Best Policy Tracking in Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Reinforcement learning commonly exhibits non-monotonic performance dynamics in single-task settings. During learning, a sequence of policies is generated whose true performance is unknown and may oscillate significantly across iterations. Despite this well-known phenomenon, standard experimental practice typically reports either the final visited policy or the best among periodically evaluated ones, implicitly assuming that these selection criteria reliably approximate the best-performing policy visited throughout the entire sequence.

In this work, we present an empirical study on the implications of non-monotonicity for policy selection in single-task policy-based reinforcement learning. We introduce metrics both to characterize performance degradation and to assess the accuracy and efficiency of different selection criteria. These metrics allow us to evaluate commonly used selection strategies across standard continuous control benchmarks and actor-critic method. Our results show that widely adopted selection practices can deviate considerably from the best policy encountered during learning, leading to underperforming estimates and suboptimal use of computational resources. As an alternative, we recommend a validation-based policy selection strategy constrained by computational cost, which provides a more consistent and accurate approximation of the best visited policy while improving selection efficiency. Overall, our findings highlight policy selection as an underexplored factor in the evaluation protocols of reinforcement learning and emphasize the need for more explicit and standardized validation-based selection procedures.

1 Introduction

Policy optimization in Reinforcement Learning (RL) consists of finding a decision-making model that allows an agent to maximize a cumulative reward while solving a task through interaction with an environment (Sutton & Barto, 2018). With the incorporation of neural networks, this learning paradigm has achieved remarkable results in many decision-making and control problems (Zhang et al., 2023; 2018). For instance, methods such as the Deep Q-Network (Mnih et al., 2015) and Proximal Policy Optimization (Schulman et al., 2017) have shown competitive performance in both discrete and continuous control settings (Bellemare et al., 2013; Mnih et al., 2013; Arza et al., 2025; Makoviychuk et al., 2021), while systems such as AlphaZero (Silver et al., 2018) have surpassed human-level performance in classic games like chess. These results have led to a primary focus on final performance (Jordan et al., 2024). However, this emphasis may overlook important characteristics of the learning process itself, which may affect the reliability of reported performances and the efficient use of computational resources, both crucial concerns given the high cost of experimentation in RL (Agarwal et al., 2021; Jordan et al., 2024; Patterson et al., 2024).

A well-known characteristic of RL is the presence of non-monotonic performance dynamics when solving a specific task (Sutton, 1999; Wagner, 2014; Bertsekas, 2011; van Hasselt et al., 2018). During each policy optimization process, RL methods generate a sequence of policies as the result of iterative updates driven by experience collected through interaction with the environment. However, the performance of these intermediate policies cannot be directly and exactly evaluated (Zhu & Matsubara, 2020; Sutton & Barto, 2018), as it depends on an evolving and partially observed learning process (Zhang et al., 2023; Liu et al., 2023; Javed & Sutton, 2024). As a result, the true performance of the policies can oscillate over the course of learning (Wagner, 2014; Zhu & Matsubara, 2020), with policies repeatedly improving, deteriorating, and

recovering during the optimization process, while updated policies may temporally degrade in performance with respect to the best-performing policy visited so far (Dohare et al., 2023).

Previous research has approached non-monotonicity in RL mainly from three perspectives: understanding its causes (Lyle et al., 2024; van Hasselt et al., 2018; Liu et al., 2023), mitigating its effects through algorithmic design (Zhang et al., 2023; Thomas & Brunskill, 2016; Schulman et al., 2017; Haarnoja et al., 2018), and developing more rigorous evaluation protocols capable of handling the high variability of RL processes (Islam et al., 2017; Khetarpal et al., 2018; Colas et al., 2018; Patterson et al., 2024; Machado et al., 2018; Patterson et al., 2024). As a result, non-monotonicity is now widely recognized as an intrinsic characteristic of RL learning dynamics rather than an exceptional behavior. However, considerably less attention has been devoted to analyzing its implications for the selection of the policy that is ultimately reported as the result of the learning process.

Given the current state-of-the-art approaches to non-monotonicity in RL, a natural question arises: how is the policy ultimately returned as the learned outcome selected from the sequence of visited policies? More importantly, are the selection criteria used in practice able to capture the best performance achieved during learning, while also justifying the invested computational time? If not, these criteria may report a policy with performance below the maximum achieved, thereby attributing to the learning process a lower level of performance than that actually achieved. Moreover, this would imply that part of the learning time may have been unnecessary to justify the reported performance, since the same performance could have already been reached at earlier iterations preceding the visit of the best policy.

Despite the possible practical consequences, policy selection is often treated as a secondary implementation detail within experimental settings. The learned policy is commonly chosen either as the final visited model or as the best among periodically evaluated ones (Ouyang et al., 2022; Le et al., 2025; Machado et al., 2018), implicitly assuming, without guarantees, that these criteria provide a reliable approximation of the best policy encountered during learning. Moreover, several works have pointed out that, although unstable learning behaviors are clearly present, these phenomena remain insufficiently characterized and measured in RL (Dohare et al., 2023; Liu et al., 2023). At the same time, concerns about reliable selection protocols have increasingly emerged (Chan et al., 2020), especially when compared to supervised learning settings where procedures such as cross-validation and separate validation stages are standard practice (Zhang et al., 2018). This work places the overlooked policy selection under non-monotonic learning dynamics at the center of the analysis. Our contributions are as follows:

1. We make explicit commonly used policy selection criteria within the standard single-task policy-based RL framework, integrating them into the sequence of visited policies generated during learning and elevating them from implicit implementation-level heuristics to a unified formal definition.
2. We provide measures both to characterize performance degradation and to assess the accuracy and efficiency of selection criteria in non-monotonic learning processes.
3. Based on the proposed metrics, we analyze policy selection criteria across standard continuous control benchmarks and actor-critic method. We quantify their reliability in identifying the best-performing policy, as well as their robustness across different degradation levels observed throughout non-monotonic learning processes.
4. We identify a systematic gap between the best policy encountered during learning and the policy typically returned by standard selection criteria, showing that policy selection criteria can systematically distort reported performances in RL evaluations.
5. We recommend a validation-based selection strategy constrained by computational cost, which provides a more consistent and accurate approximation of the best visited policy while improving selection efficiency.

Overall, our results show that policy selection under non-monotonic learning dynamics is not a neutral implementation detail, but a factor that can significantly influence both the interpretation of learning performance and the efficiency of experimentation in RL. We hope this work contributes to increasing awareness of how selection decisions may shape reported conclusions, encouraging more explicit and carefully designed selection protocols in future research.

The remainder of this work is organized as follows. Section 2 provides an overview of the causes of non-monotonicity in RL, as well as prior work on its measurement and mitigation, together with existing concerns regarding commonly used policy selection criteria in practice. Section 3 introduces the background formalization and terminology necessary to define the single-task policy-based framework on which our experimental study focuses, integrating the explicit definition of commonly used policy selection criteria, and introducing the metrics proposed to quantify performance degradation, selection accuracy and selection efficiency. Section 4 specifies the design choices used in our experimental analysis, which is presented in Section 5, along with the corresponding results and findings. Section 6 discusses the implications of our findings and outlines future research directions. Finally, Section 7 concludes the paper.

2 Related work

Awareness of non-monotonicity in single-task RL dates back several decades (Sutton, 1999), where it was recognized as a consequence of the gap between strong theoretical assumptions ensuring monotonic learning guarantees and practical settings that relax these assumptions to increase the applicability of classical RL formulations (Sutton & Barto, 2018; Bertsekas, 2007; Dong et al., 2022).

Modern RL relies on function approximation, data generated by the learning process itself, and bootstrapping (Boyan & Moore, 1994; Sutton, 1999; van Hasselt et al., 2018), which together introduce several sources of instability. Firstly, the learning objective is estimated from data generated by the current policy, and often even from policies at previous iterations (Thomas & Brunskill, 2016). Furthermore, the data distribution becomes non-stationary and temporally correlated, as it depends on a continuously evolving policy (Zhang et al., 2023; Liu et al., 2023; Javed & Sutton, 2024). Additionally, stochasticity is intentionally introduced to enable exploration balanced with exploitation (Sutton & Barto, 2018), further increasing variability during learning. Altogether, the learning process defines a closed-loop system in which the policy generates the data used to update itself (Wagner, 2014; Han et al., 2020). All of this breaks assumptions that guarantee monotonic learning, such as data being independent and identically distributed (Zhang et al., 2023; Liu et al., 2023), while bootstrapping further propagates approximation errors by updating estimates from previous estimates, potentially creating fatal feedback loops (Fujimoto et al., 2018). As a result, value-based methods exhibit unstable and divergent learning, with chattering behaviours around a good policy (Sutton, 1999; Wagner, 2014; Bertsekas, 2011). More generally, non-monotonic dynamics can manifest in other forms such as policy collapse, observed in both value-based and policy-based methods, where the learned policy can dramatically worsen after some initial learning as the agent continues to interact with the environment (Dohare et al., 2023).

Several works have attempted to quantify non-monotonicity throughout the learning process of a single task, often using a wide variety of terms to refer to the same underlying phenomenon. Recent studies (Liu et al., 2023; Zhang et al., 2023) measure interference caused by state distribution drift in value-based methods, building on metrics originally developed for catastrophic forgetting in multi-task learning (Serra et al., 2018; Chaudhry et al., 2019). Zhang et al. (2023) argue that temporally correlated and non-stationary training data may lead to catastrophic interference and performance collapse even within single-task learning. They highlight that this can be interpreted as a special case of continual RL, which requires strategies to mitigate interference together with mechanisms to detect and identify sets of states distributed close to each other during single-task learning, playing the role of well-distinguished tasks in multi-task RL (Malagon et al., 2024). Fedus et al. (2020) empirically show how interference arises within Atari games that have long been considered single-task settings. Extending the focus to policy-based methods, Chan et al. (2020) propose variability and risk metrics for quantifying stability across time during training runs. Despite these efforts in measurement, the effect of non-monotonicity in single-task RL remains unclear (Dohare et al., 2023), with no well-established online characterization or standard measures (Liu et al., 2023), especially for policy-based gradient methods (Zhang et al., 2023).

Another line of work in single-task policy-based RL addresses instability by introducing diverse regularization mechanisms that lead to different algorithmic variants. Examples include constrained policy updates enforcing similarity between consecutive policies via KL divergence (Schulman et al., 2015) or clipping functions (Schulman et al., 2017), entropy maximization to incentivize more exploratory policies (Haarnoja et al.,

2018), delayed policy updates to mitigate overestimation biases (Fujimoto et al., 2018), and replay buffers that store past experiences collected under previous policies to capture the evolving data distribution (Mnih et al., 2013; Haarnoja et al., 2018; Fujimoto et al., 2018). Some of these approaches provide theoretical guarantees of monotonic improvement under strong assumptions, such as specific knowledge of environment dynamics (Han et al., 2020), and accurate value estimation or bounded policy updates (Schulman et al., 2015; Eshwar et al., 2025; Zhu & Matsubara, 2020). However, in practice these conditions rarely hold (Zhu & Matsubara, 2020), resulting in methods that mitigate performance degradation in certain scenarios at the cost of sacrificing other desirable properties, but do not eliminate the non-monotonic dynamics observed during learning (Lyle et al., 2024). For example, Schulman et al. (2015) state that the proposed practical algorithms tend to provide monotonic improvements in simulated robotic tasks, although at the cost of hyper-parameter tuning. Dasagi et al. (2019) argue that while constraining policy updates can successfully prevent large negative performance drops, it can also limit exploration that leads to positive improvements. Overall, these techniques can partially mitigate instability during learning, but leave unaddressed how residual non-monotonicity affects experimental evaluation and policy selection.

