot pomdps. *Technical report, Stanford CS Dept, Tech. Rep.*, 2005.

H. Yu, W. Xu, and H. Zhang. Taac: Temporally abstract actor-critic for continuous control. *Advances in Neural Information Processing Systems*, 34:29021–29033, 2021.

R. Yuan, R. M. Gower, and A. Lazaric. A general sample complexity analysis of vanilla policy gradient. In *International Conference on Artificial Intelligence and Statistics*, pages 3332–3380. PMLR, 2022.

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-step MDP. Then, for every $s \in \mathcal{S}$, it holds that:*

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

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:

$$Q^\pi(s,a) = \mathbb{E}_\pi \left[\sum_{h=0}^{H-1} \gamma^h r(s_h, a_h) \middle| s_0 = s, a_0 = a \right]. \tag{4}$$

For a W-MDP, the definition of $Q^{\pi^W}_W$ follows directly.

We start the proof by observing that, for every MDP and corresponding W-MDP, we have $Q^*_1(s,a) \geq Q^*_W(s,a)$.

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

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

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

$$Q^*_{1,\overline{P}_W}(s,a) = Q^*_{W,\overline{P}_W}(s,a),$$

as where we are not able to take action, we have deterministic transitions. This is equal to say that in the MDP with $\overline{P}_W$ we have:

$$\max_{\pi:\mathcal{S}\to\mathcal{A}} \mathbb{E}\left[\sum_h \gamma^h r_h\right] = \max_{\pi^W:\mathcal{S}\to\mathcal{A}^W} \mathbb{E}\left[\sum_h \gamma^h r_h\right].$$

Now the goal is to try to bound the difference between $Q^*_{1,P}$ and $Q^*_{W,P}$. This is equal to:

$$\begin{aligned} Q^*_{1,P} - Q^*_{W,P} &= Q^*_{1,P} - Q^*_{W,P} \pm Q^*_{W,\overline{P}_W} \\ &\leq Q^*_{1,P} - Q^*_{1,\overline{P}_W} + Q^*_{W,\overline{P}_W} - Q^*_{W,P} \\ &\leq Q^{\pi^{1*}}_{1,P} - Q^{\pi^{1*}}_{1,\overline{P}_W} + Q^{\overline{\pi}^{W*}}_{W,\overline{P}_W} - Q^{\overline{\pi}^{W*}}_{W,P} \end{aligned}$$

Since we are to bound the difference between two Q-functions of the same policy under different transition models, we proceed analyzing the first term only, as the second one is analogous:

$$Q^{\pi^*,P}_1 - Q^{\pi^*,\overline{P}_W}_1 \tag{5}$$

$$= \mathbb{E}_{\bar{P}}\left[\sum_{t=0}^{\infty} \bar{\bar{\gamma}}^t r(s_t,a_t)\right] - \mathbb{E}_P\left[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t)\right] \tag{6}$$

$$= \int_\tau \left(\prod_{l=0}^{\infty} P(s_{l+1}|s_l,\underline{a}_l)\pi^{1*}(a_l|s_l) - \prod_{l=0}^{\infty} \overline{P}_W(s_{l+1}|s_l,a_l,t)\pi^{1*}(a_l|s_l)\right) \sum_{t=0}^{\infty} \gamma^t r(s_t,a_t) d\tau \tag{7}$$

$$= \sum_{t=0}^{\infty} \gamma^t \int_\tau \left(\prod_{l=0}^{t} P(s_{l+1}|s_l,a_l)\pi^{1*}(a_l|s_l) - \prod_{l=0}^{t} \overline{P}_W(s_{l+1}|s_l,a_l)\pi^{1*}(a_l|s_l)\right) r(s_t,a_t) d\tau \tag{8}$$

$$= \sum_{t=0}^{\infty} \gamma^t \int_{\tau} \sum_{v=0}^{t-1} \prod_{l=0}^{v-1} P(s_{l+1}|s_l, a_l)\pi^{1*}(a_l|s_l)(P(s_{v+1}|s_v, a_v, v) - \overline{P}_W(s_{v+1}|s_v, \underline{a}_v))\pi^{1*}(a_v|s_v) \tag{9}$$