At the implementation level, within code-level optimizations (Engstrom et al., 2020), multiple RL libraries (Raffin et al., 2021; Petrenko et al., 2020; Makoviychuk et al., 2021) provide heuristic mechanisms for selecting the policy returned as the learned model, often based on periodically saved checkpoints. These approaches either reuse data collected during training or rely on separate evaluation phases to identify the best-performing policy among those visited. However, the reliability and quality of these criteria have rarely been studied systematically (Bellemare et al., 2013). Moreover, a common default choice in many implementations is to simply return the final policy (Machado et al., 2018; Nguyen et al., 2025), despite the lack of guarantees that it corresponds to the best-performing policy encountered during learning. In addition, learning curves are often constructed using estimates derived from training data to avoid additional evaluation costs (Chan et al., 2020; Machado et al., 2018), even though such estimates have been shown to be optimistic compared to validation-based evaluations (Zhang et al., 2018). These practices on implicit selection criteria in libraries leave open questions about their possible influence on the fair and efficient representation of learning processes under non-monotonic dynamics. Chan et al. (2020) argue that high instability may imply that RL methods must be checkpointed and evaluated more frequently in order to capture peak performance, which can be expensive. Furthermore, they state that during training, it can be a waste of computational resources to train an unstable method that tends to forget previously learned behaviors.

While prior work has explored different aspects of non-monotonicity in RL, from its underlying causes to partial measurement and mitigation strategies, its impact on the policies selected and reported in practice remains poorly characterized and analyzed. In this work, we present an empirical study focused on measuring the magnitude of non-monotonicity in RL processes, together with a systematic analysis of the evaluation biases and resource inefficiencies introduced by commonly used selection criteria under such dynamics, focusing on standard continuous control benchmarks and actor-critic method.

3 Background and theoretical framework

In the following, we introduce the formal background and terminology that define the RL framework in which the learning processes analyzed in this work are generated. In addition, we describe the policy selection criteria commonly used in practice to identify the policy returned as the learned model, and define the metrics proposed in this work to quantify non-monotonicity and assess the reliability of these criteria under such phenomenon.

3.1 Interaction-based single-task policy optimization

In RL, interaction-based Single-Task Policy Optimization (STPO) consists of finding the explicit decision-making approach that allows an agent to best solve a specific task, based on the experience observed during the interaction with an environment (Sutton & Barto, 2018). Formally an environment is defined as a Markov decision process, a tuple $\mathcal{E} = (S, A, p, r)$ where S is the *state space*, A is the *action space*, $p\colon S \times A \to \Delta(S)$ is the *state transition probability* function that defines a probability distribution $p(\cdot|s, a)$ with support in S for a given state-action pair $(s, a) \in S \times A$, and $r\colon S \times A \to \mathbb{R}$ is the *reward* function. The agent interacts

with the environment according to a probability function $\pi\colon S \to \Delta(A)$, called the *policy*, which defines a probability distribution $\pi(\cdot|s)$ with support in A for a given state $s \in S$.

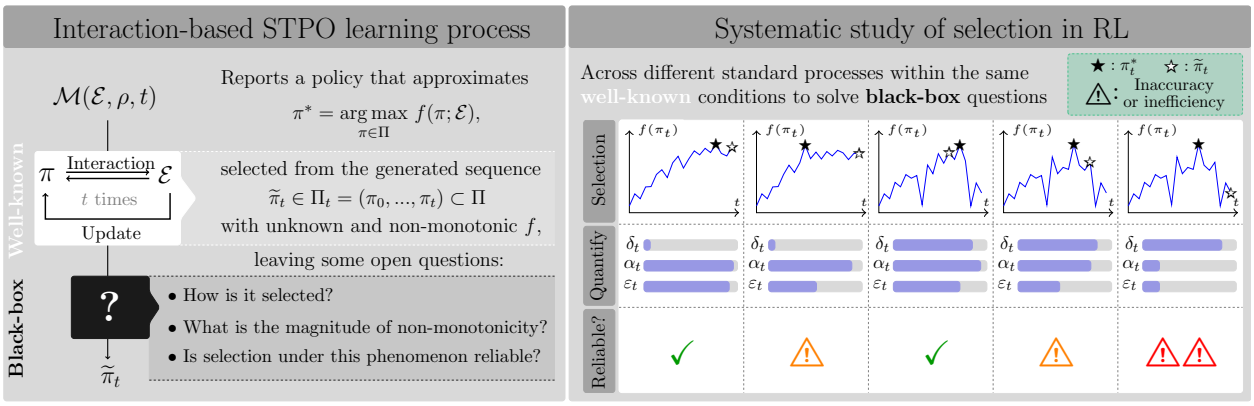

Figure 1: The box on the left summarizes the well-known conditions of the learning processes resulting from interaction-based STPO, along with the open questions left by the black-box use of RL libraries in experimental practice. The box on the right illustrates the contribution of our experimental study focused on selection in RL, which analyzes the reliability of commonly used selection criteria by quantifying performance degradation (δ_t), selection accuracy (α_t), and selection efficiency (ε_t), across different possible scenarios.

At each step of the interaction, the agent starts from the current state of the environment $s \in S$ and chooses an action $a \sim \pi(\cdot|s)$. Then, the environment rewards the chosen action $r(s, a)$ and updates the state $s' \sim p(\cdot|s, a)$. The agent-environment interaction breaks down intentionally when completing a number of maximum steps, or naturally upon reaching a terminal state from $S_T \subset S$. In the latter case, the environment is reset before the next interaction, with the initial state drawn from the initial state distribution $p_I \in \Delta(S_I)$, where $S_I \subset S$. The sequence generated by the state-action pairs $\tau = \big((s_0, a_0), ..., (s_n, a_n)\big)$ during a n step interaction is named *trajectory*, which constitutes an *episode* when the first state of a trajectory is initial $s_0 \in S_I$, the last state-action pair leads to a terminal state $s_{n+1} \in S_T$ and the intermediate states are not terminal $s_i \in S \setminus S_T$ for all $i \in \{1, ..., n-1\}$. Each trajectory is given a reward computed from the rewards of its state-action pairs defined as the discounted cumulative reward that, with a slight abuse of notation, we formally denote as

$$r(\tau) = \sum_{i \in \{0, \ldots, |\tau|\}} \gamma^i r(s_i, a_i), \tag{1}$$

where $\gamma \in [0, 1]$ is the *discount factor* and $|\tau|$ is the length of τ, which is a priori unknown for episodes.

In a STPO problem, the objective is to find the policy that maximizes the average episodic reward over randomly initialized episodes (Schulman, 2016). Since policies are stochastic probabilistic models, in a fixed environment starting from the same initial state, the same policy may generate different episodes. Therefore, the performance of a policy π in an environment $\mathcal{E}$ is measured as the expected episodic reward, hereafter referred to simply as the *policy reward*[1]

$$f(\pi; \mathcal{E}) = \mathop{\mathbb{E}}_{s_0 \sim p_I} \; \mathop{\mathbb{E}}_{\tau \sim (s_0, \pi, p)} \; r(\tau), \tag{2}$$

where $\tau \sim (s_0, \pi, p)$ denotes the episode generated by policy π in the environment with state-transition probability distribution p, starting from the initial state $s_0 \in S_I$. Consequently, the STPO problem is formulated as an optimization problem

$$\pi^* = \arg\max_{\pi \in \Pi} \; f(\pi; \mathcal{E}), \tag{3}$$

where Π is the space of all possible policies.

[1]Once the environment is established and known, we omit it from the notation and write $f(\pi) := f(\pi; \mathcal{E})$. For simplicity, and because the experimental analysis is entirely based on this quantity, we subsequently refer to $f(\pi)$ as the *policy reward*, or simply *reward* when clear from the context.

Interaction-based STPO methods involve learning while interacting with the environment (Sutton & Barto, 2018). Among them, policy gradient methods stand out, which optimize Equation 3 using stochastic gradient descent (Grondman et al., 2012). We have formalized the optimization process defined by an interaction-based STPO method in Algorithm 1.

Algorithm 1 Interaction-based STPO process

Input:
$\mathcal{E}$: Environment;
ρ: Random seed;
t_{max}: Maximum learning interaction steps;
Output: Learned policy

1: $t \leftarrow 0$, $D_t \leftarrow \emptyset$
2: $\pi_0 \leftarrow$ Initialize policy with ρ, $\Pi_t \leftarrow \{\pi_0\}$
3: **while** $t < t_{max}$ **do**
4: $\quad \tau_t \leftarrow$ Interaction of π_t with $\mathcal{E}$ during n steps
5: $\quad D_t \leftarrow D_t \cup \{\tau_t\}$
6: $\quad \widetilde{\pi}_t \leftarrow$ Select best policy in Π_t with a criteria
7: $\quad \pi_t \leftarrow$ Update π_t using D_t
8: $\quad t \leftarrow t + n$
9: $\quad \Pi_t \leftarrow \Pi_t \cup \{\pi_t\}$
10: **return** $\widetilde{\pi}_t$

Given an environment $\mathcal{E}$ and starting from a policy π_0 initialized from a random seed ρ (line 2), the agent–environment interaction and policy update are iteratively repeated until the maximum available interaction steps t_{max} for learning are exhausted (line 3). In each iteration, the gradient of $f(\pi_t)$ is approximated from the trajectory rewards of the finite number of trajectories D_t generated during the previous learning interaction of policies $\pi_0, ..., \pi_t$ with the environment $\mathcal{E}$ (lines 4–5), and the policy is updated according to this estimate[2] (line 7). This results in a sequence of policies Π_t that increases with each learning iteration (line 9) and completely determines the *learning process* defined by the method, the environment, and the random seed, to which the policy that is ultimately returned as learned, $\widetilde{\pi}_t$, belongs (lines 6 and 9).

3.2 Selection criteria

In recent years, many libraries (Bou et al., 2024; Eschmann et al., 2024; Huang et al., 2022; Makoviychuk et al., 2021; Petrenko et al., 2020; Raffin et al., 2021; Makoviichuk & Makoviychuk, 2021) have been developed, aimed at supporting research in RL. They contain implementations of multiple RL environments and methods, providing baselines to experimentally test the performance of new methods (Chua et al., 2018; Pinneri et al., 2021). In experimental studies, we observed that these tools are often treated as black boxes. First, an environment $\mathcal{E}$ and a method $\mathcal{M}$ are selected from those implemented by the library. Then, a random seed ρ is set, and the method is run in the environment for a certain time t, leading to a learning process. Finally, the policy that is returned by default is used as the learned policy without paying attention to how it was selected from the entire sequence of visited policies. Thus, we denote the anytime learning process as $\mathcal{M}(\mathcal{E}, \rho, t) \rightarrow \widetilde{\pi}_t$ (see the box on the left of Figure 1).

Despite that black-box use of RL libraries, it is well known that during an interaction-based STPO learning process, the exact reward of the visited policies is not available due to unknown environment dynamics determined by p, which are only partially observable during training (Zhu & Matsubara, 2020; Sutton & Barto, 2018; Javed & Sutton, 2024). As described in Algorithm 1, policies are updated based on approximations obtained from trajectories collected during finite learning interactions. Furthermore, there is no guarantee that the resulting learning processes will be monotonically increasing (Boyan & Moore, 1994). This means that the evolution of the reward across the visited policies Π_t does not necessarily improve with each iteration, i.e., there may exist consecutive updates from policy π_i to π_j such that $f(\pi_i) < f(\pi_j)$. Under these well-known conditions (highlighted in white in Figure 1, in contrast to black-box practices), the identification of the best policy may not be trivial.

We reviewed some open-source state-of-the-art RL libraries and observed that three simple criteria are implicitly considered for the selection of the output policy. Some of them are based on empirical estimates of

[2]This formulation is intentionally kept abstract to encompass both *on-policy* and *off-policy* learning schemes (Haarnoja et al., 2018; Sutton & Barto, 2018). In on-policy settings (Schulman et al., 2015; 2017), D_t typically contains data generated by the current policy, whereas off-policy methods (Haarnoja et al., 2018; Fujimoto et al., 2018) may reuse or combine data from previous policies. This abstraction does not affect the Markovian nature of the environment, which is defined at the level of state transitions and is independent of the data collection strategy used for policy update.

the reward (see Equation 2) of the policies in the sequence Π_t, considering the sample mean of the episodic reward over a set of episodes D such that

$$\widetilde{f}(\pi; D) = \frac{1}{|D|} \sum_{\tau \in D} r(\tau), \tag{4}$$

where $\pi \in \Pi_t$.

Last visited policy. Libraries such as Stable Baselines3 (Raffin et al., 2021) by default return the last policy of the sequence of policies. We call this criterion *last*, which we formally define as

$$\widetilde{\pi}_t = \pi_t. \tag{5}$$

Best policy using training data. The best policy of the sequence is returned according to the empirical estimate of Equation 4, reusing trajectories stored during learning interactions. For example, Sample Factory (Petrenko et al., 2020) and RL Games (Makoviichuk & Makoviychuk, 2021) consider a finite number of recent κ episodes completed during concatenated trajectories in D_t as the sample D. Finally, the selected policy is the one in the sequence with the highest value of that estimate. In this criterion, using a larger number of samples increases the amount of data available for the estimation, at the cost of mixing episodes collected under multiple previously visited policies. We call this criterion *train*, which we formally define as

$$\widetilde{\pi}_t = \underset{\pi \in \Pi_t}{\arg\max}\ \widetilde{f}(\pi; D). \tag{6}$$

Best policy using validation data. The best policy of the sequence is returned in terms of the empirical estimate of Equation 4 based on additional validation episodes generated in an independent interaction from training. For instance, Stable Baselines3 (Raffin et al., 2021) allows evaluating, with a period of interaction steps φ, the policies using a finite number of additional episodes κ, which constitute D and are generated by the policies themselves starting from initial states drawn from p_I. Finally, the policy with the highest estimate among the evaluated ones is returned. This criterion involves an additional computational overhead proportional to the number of validations and controllable by the evaluation period. We call this criterion *validation*, which we formally define as

$$\widetilde{\pi}_t = \underset{\pi_j \in \{\pi_i \in \Pi_t \,:\, \varphi | i\}}{\arg\max}\ \widetilde{f}(\pi_j; D). \tag{7}$$

In this section, Algorithm 1 and Equations 5, 6, and 7 formalize the policy selection that is implicitly performed in practice at any time during learning. The equations provide a mathematical definition of the selection criteria, explicitly indicating which policy is ultimately returned from the sequence of visited policies. In practical implementations, however, these criteria rely on equivalent tracking procedures that avoid storing the complete sequence of visited policies. For a detailed description of these practical implementations within the learning process of Algorithm 1, we refer the reader to Appendix C.

3.3 Measurements for non-monotonicity and selection criteria

In interaction-based STPO experimental practice, the identification of the best policy is left to the selection criteria described in Section 3.2. However, the magnitude of non-monotonicity remains underexplored (Zhang et al., 2023; Liu et al., 2023), and the reliability of the policies selected and returned as learned has not been systematically studied, despite previous observations of possible dramatic performance drops (Dohare et al., 2023), optimistic estimates from training data (Zhang et al., 2018), and potentially costly validation procedures (Chan et al., 2020). This generates uncertainty in the tracking of the best policy during the learning process. In other words, it is unknown how far the learning curves observed in experimental practice differ from the *ground-truth learning curve*, defined as the curve representing the evolution of $f(\pi_t^*)$

for all iterations[3] $t \leq t_{max}$ such that

$$\pi_t^* = \underset{\pi \in \Pi_t}{\arg\max}\ f(\pi), \tag{8}$$

as well as to what extent these differences may affect the efficiency with which the invested learning time is exploited. To address these issues, we propose three simple diagnostic metrics, computed with respect to the ground-truth learning curve, to quantify performance degradation and the reliability of selection criteria throughout non-monotonic learning processes (see illustrative examples in the box on the right of Figure 1).

Definition 1. Let π_t be the policy visited in the t-th iteration of an interaction-based STPO learning process. We define the performance *degradation* of the learning process at iteration t as the overall reward decrease relative to the average reward achieved so far, which we formally define as

$$\delta_t = \frac{f(\pi_t^*) - f(\pi_t)}{f(\pi_t^*) - \min\left\{f(\pi_t), \text{mean}_{0,t}\right\}}, \tag{9}$$

where $\text{mean}_{0,t}$ denotes the mean of $\{f(\pi_0), \ldots, f(\pi_t)\}$.

Notice that δ_t is defined on $[0, 1]$ for all t, and that at its extreme values, non-monotonicity induces either the largest or smallest decrease in learning performance. When $\delta_t = 1$, at the final learning iteration, the reward has decreased to a level equal to or lower than the average performance achieved so far. When $\delta_t = 0$, however, the policy obtained in the last iteration achieves the best reward observed so far.

In Figure 2, we illustrate three examples of degradation at iterations 39, 66, and 89. In each case, the light grey vertical band represents the difference between the reward of the best policy and the mean reward up to that iteration (denominator of Equation 9), while the darker band within it indicates the difference between the best reward and the reward of the last policy in the considered iteration (numerator of Equation 9). At iteration 39, both policies coincide, so degradation is zero. At iterations 66 and 89, the last policy differs from the best one, exhibiting a decrease in reward that is respectively greater and smaller than half of the difference with respect to the mean reward achieved in each case. Moreover, it can be observed that using the mean reward as a reference leads to comparable degradation values for similar reward drops across different times, since the mean itself tends to increase throughout learning.

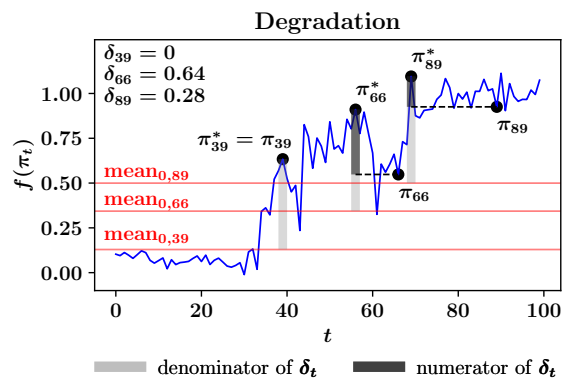

Figure 2: Examples of degradation δ_t (see Equation 9) at different iterations, over 100 synthetic iterations of policy reward.

Additionally, in order to systematically analyze the quality of the selection, we propose two metrics which, like degradation, depend on the learning iteration and are defined on the interval $[0, 1]$.

Definition 2. Let $\widetilde{\pi}_t$ be the policy selected and reported as learned after an interaction-based STPO learning process of t iterations. Firstly, we define the *accuracy* of the selection criterion, in terms of performance, as the reward of the policy selected by the criterion relative to that of the best policy visited so far, i.e.,

$$\alpha_t = \frac{f(\widetilde{\pi}_t) - \min_{0,t}}{f(\pi^*_{t+t'}) - \min_{0,t}}, \tag{10}$$

where t' is the additional time spent validating, which only applies to the *validation* criterion. Secondly, we define the *efficiency* of the selection criterion, in terms of time, as the ratio of the total number of learning

[3]In Section 3.1, the term t denotes the "number of steps", so the subindices of the policies that form the sequence Π_t are the exact number of steps taken in each iteration. This notation aligns with how interaction-based STPO methods operate in practice, as illustrated in Algorithm 1. From Section 4 onwards, for the sake of brevity and with a certain abuse of notation, we consider that t represents the number of iterations, and we denote the subindices of the visited policies as $\pi_0, \pi_1, ..., \pi_t$, unless explicitly stated otherwise.

iterations required to visit, for the first time, a policy with the same or higher reward than that of the policy selected by the criterion, i.e.,

$$\varepsilon_t = \frac{\min\left\{i \leq t\colon f(\pi_i) \geq f(\widetilde{\pi}_t)\right\}}{t}. \tag{11}$$

In Figure 3, we illustrate two selections at iteration 65, together with its accuracies. Specifically, $\widetilde{\pi}_{65}$ simulates the policy selected by both the *train* criterion (in orange) and the *validation* criterion (in green). In the first case, $\widetilde{\pi}_{65}$ is compared against the best policy up to iteration 65, π^*_{65}. Notice that, the difference between the selected policy and the minimum reward up to the current iteration (numerator of Equation 10) is less than half of the difference between the best policy and this minimum (denominator of Equation 10), meaning that the selected policy recovers less than half of the achievable reward. In the second case, $\widetilde{\pi}_{65}$ is compared against the best policy that could be obtained if additional validation time equivalent to $t' = 35$ iterations were used for learning, π^*_{100}. For this selection, accuracy increases, reaching approximately 76% of the best achievable reward relative to the minimum.

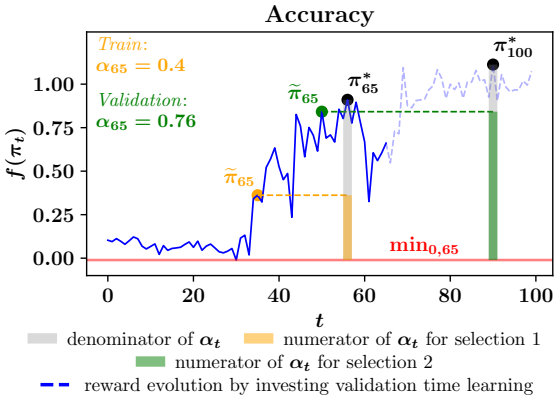

Figure 3: Examples of selection accuracy α_t (see Equation 10) for two different selections, over 100 synthetic iterations of policy reward.