$$\cdot \prod_{l=v+1}^{H} \overline{P}_W(s'_{l+1}|s_l, a_l)\pi^{1*}(a_l|s_l)r(s_t, a_t)d\tau \tag{10}$$

$$= R_{\max} \sum_{t=0}^{\infty} \gamma^t \sum_{v=0}^{t-1} D_v = (*) \tag{11}$$

where:

$$D_v \coloneqq \max_{s,a} \left\| P(\cdot|s, a, v) - \overline{P}_W(\cdot|s, a) \right\|_1 \tag{12}$$

$$= \max_{s,a} \int_{s'} (\bar{P}(\cdot|s, a, v) - \overline{P}_W(\cdot|s, a))ds' \tag{13}$$

$$\leq D\mathbb{1}\{v \mod W \neq 0\}. \tag{14}$$

Eventually, we have:

$$(*) = DR_{\max} \sum_{t=0}^{\infty} \gamma^t \sum_{v=0}^{t-1} \mathbf{1}\{v \bmod W \neq 0\} \tag{15}$$

$$= DR_{\max} \left(\frac{\gamma}{(1-\gamma)^2} - \frac{\gamma^W W}{(1-\gamma^W)^2} \right). \tag{16}$$

□

Lemma 4.2. *Under Assumptions 4.1 and 4.2, $\forall W \in [\![H]\!]$, it holds that:*

$$\left\| \nabla^2 J(\theta) \right\| \leq \frac{R_{max}}{(1-\gamma)} \frac{(G^2+F)}{1-\gamma^W} \left[1 - \frac{H}{W} \left(\gamma^H - \gamma^{H+W} \right) - \gamma^H \right] \coloneqq L.$$

Proof. Recalling Assumption 4.1 and 4.2 we resume (Yuan et al., 2022, Lemma 4.4, Equation 69,72) applying it on W-step MDPs we get,

$$\left\| \nabla^2 J(\theta) \right\| \leq (G^2+F)R_{\max} \sum_{t=0}^{\infty} \gamma^t (t+1) \ \leq \ \frac{R_{\max}}{(1-\gamma)^2} (G^2+F) \left[1 - H(\gamma^H - \gamma^{H+1}) - \gamma^H \right]$$

Eventually, applying Definition 2.1, thus $H \leftarrow \frac{H}{W}$, $\gamma \leftarrow \gamma^W$ and $R_{\max} \leftarrow R_{\max,W}$, we get,

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

□

Lemma 4.1. *Under Assumption 4.1, the variance of the* `GPOMDP` *estimator with batch size N is bounded for every $W \in [\![H]\!]$ as:*

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

Proof. Starting from (Papini et al., 2018, Lemma 29), recalling the Assumption 4.1, and differentiating the notation using $G^2 \leftarrow \xi_2$ we transpose the lemma in W-step MDPs, applying Definition 2.1, thus $H \leftarrow \frac{H}{W}$, $\gamma \leftarrow \gamma^W$ and $R_{\max} \leftarrow R_{\max,W}$, as

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

$$\leq \frac{R_{\max}^2}{(1-\gamma)^2} \frac{G^2(1-\gamma^H)}{N(1-\gamma^W)} \left[1 - \frac{H}{W} \left(\gamma^H - \gamma^{H+W} \right) - \gamma^H \right]$$

□

Theorem 4.3. *Consider an algorithm using the update rule of Eq.* (2)*. Under Assumptions 4.1 and 4.2, with a suitable step size to guarantee* $\mathbb{E}\left[\|\nabla J(\theta;W)\|\right] \leq \epsilon$*, the sample complexity is at most:*

$$NK \geq \Omega\left(\frac{R_{\max}^3(1-\gamma^H)}{\epsilon^2(1-\gamma)^3(1-\gamma^W)^2}\left[1-\frac{H}{W}\left(\gamma^H-\gamma^{H+W}\right)-\gamma^H\right]^2\right).$$

Proof. We first apply from (Papini, 2020, Theorem 7.1), recalling that the assumption enforced in the statement of Theorem 4.3 recall the same as the previous one.