Regarding the efficiency measure, in Figure 4 we illustrate two examples at iteration 100. Each $\widetilde{\pi}_{100}$ corresponds to a different policy selection under an arbitrary criterion. The green case shows an accurate but less efficient selection, since the same reward level was first reached after using only 70% of the already spent learning iterations, while during the remaining 30% the reward oscillates around similar values due to convergence. The orange case, instead, shows a substantial reduction in efficiency due to selection inaccuracy, as the criterion selects a policy whose reward corresponds to early-stage iterations, despite the fact that higher-reward policies were visited during the entire 100 iterations of learning.

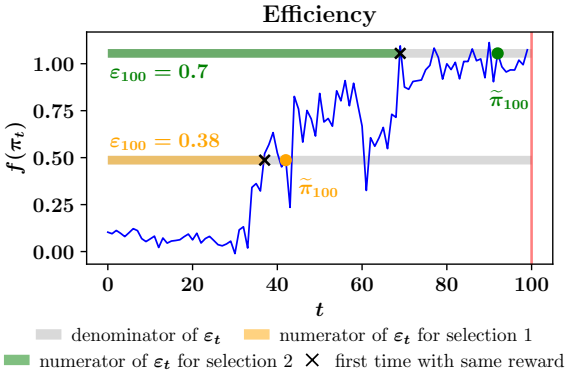

Figure 4: Examples of selection efficiency ε_t (see Equation 11) for two different selections, over 100 synthetic iterations of policy reward.

As the green example in Figure 4 already suggests, an accurate selection does not necessarily imply an efficient one. The proposed accuracy and efficiency measures therefore capture different aspects of the reliability of policy selection. Consequently, both metrics must be considered jointly to properly assess the quality of a selection criterion.

4 Experimental framework

In this section, we describe the experimental setup used within the interaction-based STPO framework to carry out the analysis of selection in RL.

4.1 Learning processes from standard setups

In this work, we analyze learning processes within standard interaction-based STPO setups. We focus on Proximal Policy Optimization (Schulman et al., 2017) method, which is one of the on-policy actor-critic

reference methods (Arza et al., 2025; Makoviychuk et al., 2021), and we consider its open-access implementation in Stable-Baselines3 library (Raffin et al., 2021). The study is conducted on widely used continuous control benchmark environments, specifically Box2D environments LunarLanderContinuous (LLC) and BipedalWalker (BW), and MuJoCo environments Swimmer (S), HalfCheetah (HC), Hopper (H), Ant (A), and Walker2d (W2d), all of which are available in the Gymnasium library (Towers et al., 2025). To ensure comparability with experimental practice and avoid introducing biases in the magnitude of degradation, we rely on the default configurations provided by these libraries, complemented with hyperparameters tuned in RL Baselines3 Zoo (Raffin, 2020). Following current recommendations for robust RL experimentation (Jordan et al., 2024; Patterson et al., 2024), we intentionally minimize intervention in the configuration of the learning processes, and the few modifications considered are detailed in Appendix A.

4.2 Ground truth

To quantify degradation δ_t, accuracy α_t, and efficiency ε_t throughout the learning processes, we require an estimate of the reward (see Equation 2) of the visited policies Π_t. Since the exact value of $f(\pi)$ for each $\pi \in \Pi_t$ cannot be computed in practice due to the unknown environment dynamics and the potentially infinite number of episodes generable by each policy, we approximate it using the empirical estimate defined in Equation 4. For each environment, we fix a large subset of 500 initial states and estimate the reward of each policy from the episodes generated by the policy starting from those states.

The sample mean, when used as a ground truth estimate, remains a point estimate and does not explicitly capture the uncertainty of the underlying random variable (Agarwal et al., 2021). Nevertheless, since episodes are generated by the same policy π under a fixed initial state distribution p_I, the sample mean constitutes an unbiased estimator. Furthermore, by considering a large number of episodes, the variance of the estimator is small compared to the estimates typically used by the *validation* criterion. At the same time, we use the same initial states for all policies in the sequence Π_t, which leads to paired comparisons of rewards between policies, with lower variance in performance differences across learning iterations. Finally, in order to ensure a fair comparison between ground truth and validation (Zhang et al., 2018), notice that the subset of initial states considered for the ground truth estimate differs from the one used during validation in the *validation* criterion. Additional details and empirical evidence supporting the stability of this estimate are provided in Appendix B.

4.3 Representation of ground truth, accuracy and efficiency

Given a method, an environment and a runtime, different random seeds can define completely different ground-truth learning curves (Cheng et al., 2019; Henderson et al., 2018). In fact, the random seed is one of the factors causing variability between learning processes in RL (Chan et al., 2020). To account for this variability in our analysis, following best practice guidelines in RL (Agarwal et al., 2021; Chan et al., 2020), we consider 30 random seeds for each combination of method and environment, and we represent the ground truth $f(\pi_t)$, degradation δ_t, accuracy α_t and efficiency ε_t using the median and interquartile range across the seeds.

Due to the dependence of the defined metrics on the number of learning iterations t, for each learning process we have multiple values, as many as the maximum number of iterations we consider. However, the behaviour of the policy reward is not homogeneous over the learning process, and both the selection criteria and the proposed metrics are inherently dependent on this behaviour. Therefore, we characterize the learning process into three regions, namely *initialization*, *learning*, and *convergence*. This allows us to group metric values within each region and analyze the behaviour of selection criteria under more homogeneous learning dynamics, following the recommendation by Chan et al. (2020).

Intuitively, the initialization phase corresponds to the early stage of training, where performance is highly unstable and no consistent improvement is observed. The learning phase captures the period in which the policy progressively improves, although non-monotonic behavior may still occur. Finally, the convergence phase corresponds to the stage where performance stabilizes and no further significant improvements are observed. We delimit the transition between these regions by two iteration indices, the *learning start* a and the *convergence start* b, which we formally define as follows

$$
\begin{aligned}
a &= \min\big\{\max\{t \in A : \exists i \in A,\ t - w \le i < t\} \cup \{w\}, t_{\max} - w\big\},\\
b &= \max\big\{\min\{t \in B : \exists i \in B,\ t < i \le t + w\} \cup \{T - w\}, a + w\big\},
\end{aligned}
\tag{12}
$$

where $A = \{t\colon f(\pi_t) - \min_{0,1} < \mathrm{std}_{0,t}\}$ contains iterations in which the current reward remains within the observed variability (indicating that no clear improvement has yet occurred), and $B = \{t\colon \max_{t,T} - f(\pi_t) < \mathrm{std}_{t,T}\}$ contains iterations close to the best future reward values (indicating proximity to convergence). Here, $T = 1.5 \cdot t_{\max}$ denotes the maximum execution time extended by half to favour the apparition of convergence in the generated processes.

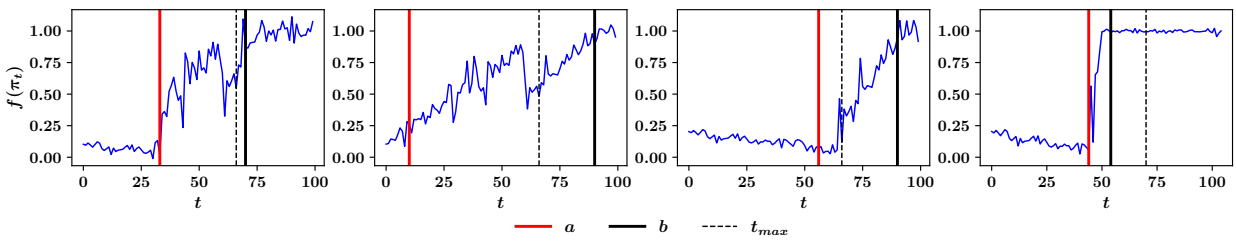

Figure 5: Characterization of the regions according to Equation 12 across four different cases, with 100 synthetic iterations of the policy reward. From left to right: a and b determined by the statistical condition in A and B; a and b imposed by the minimum size condition; a truncated not to exceed t_{max}; and b truncated to ensure a well-defined interval $[a, b]$ with a minimum length of 10%.

Notice that, in Equation 12, we impose a certain local persistence of the conditions in A and B with a window $w = \lfloor 0.1 \cdot T \rfloor$ of 10% length which, in practice, do not hold consistently. Furthermore, the unions guarantee a minimum size for the regions, and the minimum and maximum in a and b respectively, ensure that a is not too large so that $a < t_{max}$ always holds, nor b too small so that $b > a$ always holds. These characteristics guarantee the correct characterization of the regions $[0, a]$, $[a, b]$, and $[b, T]$, which constitute initialization, learning, and convergence, respectively (see Figure 5 and Appendix E for synthetic and benchmark examples).

4.4 Configuration of the selection criteria

The *last* criterion is the simplest of the three described in Section 3.2, and does not require the definition of any parameters for its application. However, the *train* criterion depends on the number of most recent episodes κ completed during learning for the estimation $\widetilde{f}$, and in the case of *validation*, in addition to the number of validation episodes κ, the validation period φ must also be determined.

To carry out the experimental analysis, we set the number of episodes κ for the *train* criterion by evaluating the selection accuracy (see Equation 10) across the three regions of the different environments on a grid of possible values, ultimately selecting the value $\kappa = 50$ that yields the highest average accuracy. For the *validation* criterion, the parameter κ determines the number of validation episodes to be considered in an additional interaction that involves an extra computational cost. Validation with the same number of episodes from different policies does not necessarily imply the same cost, so the same pairs of κ and φ may lead to different validation costs in different processes. To promote a controlled and comparable cost across processes, for a fixed κ we determine the validation period automatically based on a predetermined threshold φ_c for the ratio between validation cost and learning cost. In this way, the resulting test criterion with fixed parameters κ and φ_c adapts the validation period dynamically, to keep the validation-to-learning cost ratio close to φ_c throughout the learning process. We select the pair $\kappa = 5$ and $\varphi_c = 0.2$ from a grid with which we observe the highest average accuracy.

The exact practical implementation of the cost-driven *validation* criterion can be found in Appendix C, while Appendix D details the selected configurations and discusses the validation cost associated with *validation* criteria.

5 Experimental analysis

Building on the established experimental framework, in this section we conduct the experimental analysis. We first examine the distributions of degradation, accuracy, and efficiency across the selected setups. We then analyze the relationship between degradation and the accuracy and efficiency of the selection criteria. Finally, we provide a comparative analysis of the criteria, conditioned on the observed degradation levels. The experimental analysis is supported by figures that are first described in terms of what they show and subsequently interpreted to extract the main findings. Detailed descriptions of the construction underlying each figure are provided in Appendix G.

5.1 Distribution of degradation, accuracy and efficiency

The first plot of Figure 6 shows the distribution of degradation (see Equation 9) for each environment and region across the 30 generated policy sequences. Each distribution is represented by a histogram with 10 bins, where darker colors indicate a larger number of degradation samples within the corresponding interval, and the median is highlighted in red. In general, across all environments and regions, median degradation values are below 0.5 in 18 out of 21 cases, indicating that high degradation levels are relatively uncommon across the considered scenarios.

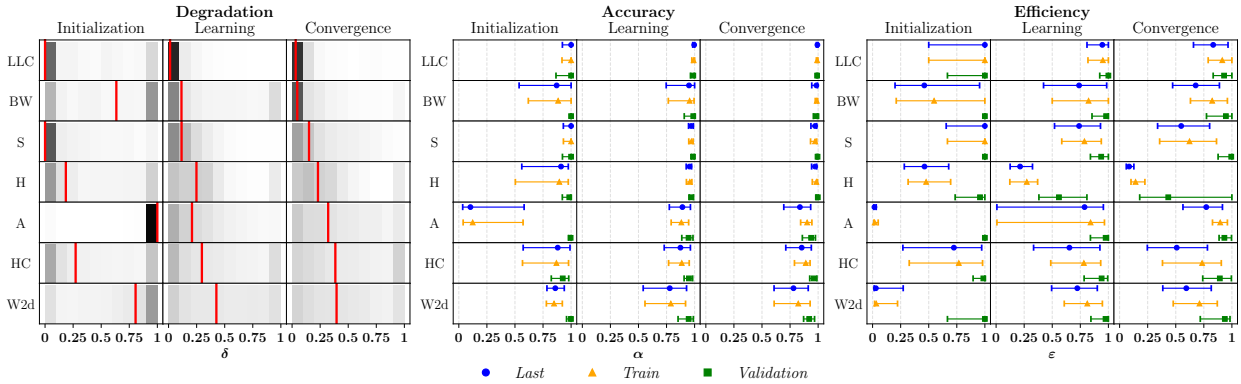

Figure 6: On the left, distributions of degradation in each region for each environment shown as a color-coded histogram with medians in red. In the centre and on the right, distributions of accuracy and efficiency respectively for each selection criterion, with the interquartile range and the median.

We order environments by their median degradation across the three regions to facilitate the analysis of its influence on selection accuracy and efficiency (see Table 1). Interestingly, we observe that LLC and BW exhibit the lowest degradation levels, whereas A, HC, and W2d tend to present higher values.

This ordering is broadly consistent with the relative physics simulation complexity of the environments. As described in their original formulations by Towers et al. (2025), the former are based on the Box2D engine, which involves simpler toy-game-based 2D rigid-body simulations for control tasks, whereas the latter belong to the MuJoCo engine, which represents more challenging robotics-based problems with higher dimensionality and non-linear dynamics. We note that environments with lower degradation are typically those described as simpler in terms of simulation engine and state-action dimensionality, while higher degradation values tend to appear in environments considered more complex according to standard benchmark descriptions.

Table 1: Environments ordered according to the median degradation across 30 policy sequences, together with their characteristics related to the physics simulation complexity. This ordering is used consistently across all figures in the analysis.

$\mathcal{E}$	Engine	$\lvert S\rvert$	$\lvert A\rvert$	Median δ
LLC	Box2D	8	2	0.02
BW	Box2D	24	4	0.10
S	MuJoCo	8	2	0.12
H	MuJoCo	11	3	0.23
A	MuJoCo	27	8	0.30
HC	MuJoCo	17	6	0.32
W2d	MuJoCo	17	6	0.44

The most pronounced differences of degradation between environments occur during initialization. In this region, degradation values are closer to 1, particularly for BW, W2d, and A. These environments also correspond to those with the highest overall degradation levels within each engine group (see Table 1), suggesting that more complex environments require a longer initialization phase before the learning process begins to consistently improve upon the initial random policy. Simpler environments, such as LLC and S, instead tend to exhibit more stable behaviour under random policies, leading to lower degradation during the earliest stages of learning. This interpretation is further supported in Table 5 of Appendix E, where larger learning start limits can be observed for the environments with higher degradation.

The remaining plots of Figure 6 show the selection accuracy (see Equation 10) and efficiency (see Equation 11) of the three criteria for each environment and region. In these cases, distributions are reported as segments, where the central marker indicates the median and the interval represents the interquartile range. Table 2 complements and summarizes these results by providing a compact quantitative overview of the corresponding distributions, dividing them into three ranges delimited by the values 0.25 and 0.75.

Table 2: Count of environment-region cases where median accuracy (α) or efficiency (ε) over 30 policy sequences falls into predefined ranges.

Metric	Criterion	Low	Middle	High
α	*last*	1	0	20
	train	1	0	20
	validation	0	0	21
ε	*last*	4	11	6
	train	3	6	12
	validation	0	2	19

In terms of accuracy, most criteria achieve values above 0.75 in 61 out of 63 cases. The *validation* criterion achieves the highest values above this threshold in all cases (21/21). In contrast, *last* and *train* exhibit lower median accuracies. This trend is more pronounced during initialization, particularly in the case of A, with a median lower than 0.25, which coincides with the highest degradation levels. A similar pattern is observed for efficiency, although with lower values overall (compare the rows of the column named *High* of Table 2). Efficiency is above 0.75 in 36 out of 63 cases. The *validation* criterion again shows consistently high values (19/21), whereas *last* and *train* fall below 0.75 in a larger number of cases (15/21 and 9/21, respectively). In contrast to accuracy, the few cases below 0.25 include environment H, which corresponds to the lowest convergence start limits among all environments (see Table 5 in Appendix E), explaining the decrease in efficiency due to an excessively long learning horizon.

5.2 Relationships between degradation and selection reliability

To analyze the relationship between degradation and the reliability of the selection criteria, Figure 7 shows scatter plots of accuracy and efficiency against degradation, considering all regions for each environment. Each point corresponds to a criterion-region pair within an environment, and its coordinates are determined by the median values of the corresponding metrics. A grid is included to mark the three value ranges (low, middle and high), delimited by the thresholds 0.25 and 0.75, to facilitate the analysis.

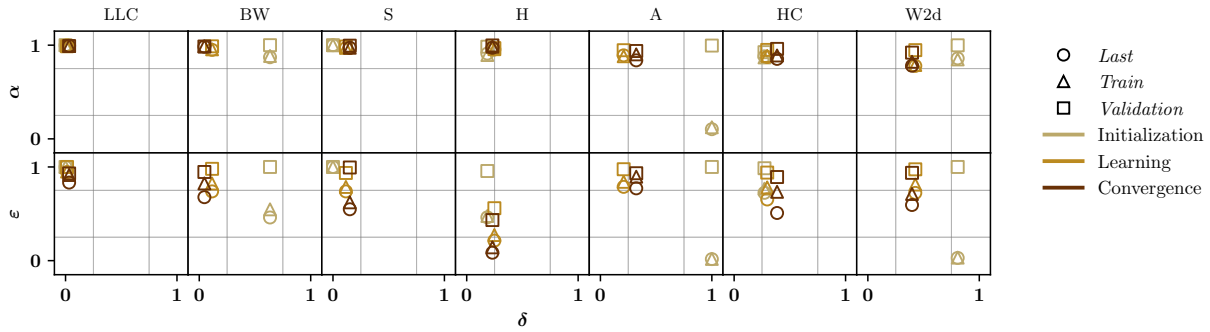

Figure 7: Scatter plots of median degradation, δ, versus selection accuracy, α, (first row) and efficiency, ε, (second row) for the different selection criteria (distinct markers), across regions (colors) and environments (columns).

The scatter plots make explicit several of the patterns already suggested in Section 5.1. In the first row, accuracy remains generally high, with lower values mainly corresponding to the *last* and *train* criteria in initialization regions with higher degradation levels. In contrast, the *validation* criterion remains consistently accurate across all environments, regions, and degradation levels. This suggests that the selection capability of the *last* and *train* criteria deteriorates in scenarios with higher degradation, where identifying the best policy becomes more challenging. Instead, in near-zero degradation cases such as LLC, more sophisticated criteria like *train* and *validation* are not necessary, as the simpler *last* criterion already achieves accurate selections.

In the second row, while *validation* maintains high efficiency levels, both *last* and *train* show a decrease even under moderate degradation levels. Notably, this drop in efficiency is not solely linked to selection inaccuracies, as accuracy remains comparatively high in many of these cases. There is a second source of inefficiency mainly associated with the use of excessively long learning horizons, particularly in environments with later convergence starts, such as H. The two sources of inefficiency coincide with those illustrated in the examples of Figure 4.

An additional observation is the strong similarity between *last* and *train* across both accuracy and efficiency. This is consistent with the fact that *train* is based on empirical estimates of the policy reward computed from trajectories generated during training, which are the same data used for policy updates in on-policy RL methods. This shared data source may limit the ability of the *train* criterion to track occasional performance peaks during the learning process, with estimates that may inherit part of the bias introduced during policy updates. This phenomenon is related to overfitting effects in RL (Zhang et al., 2018), where reward estimates based on training data fail to generalize to the true underlying policy reward. A more detailed analysis of training-based estimations is provided in Appendices B.2 and B.3.

5.3 Comparative analysis of criteria

We continue the analysis by performing pairwise comparisons between the selection criteria. Figure 8 shows the proportion of times each criterion selects a policy with higher reward than its counterpart.

Consistent with the previous analysis, the *validation* criterion achieves the highest proportion of wins. Averaged across all environments and regions, it outperforms both *last* and *train* in approximately 66% of the comparisons. This trend is also reflected in Figure 8, where the green segments corresponding to *validation* are predominant. Furthermore, the proportion of ties, represented by the white spaces, averages only 8% in comparisons involving *validation*, compared to 31% in the *last* versus *train* comparison. This indicates that *validation* not only wins more frequently, but also tends to do so by a larger margin than the other criteria.

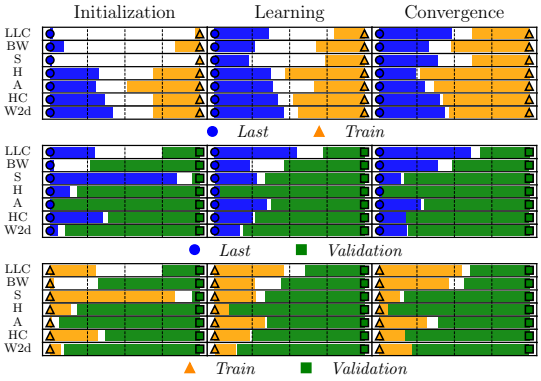

Figure 8: Proportion of times each criterion outperforms the other in pairwise comparisons by region, with ties in white. Dashed vertical lines highlight the proportions 0.25, 0.5, and 0.75.

To further analyze how degradation affects the relative performance of the selection criteria, Figure 9 shows the outcomes of the pairwise comparisons conditioned on degradation levels. For each environment-region pair, each row corresponds to a pair of criteria distinguished by their corresponding colors, while columns represent degradation intervals. Within each cell, the color intensity indicates the proportion of cases in which one criterion outperforms the other. For each pair of criteria, we count the number of cases in which either criterion outperforms the other within each degradation interval (excluding ties). Color intensities are then normalized jointly across both criteria and all degradation intervals using a common maximum frequency. As a result, the figure simultaneously reflects two aspects: the relative proportion of wins between the two criteria within each degradation interval, and the relative number of observed comparisons across degradation levels. Consequently, darker colors have two interpretations. Within each cell, they indicate

a higher probability of one criterion outperforming the other. Across cells, they also indicate that a larger number of comparisons fall within the corresponding degradation range. Table 3 provides the corresponding numerical values by reporting the proportion of pairwise comparisons won by each criterion across three degradation ranges delimited by 0.25 and 0.75 values and also by the three learning regions.