$$NK = 2\left(J^*(W)-J(\theta_0;W)\right)\left(\frac{WN}{\epsilon}+\frac{W\nu_1}{\epsilon^2}\right)$$

We redefine $V \leftarrow \nu$ and $L \leftarrow L$, applying Lemma 4.2 and 4.1. Then, neglecting constants, we obtain

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

□

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

$$NK \geq \widetilde{\Omega}\left(\frac{R_{\max}^3}{(1-\gamma)^3\epsilon^3}\frac{(1-\gamma^H)}{(1-\gamma^W)^2}\left[1-\frac{H}{W}\left(\gamma^H-\gamma^{H+W}\right)-\gamma^H\right]^2\right).$$

Proof. We first apply from (Montenegro et al., 2024, Theorem F.1), recalling that the assumption enforced in the statement of Theorem 4.3 recall the same as the previous one.

$$NK = \frac{16\alpha^4 L_{2,A}V_A}{\epsilon^3}\log\frac{\max\{0, J^*(W)-J(\theta_k)-\beta\}}{\epsilon}$$

We redefine $V \leftarrow V_A$ and $L \leftarrow L_{2,A}$, applying Lemma 4.2 and 4.1. Then, neglecting logarithmic values and constants, we obtain

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

□

B 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.

B.1 Experimental setting

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

$$\begin{aligned}\hat{\nabla}_{\theta_t} J_{\theta_t} &= \mathbb{E}_{p_\theta(\tau)}\left[\sum_{j=0}^{T-1}\sum_{t=0}^{j}\nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)\left(r_j - b_j\right)\right], \\ b_j &= \frac{\mathbb{E}_{p_\theta}(\tau)\left[\left(\sum_{t=0}^{j}\nabla_{\theta_h}\log \pi_\theta\left(a_t \mid s_t\right)\right)^2 r_j\right]}{\mathbb{E}_{p_\theta}(\tau)\left[\left(\sum_{t=0}^{j}\nabla_{\theta_h}\log \pi_\theta\left(a_t \mid s_t\right)\right)^2\right]}.\end{aligned} \tag{17}$$

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 (Kingma, 2014) to adaptively set the step size. The exact method adopted for each environment is shown in Table 1.

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;
- GPU: 16-core GPU.

All the performances are run over 8 CPU cores. Refer to Figures 2b, 3b and 4b for the computational times for each environment for relevant experiments.

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 1: Parameters of the environments.

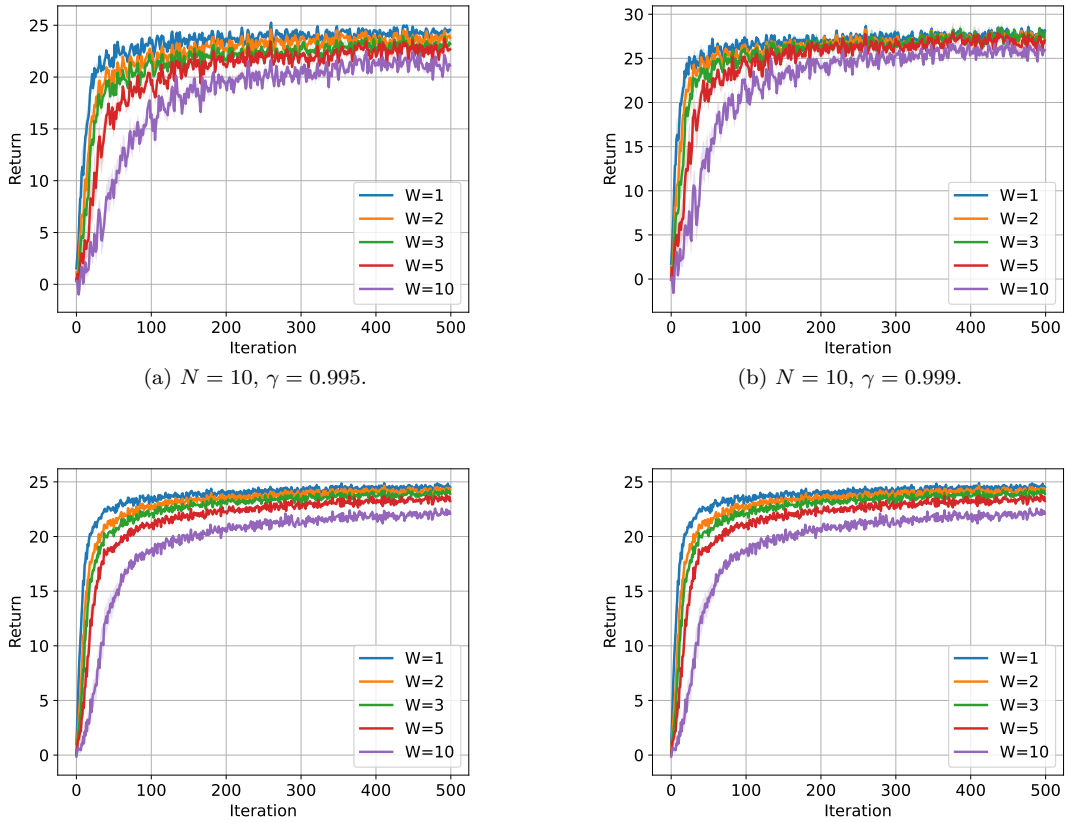
Environment	σ	η	Type
Dam control	0.5	0.005	Adam
Swimmer	1	0.001	Constant
Half-Cheetah	0.1	0.001	Adam

Table 2: Training Parameters for the environments.

B.2 Additional Results

In the following, we present the entire experimental campaign not present in the main paper.

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 Figures 5, 6 and 7, 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 the statements of Theorem 4.4, 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 (Figures 6 and 7).



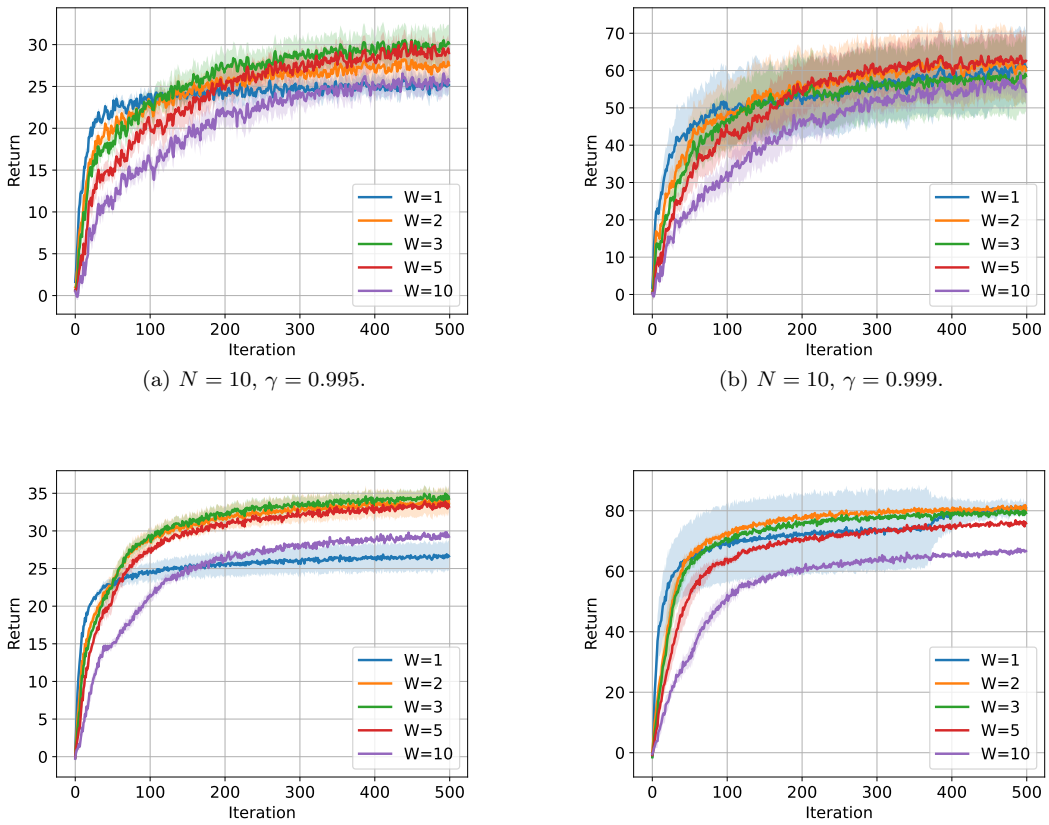
(a) $N = 10$, $\gamma = 0.995$.