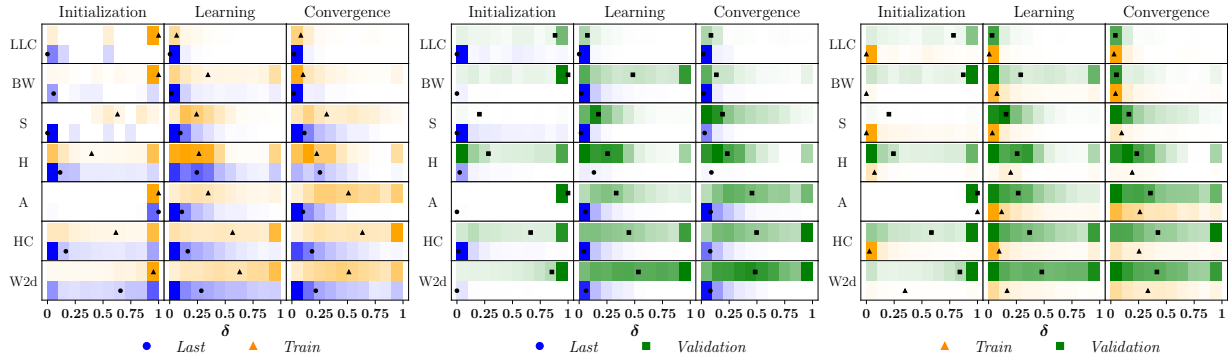

Figure 9: Pairwise comparison of selection criteria across degradation levels for different environments and regions. For each environment-region pair, rows correspond to compared criterion pairs and columns to degradation intervals. Color intensity encodes both win proportions (within each cell) and the number of comparisons (across cells) via joint normalization, while markers indicate median degradation values for each criterion.

Several clear patterns can be observed. First, the advantage of *validation* over both *last* and *train* increases with degradation. Considering all regions together, *validation* outperforms *last* in 66% of low-degradation cases, 95% of middle-degradation cases, and all high-degradation cases (100%). A similar trend is observed against *train*, where the winning proportion increases from 75% to 88% and 92% across the same degradation ranges. These results indicate that the benefits of validation-based selection become increasingly important as degradation grows. In contrast, the differences between *train* and *last* are considerably smaller. Across all regions, *train* outperforms *last* in 44% of low-degradation cases, 61% of middle-degradation cases, and 80% of high-degradation cases. Although *train* tends to perform better on average, the gap remains substantially smaller than that observed for comparisons involving *validation*, reinforcing the similarity between both criteria already observed in Figures 6 and 7.

Table 3: Proportion of pairwise wins between selection criteria across learning regions (initialization, learning and convergence) and degradation ranges (low, middle and high).

Case	Initialization			Learning			Convergence			All regions		
	Low	Middle	High	Low	Middle	High	Low	Middle	High	Low	Middle	High
train wins *last*	0.33	0.41	0.62	0.33	0.57	0.80	0.58	0.68	0.90	0.44	0.61	0.80
validation wins *last*	0.55	0.89	0.99	0.56	0.94	1.00	0.81	0.97	1.00	0.66	0.95	1.00
validation wins *train*	0.62	0.86	0.96	0.69	0.88	0.91	0.86	0.88	0.91	0.75	0.88	0.92

Overall, the results highlight the importance of validation-based selection criteria, which are considerably more robust to degradation than those relying exclusively on training trajectories or on the final visited policy in the learning process.

6 Discussion and future directions

The analysis presented in this work focuses on the best policy selection criteria in interaction-based single-task policy optimization. By explicitly characterizing the degradation patterns along policy sequences and evaluating how different selection criteria respond to them, we provide a framework for understanding the conditions under which best-policy tracking becomes reliable. Beyond the experimental findings themselves, several broader implications and future research directions emerge from this analysis.

6.1 Best-policy tracking beyond on-policy benchmark settings

The experimental study was conducted using Proximal Policy Optimization (Schulman et al., 2017) on a collection of standard continuous-control benchmark environments (Towers et al., 2025). However, the proposed framework operates on the sequence of policies generated during learning and characterizes the degradation patterns observed within that sequence, ultimately analyzing the behaviour of the selection criteria conditioned on the observed degradation level. Consequently, the analysis depends on the properties of the policy sequence itself rather than on the mechanism that generated it.

Specifically, the analysis is conducted on a set of policy sequences that covers a broad range of degradation levels across the initialization, learning, and convergence regions, ranging from nearly monotonic learning dynamics to highly degraded sequences. From this perspective, policy sequences generated by other optimization methods and environments, including off-policy algorithms (Haarnoja et al., 2018; Fujimoto et al., 2018) and more realistic domains (Makoviychuk et al., 2021), are expected to exhibit degradation levels that fall within the range covered by our analysis. Therefore, although the empirical evaluation is necessarily limited to a representative set of methods and environments, the conclusions regarding the behaviour of the selection criteria are expected to remain applicable beyond the specific experimental setting considered in this work.

6.2 Selection criteria within the representation of policy sequences

In practice, sequences of policies are typically represented through learning curves constructed from the reward of the policy selected as best throughout the learning process. As observed in our analysis, different selection criteria may identify different policies within the same sequence, meaning that the resulting learning curve can vary depending on the criterion employed. This observation suggests that selection criteria may influence not only the final policy returned by a learning process, but also empirical practices that rely on comparisons between learning curves. Examples include the comparison of RL algorithms (Patterson et al., 2024; Henderson et al., 2018) and resource-allocation strategies based on intermediate reward estimates (Li et al., 2018; Bohdal et al., 2023).

One illustrative example is the family of early-stopping and adaptive resource-allocation methods, such as successive halving (Jamieson & Talwalkar, 2016), which repeatedly compare the rewards of different policy sequences in order to decide how computational resources should be distributed. If selection criteria introduce systematic distortions in the corresponding learning curves, resource-allocation decisions may also be affected. Preliminary experiments exploring this phenomenon are reported in Appendix F, where initial evidence suggests that default selection criteria implemented in RL libraries may lead to inefficient resource-allocation patterns. A more comprehensive investigation of the influence of selection criteria on fair comparisons between learning curves remains an open research question.

6.3 Adaptive validation-to-learning cost threshold regulation

The validation-based criterion highlighted in our analysis stands out for maintaining high accuracy and efficiency across different degradation levels, outperforming the remaining criteria. This criterion relies on a validation-to-learning cost ratio threshold that controls the computational resources invested in policy validation. Our analysis is based on a threshold value of 0.2, for which we observed consistently accurate best-policy tracking across environments and learning regions, while considering an investment of 20% of the resources devoted to learning in policy validation to be a practical and viable compromise. However, this parameter represents a compromise between tracking accuracy and validation overhead. Larger threshold values allow more policies to be validated, potentially improving tracking accuracy, but at the expense of reducing the computational budget available for learning itself. Conversely, smaller thresholds reduce validation costs but may increase the risk of overlooking high-performing policies. Thus, we consider that studying this trade-off could contribute to the efficient experimental protocols in RL.

In particular, an interesting open question concerns the minimum validation-to-learning cost ratio required to maintain accurate best-policy tracking while reducing validation effort. A related question can be formulated when policy sequences are compared rather than tracked individually. In such settings, the objective is not

only to identify the best policy within a sequence, but also to preserve the reliability of comparisons across different learning processes. This introduces a trade-off between validation cost and comparison accuracy, suggesting the possibility of adaptive validation mechanisms that dynamically regulate the validation-to-learning cost threshold while maintaining sufficiently accurate comparisons. Preliminary evidence related to this question is presented in Appendix F. Moreover, adaptive strategies for controlling such kind of trade-offs have proven useful in other areas of optimization with evolutionary algorithms (Echevarrieta et al., 2024). Therefore, extending these ideas to interaction-based policy optimization constitutes a promising direction for future research.

7 Conclusions

In this work, we studied the implications of non-monotonic learning dynamics for policy selection in single-task policy-based RL. Rather than treating the policy returned at the end of training as an intrinsic outcome of the learning process, we explicitly framed it as the result of a selection criterion operating over the sequence of visited policies generated during learning whose true performance may oscillate significantly throughout learning. Within this perspective, we formalized commonly used selection criteria adopted in experimental practice and analyzed their reliability across standard actor-critic method and continuous control benchmarks.

Our results show that, although the magnitude of performance degradation observed during learning is often moderate, it increases with the complexity of the environment and can substantially affect the quality of the selected policy. In low-degradation settings, simple criteria such as returning the last visited policy may already provide accurate and efficient selections. However, in more unstable scenarios, commonly used default criteria based on the last policy or training data can deviate considerably from the best policy visited during learning, leading to distorted representations of learning performance and inefficient use of computational resources. In contrast, we observed that validation-based selection regulated through an explicit computational cost constraint provides a more consistent and reliable approximation of the best visited policy while maintaining high selection efficiency. Furthermore, our analysis shows that these effects emerge even in standard experimental setups based on widely used method, benchmark environments, and default library configurations.

Overall, our findings suggest that policy selection should not be regarded as a secondary implementation detail in RL experimentation. Moreover, the contribution of this work is not to introduce a fundamentally new selection mechanism, but to make explicit that RL experimentation already relies on implicit policy selection procedures whose consequences are rarely formalized or systematically analyzed. We hope this work contributes toward more explicit and standardized validation-based selection protocols in RL evaluation, analogous to the role that validation practices play in supervised learning, and encourages future research on the interaction between instability, performance representation, and computational efficiency in RL.

Alongside the paper, we provide the source code in our GitHub repository (publicly available upon acceptance). This code allows the reproduction of the experimental analysis, and serves as a reference for building on the recommended validation-based selection criteria.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep Reinforcement Learning at the Edge of the Statistical Precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Etor Arza, Welf Rehberg, Philipp Weiss, Mihir Kulkarni, and Kostas Alexis. Performance-guided Task-specific Optimization for Multirotor Design, October 2025. URL `http://arxiv.org/abs/2510.04724`.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47:253–279, June 2013. ISSN 1076-9757. URL `https://jair.org/index.php/jair/article/view/10819`.

Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific Optimization and Computation Series. Athena scientific, Belmont (Mass.), 3e ed edition, 2007. ISBN 978-1-886529-26-7 978-1-886529-30-4.

Dimitri P. Bertsekas. Approximate policy iteration: A survey and some new methods. *Journal of Control Theory and Applications*, 9(3):310–335, August 2011. ISSN 1672-6340, 1993-0623. URL http://link.springer.com/10.1007/s11768-011-1005-3.

Ondrej Bohdal, Lukas Balles, Martin Wistuba, Beyza Ermis, Cédric Archambeau, and Giovanni Zappella. PASHA: Efficient HPO and NAS with Progressive Resource Allocation. *ICLR*, 2023. URL https://arxiv.org/abs/2207.06940.

Albert Bou, Matteo Bettini, Sebastian Dittert, Vikash Kumar, Shagun Sodhani, Xiaomeng Yang, Gianni De Fabritiis, and Vincent Moens. TorchRL: A data-driven decision-making library for PyTorch. In *ICLR*. arXiv, 2024. URL http://arxiv.org/abs/2306.00577.