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

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

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

Figure 5: Learning curves for Swimmer with $H = 200$.



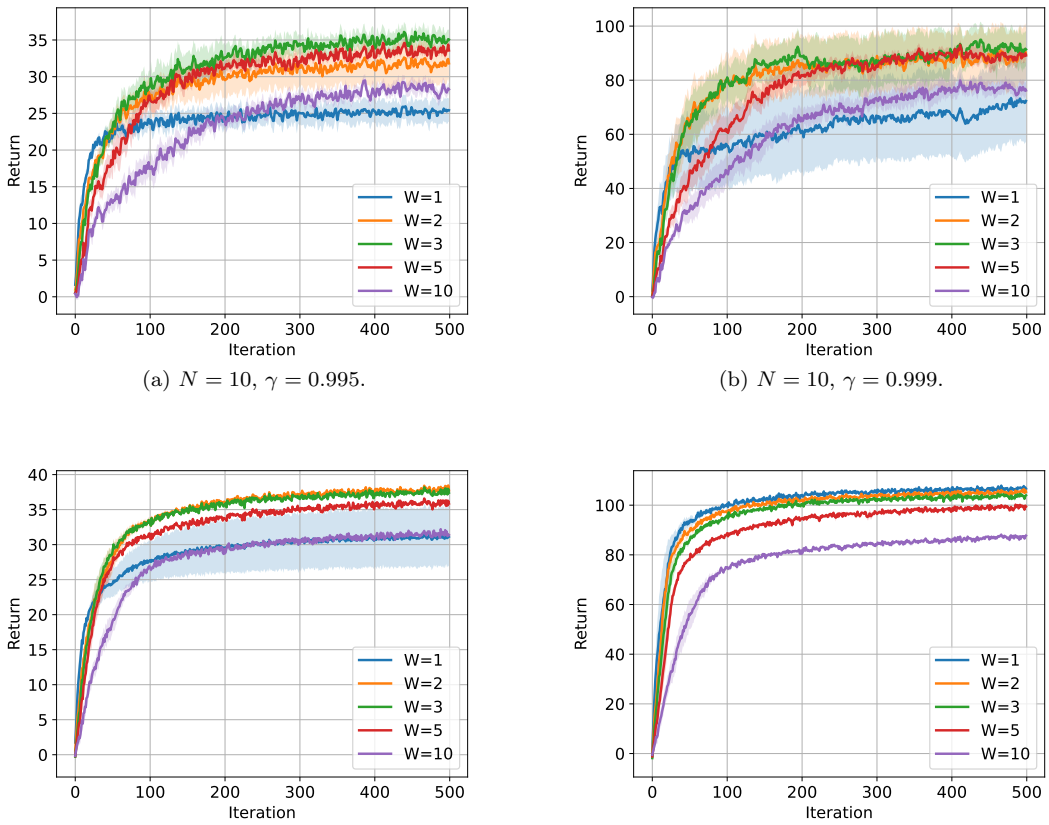
(a) $N = 10$, $\gamma = 0.995$.

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

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

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

Figure 6: Learning curves for Swimmer with $H = 500$.



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

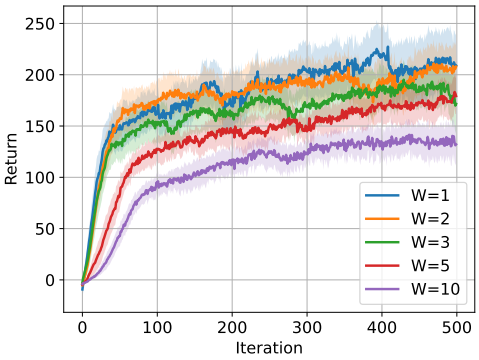
(b) $N = 10$, $\gamma = 0.999$.

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

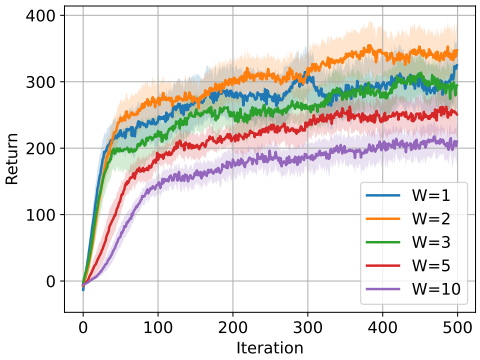
(d) $N = 100$, $\gamma = 0.999$.

Figure 7: Learning curves for Swimmer with $H = 700$.

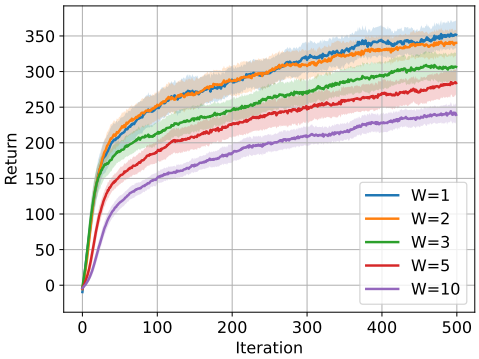
Half-Cheetah. The additional results presented on the Half-Cheetah environment demonstrate the impact of the parameter W w.r.t. longer horizons, higher discount factors and batch size. We mostly appreciate how with small a small batch size ($N = 10$) we are able to estimate less noisy gradient, thus providing more accurate policy updates. We see in Figure 8, 9 and 10, 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 ($\gamma = 0.999$) higher values of W manages also to converge faster w.r.t. $W = 1$, with the benefits of a reduced computational complexity (Figure 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 Figures 9 and 10, 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.



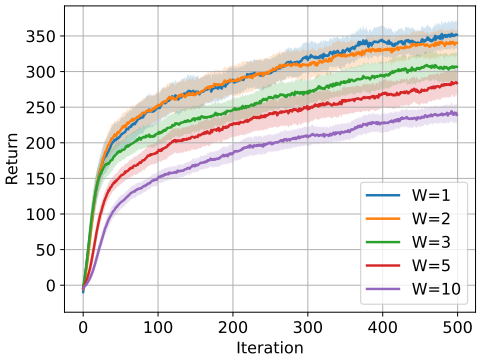
(a) $N = 10$, $\gamma = 0.995$.



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

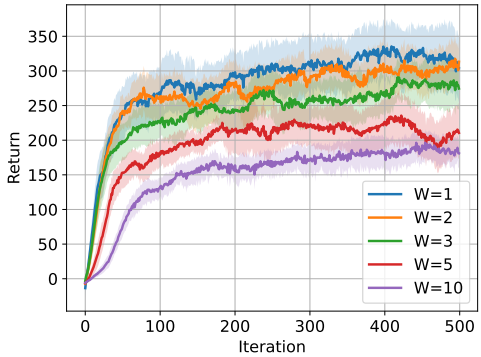


(c) $N = 100$, $\gamma = 0.995$.

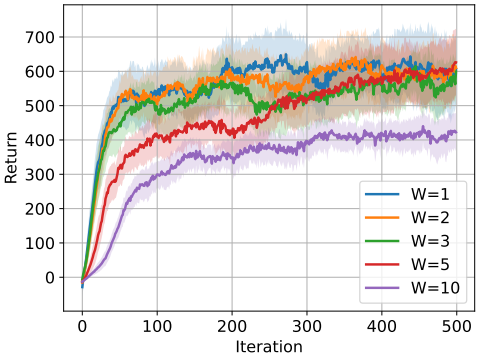


(d) $N = 100$, $\gamma = 0.999$.