Justin Boyan and Andrew Moore. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in Neural Information Processing Systems*, volume 7, 1994.

Stephanie C. Y. Chan, Samuel Fishman, John Canny, Anoop Korattikara, and Sergio Guadarrama. Measuring the Reliability of Reinforcement Learning Algorithms. In *ICLR 2020*, 2020.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K. Dokania, Philip H. S. Torr, and Marc'Aurelio Ranzato. On Tiny Episodic Memories in Continual Learning, June 2019. URL http://arxiv.org/abs/1902.10486.

Richard Cheng, Abhinav Verma, Gabor Orosz, Swarat Chaudhuri, Yisong Yue, and Joel Burdick. Control regularization for reduced variance reinforcement learning. *International Conference on Machine Learning*, pp. 1141–1150, 2019.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. How Many Random Seeds? Statistical Power Analysis in Deep Reinforcement Learning Experiments, 2018. URL https://arxiv.org/abs/1806.08295.

Vibhavari Dasagi, Jake Bruce, Thierry Peynot, and Jürgen Leitner. Ctrl-Z: Recovering from Instability in Reinforcement Learning, October 2019. URL http://arxiv.org/abs/1910.03732.

Shibhansh Dohare, Qingfeng Lan, and A Rupam Mahmood. Overcoming policy collapse in deep reinforcement learning. *Sixteenth European Workshop on Reinforcement Learning*, 2023.

Shi Dong, Benjamin Van Roy, and Zhengyuan Zhou. Simple agent, complex environment: Efficient reinforcement learning with agent states. *Journal of Machine Learning Research*, 23(255):1–54, 2022.

Judith Echevarrieta, Etor Arza, and Aritz Pérez. Speeding-Up Evolutionary Algorithms to Solve Black-Box Optimization Problems. *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2024. ISSN 1089-778X, 1089-778X, 1941-0026. URL https://ieeexplore.ieee.org/document/10387996/.

Judith Echevarrieta, Etor Arza, Aritz Perez, and Josu Ceberio. A Review on Single-Problem Multi-Attempt Heuristic Optimization. *ACM Transactions on Evolutionary Learning and Optimization*, pp. 3795878, February 2026. ISSN 2688-299X, 2688-3007. URL https://dl.acm.org/doi/10.1145/3795878.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Ma. Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO. In *ICLR*, 2020.

Jonas Eschmann, Dario Albani, and Giuseppe Loianno. RLtools: A fast, portable deep reinforcement learning library for continuous control. *Journal of Machine Learning Research*, 25(301):1–19, 2024.

S. R. Eshwar, Gugan Thoppe, Ananyabrata Barua, Aditya Gopalan, and Gal Dalal. Reliable Critics: Monotonic Improvement and Convergence Guarantees for Reinforcement Learning, October 2025. URL http://arxiv.org/abs/2506.07134.

William Fedus, Dibya Ghosh, John D. Martin, Marc G. Bellemare, Yoshua Bengio, and Hugo Larochelle. On Catastrophic Interference in Atari 2600 Games, 2020. URL https://arxiv.org/abs/2002.12499.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *International conference on machine learning*, pp. 1587–1596, 2018.

Ivo Grondman, Lucian Busoniu, Gabriel A. D. Lopes, and Robert Babuska. A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, November 2012. ISSN 1094-6977, 1558-2442. URL http://ieeexplore.ieee.org/document/6392457/.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1861–1870. PMLR, July 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Minghao Han, Lixian Zhang, Jun Wang, and Wei Pan. Actor-Critic Reinforcement Learning for Control With Stability Guarantee. *IEEE Robotics and Automation Letters*, 5(4):6217–6224, October 2020. ISSN 2377-3766, 2377-3774. URL https://ieeexplore.ieee.org/document/9146733/.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep Reinforcement Learning That Matters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32 (1), April 2018. ISSN 2374-3468, 2159-5399. URL https://ojs.aaai.org/index.php/AAAI/article/view/11694.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, and Kinal Mehta. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control. ICML, 2017.

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. *Artificial intelligence and statistics*, pp. 240–248, 2016.

Khurram Javed and Richard S Sutton. The big world hypothesis and its ramifications for artificial intelligence. *Finding the Frame: An RLC Workshop for Examining Conceptual Frameworks*, 2024.

Scott M. Jordan, Adam White, Bruno Castro da Silva, Martha White, and Philip S. Thomas. Position: Benchmarking is Limited in Reinforcement Learning Research. *Proceedings of the 41st International Conference on Machine Learning*, pp. 22551–22569, 2024.

Khimya Khetarpal, Zafarali Ahmed, Andre Cianflone, Riashat Islam, and Joelle Pineau. Re-evaluate: Reproducibility in evaluating reinforcement learning algorithms. ICML, 2018.

Hung Le, Dai Do, Dung Nguyen, and Svetha Venkatesh. Reasoning Under 1 Billion: Memory-Augmented Reinforcement Learning for Large Language Models. In *Transactions on Machine Learning Research*, 2025.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research*, 18:1–52, 2018. URL https://arxiv.org/abs/1603.06560.

Vincent Liu, Han Wang, Ruo Yu Tao, Khurram Javed, Adam White, and Martha White. Measuring and mitigating interference in reinforcement learning. In *PMLR*, pp. 781–795, 2023.

Clare Lyle, Zeyu Zheng, Khimya Khetarpal, Hado van Hasselt, Razvan Pascanu, James Martens, and Will Dabney. Disentangling the Causes of Plasticity Loss in Neural Networks, February 2024. URL http://arxiv.org/abs/2402.18762.

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. *Journal of Artificial Intelligence Research*, 61:523–562, March 2018. ISSN 1076-9757. URL https://jair.org/index.php/jair/article/view/11182.

Denys Makoviichuk and Viktor Makoviychuk. Rl-games: A high-performance framework for reinforcement learning, 2021. URL https://github.com/Denys88/rl_games.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning. In *Neural Information Processing Systems*, 2021.

Mikel Malagon, Josu Cebeiro, and Jose A. Lozano. Self-Composing Policies for Scalable Continual Reinforcement Learning. *International Conference on Machine Learning*, pp. 34432–34460, 2024.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning, December 2013. URL http://arxiv.org/abs/1312.5602.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 0028-0836, 1476-4687. URL https://www.nature.com/articles/nature14236.

Manh Nguyen, Dung Nguyen, Dai Do, Svetha Venkatesh, and Hung Le. Uncertainty-Guided Checkpoint Selection for Reinforcement Finetuning of Large Language Models, November 2025. URL http://arxiv.org/abs/2511.09864.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, and Alex Ray. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022.

Andrew Patterson, Samuel Neumann, Martha White, and Adam White. Empirical design in reinforcement learning. *Journal of Machine Learning Research*, 25(318):1–63, 2024.

Aleksei Petrenko, Zhehui Huang, Tushar Kumar, Gaurav Sukhatme, and Vladlen Koltun. Sample Factory: Egocentric 3D Control from Pixels at 100000 FPS with Asynchronous Reinforcement Learning, June 2020. URL http://arxiv.org/abs/2006.11751.

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolinek, and Georg Martius. Sample-efficient cross-entropy method for real-time planning. *PMLR*, pp. 1049–1065, 2021.

Antonin Raffin. RL Baselines3 Zoo. *GitHub repository*, 2020. URL https://github.com/DLR-RM/rl-baselines3-zoo.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22:1–8, 2021.

John Schulman. *Optimizing Expectations: From Deep Reinforcement Learning to Stochastic Computation Graphs*. PhD thesis, UC Berkeley, 2016.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pp. 4548–4557, 2018.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, December 2018. ISSN 0036-8075, 1095-9203. URL https://www.science.org/doi/10.1126/science.aar6404.

Richard S. Sutton. Open Theoretical Questions in Reinforcement Learning. In G. Goos, J. Hartmanis, J. Van Leeuwen, Paul Fischer, and Hans Ulrich Simon (eds.), *Computational Learning Theory*, volume 1572, pp. 11–17. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. ISBN 978-3-540-65701-9 978-3-540-49097-5. URL http://link.springer.com/10.1007/3-540-49097-3_2.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. A Bradford Book. 2018.

Philip Thomas and Emma Brunskill. Data-Efficient Off-Policy Policy Evaluation for Reinforcement Learning. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 2139–2148. PMLR, June 2016. URL https://proceedings.mlr.press/v48/thomasa16.html.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A Standard Interface for Reinforcement Learning Environments, November 2025. URL http://arxiv.org/abs/2407.17032.

Hado van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep Reinforcement Learning and the Deadly Triad, December 2018. URL http://arxiv.org/abs/1812.02648.

Paul Wagner. Policy oscillation is overshooting. *Neural Networks*, 52:43–61, April 2014. ISSN 08936080. URL https://linkinghub.elsevier.com/retrieve/pii/S0893608014000033.

Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A Study on Overfitting in Deep Reinforcement Learning, April 2018. URL http://arxiv.org/abs/1804.06893.

Tiantian Zhang, Xueqian Wang, Bin Liang, and Bo Yuan. Catastrophic Interference in Reinforcement Learning: A Solution Based on Context Division and Knowledge Distillation. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12):9925–9939, December 2023. ISSN 2162-237X, 2162-2388. URL https://ieeexplore.ieee.org/document/9760159/.

Lingwei Zhu and Takamitsu Matsubara. Ensuring Monotonic Policy Improvement in Entropy-regularized Value-based Reinforcement Learning, August 2020. URL http://arxiv.org/abs/2008.10806.

A Experimental settings

To generate the policy sequences used throughout the analysis, we rely on the default settings provided by the RL libraries, modifying only those aspects that could affect fair comparisons. In this way, our influence is limited solely to the selection of the library and the chosen method–environment pair used to generate the processes, since the remaining parameters determining the process described in Algorithm 1 are entirely specified by the libraries themselves. This includes the maximum execution time t_{max}, the interaction length n, and the architecture of the network modelling the policy.

We only consider three modifications, aimed at promoting comparable estimates under uncertainty, balanced computational resources between learning and validation, and homogeneous experimental conditions across environments. First, unlike the default behaviour of the Stable-Baselines3 (Raffin et al., 2021) library used to generate the learning processes, the two different subsets of initial states employed for the ground truth and for validation in the *validation* criterion remain fixed for each environment and each generated policy sequence, resulting in paired comparisons for both ground truth and validation (Patterson et al., 2024). Second, we perform the validation interaction under the same computational conditions as the learning interaction, using the same level of parallelization by keeping the `n_envs` parameter value identical to that determined in RL Baselines3 Zoo (Raffin, 2020) when creating both learning and validation environments with `make_vec_env` function in Stable-Baselines3. Third, although state normalization is enabled by default for some environments and disabled for others in RL Baselines3 Zoo, we disable normalization in all environments to maintain homogeneous experimental conditions.

B Estimation of policy reward

In this appendix, we discuss and compare in detail the different empirical estimators of the policy reward considered throughout this work.