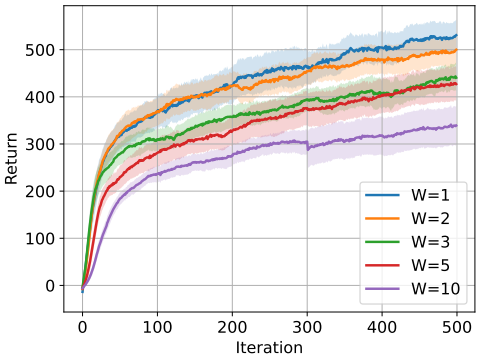
Figure 8: Learning curves for Half-Cheetah with $H = 200$.



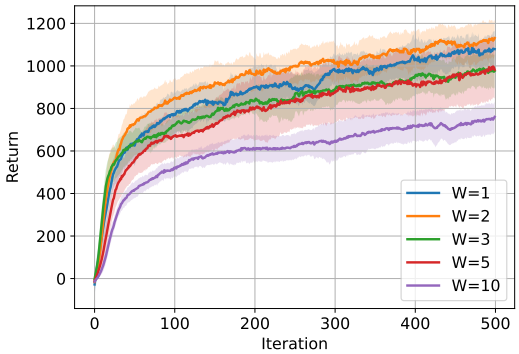
(a) $N = 10$, $\gamma = 0.995$.



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

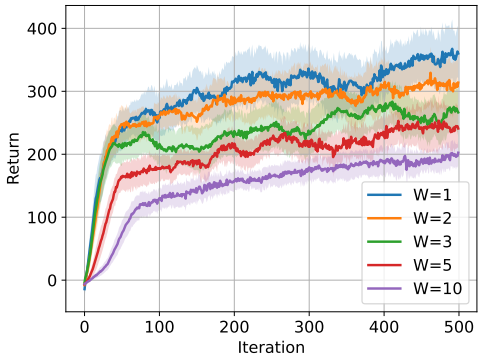


(c) $N = 100$, $\gamma = 0.995$.

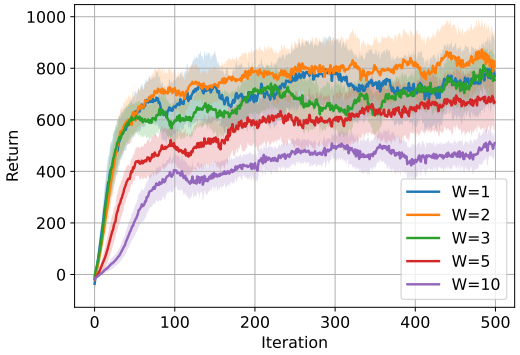


(d) $N = 100$, $\gamma = 0.999$.

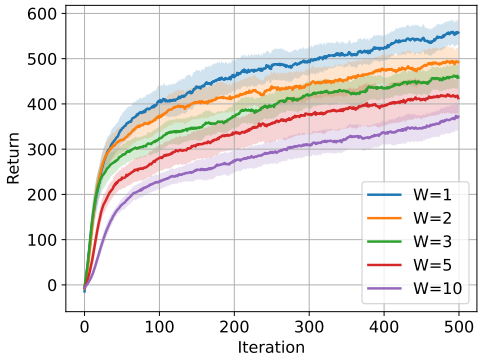
Figure 9: Learning curves for Half-Cheetah with $H = 500$.



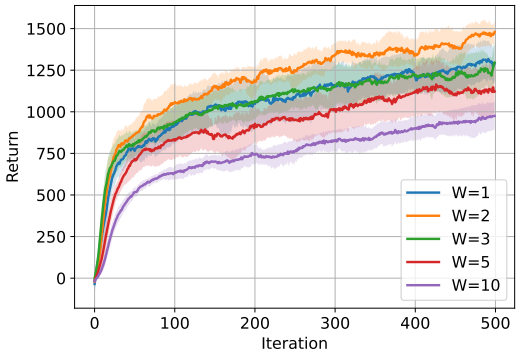
(a) $N = 10$, $\gamma = 0.995$.



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



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



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

Figure 10: Learning curves for Half-Cheetah with $H = 700$.