B.1 Ground truth

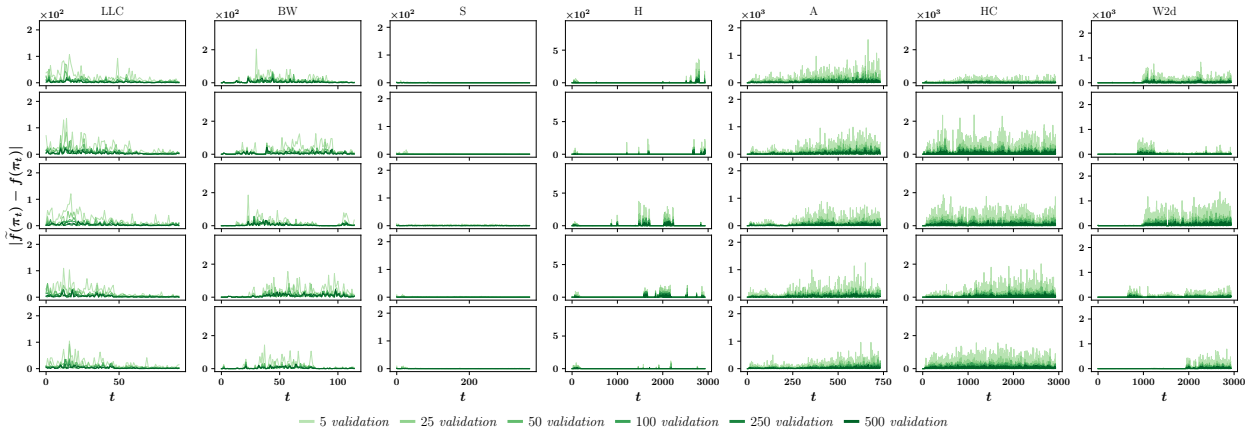

Figure 10: Evolution of the absolute differences between the ground truth and the empirical estimators of *validation* criteria across the policies in the generated sequences for each environment (by column) and for 5 different seeds (by row). Six curves are shown, corresponding to different numbers of episodes κ used for the estimation, where darker curves indicate a larger number of episodes considered.

Each policy π in the sequence of policies Π_t visited during a learning process has associated a random variable that captures the episodic rewards of all episodes that can be generated by the policy starting from all possible initial states, i.e., $X = \{r(\tau) \colon \tau \sim (s_0, \pi, p),\ s_0 \in S_I\}$. The real objective value of a policy depends on this variable (see Equation 2). However, in practice only a finite sample is observable (Jordan et al., 2024). Therefore, in this work, for each environment considered, we fix a large subset of initial states $S_I^{truth} \subset S_I$

and represent the policy ground truth objective as the empirical estimate (see Equation 4) computed from the episodes generated by the policy starting from those initial states, i.e., $D = \{\tau \colon \tau \sim (s_0, \pi, p),\ s_0 \in S_I^{truth}\}$. Specifically, we consider a subset of 500 initial states, a size for which we observe stable estimates.

In Figure 10, different darknesses of green show the evolution of the difference between the ground truth empirical estimates obtained from 500 initial states and those obtained from another independent subset of initial states of varying size. We observe that, as the size of the second subset approaches 500, the corresponding curve becomes closer to the horizontal line at 0, and this behavior already appears for substantially smaller subset sizes. Notice that only the lightest green curves visibly deviate from this horizontal line. This suggests that even using a different and smaller sample of initial states produces very similar estimates, supporting that the 500 episodes considered for the ground truth are sufficient.

B.2 Estimates by *train* and *validation* criteria

The *train* and *validation* criteria defined in Section 3.2 also rely on empirical estimates of the policy reward. However, the set D used in these criteria differs from the one considered for the ground truth in several crucial aspects, which makes their estimates inherently less reliable.

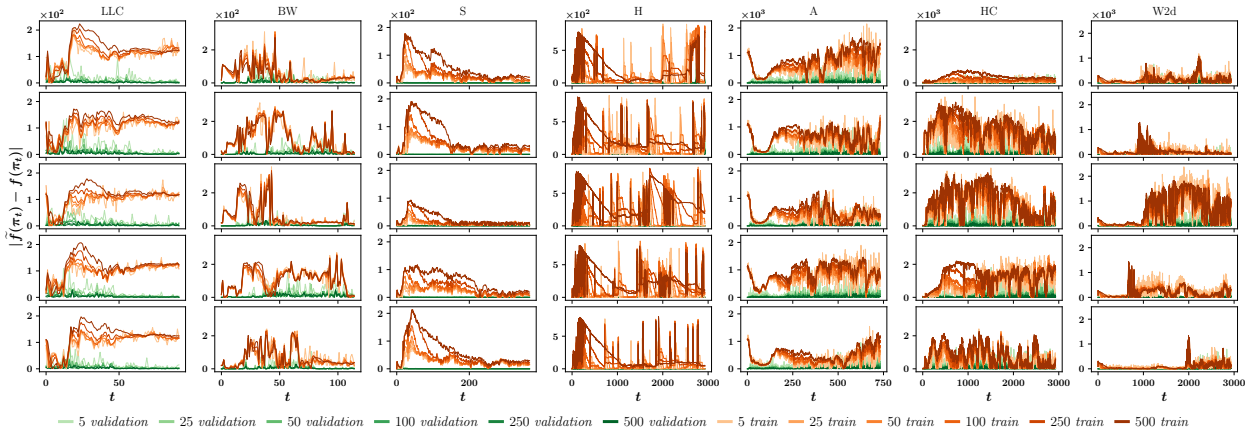

Figure 11: Evolution of the absolute differences between the ground truth and the empirical estimators of *train* (orange) and *validation* (green) criteria across the policies in the generated sequences for each environment (by column) and for 5 different seeds (by row). Six curves are shown per criterion, corresponding to different numbers of episodes κ used for the estimation, where darker curves indicate a larger number of episodes considered. The *validation* curves coincide with those shown in Figure 10.

The *train* criterion reuses episodes generated during learning, resulting from the concatenation of trajectories generated by the policies in the sequence. Consequently, the estimate associated with each policy may include episodes generated by different policies. Moreover, estimates across policies are not paired, since they rely on episodes initialized from different initial states. In addition, the episodes are generated under the stochasticity injected during training to encourage exploration, which is absent during validation. All these factors contribute to the *train* estimates being substantially worse than those of *validation*, as illustrated in Figure 11. The orange curves lie farther from the horizontal line at 0 than the green curves. In some cases, such as LLC and S, considering more episodes even increases the gap, since the darker orange curves deviate more strongly. This is consistent with the effect of incorporating too many episodes generated by policies that are increasingly different from the current one.

For the *validation* criterion, the situation improves considerably. The estimates become unbiased again because the validation episodes are generated by the evaluated policies themselves. Furthermore, comparisons are paired, since the same subset of initial states is considered when validating different policies. As a result, even the curves associated with a very small number of validation episodes remain much closer to the horizontal line at 0 than any of the *train* curves. It is worth noting that the last desirable property is

not satisfied in some libraries such as Stable-Baselines3 (Raffin et al., 2021), where, by default, validation episodes are generated from newly sampled initial states each time a policy is evaluated, potentially producing different initial state samples across validations.

B.3 Overfitting of *train* estimates

It is observed in the RL literature that reward estimates based on training data may fail to generalize to the true underlying policy reward, a phenomenon commonly referred to as overfitting (Zhang et al., 2018). This issue is particularly relevant for the *train* criterion, where the same data used to update the policy is also used to construct empirical reward estimates, potentially introducing a shared bias.

In this appendix, we further illustrate this effect by analyzing the evolution of empirical reward estimates along five policy sequences for each environment. Figure 12 compares, for each sequence, three curves: the ground-truth reward, the estimate computed from training episodes using the selected *train* configuration, and the estimate computed from validation episodes using the selected *validation* configuration. These configurations correspond to those used in the main experimental analysis in Section 5, which were selected due to their overall high accuracy across environments and learning regions (see Appendix D.1 for further details).

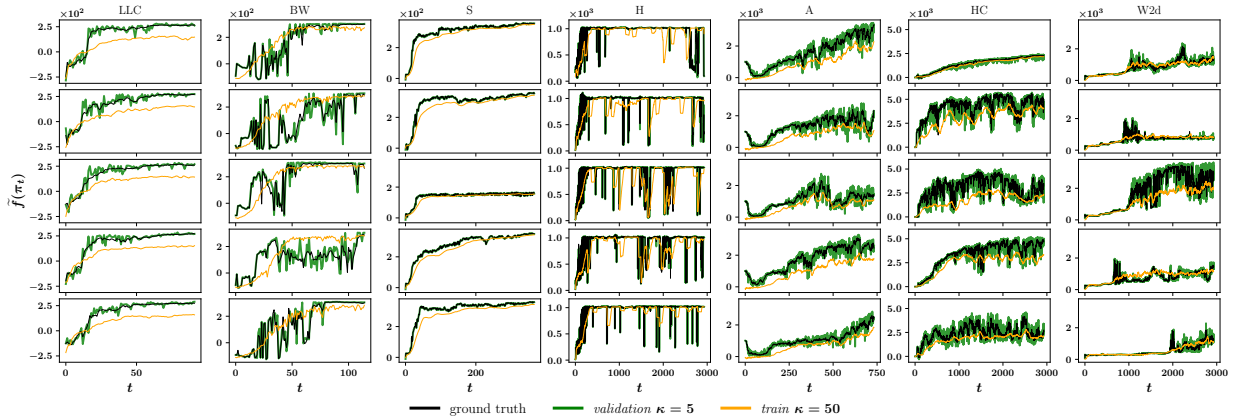

Figure 12: Evolution of the ground-truth episodic reward (black curve) and empirical reward estimates from training trajectories (orange curve for the *train* criterion) and validation episodes (green curve for the *validation* criterion) across five different policy sequences for each environment.

The results highlight clear differences between the two estimators. The *validation* estimate closely follows the ground-truth curve, accurately capturing non-monotonic behaviour and reward drops throughout training. In contrast, the *train* estimate is generally smoother and often exhibits an overly monotonic increasing trend. In some cases, it partially captures fluctuations in reward, but these are attenuated or detected with a noticeable delay. This behaviour is consistent with the use of shared training data for both policy updates and reward estimation, which can propagate bias from the learning dynamics into the estimator itself. Overall, these results provide further evidence that validation-based estimates are more reliable for tracking true policy reward during learning.

C Selection criteria within on-policy learning processes

In Section 3, the interaction-based STPO learning process and the selection criteria were defined to formalize the conditions for policy selection and to specify the policy ultimately returned. These definitions make explicit the policy that is returned in practice from the sequence of visited policies. However, in practical implementations, learning processes do not store the entire sequence of visited policies, as represented in line 9 of Algorithm 1. Instead, they only track the best policy observed so far, which becomes the only policy that is effectively stored during learning.

In this appendix, we explicitly illustrate how selection criteria operate in practice during learning processes. We present one algorithm for each criterion, all of them based on the same interaction-based learning framework described in Algorithm 1, but differing in how the best solution is tracked throughout learning. In particular, we make explicit the operations hidden behind line 6 of Algorithm 1, which are now highlighted in different colors, while also showing the actual data storage performed during learning.

It is important to note that the policies returned by the practical Algorithms 2, 3, and 4 coincide with the policies defined by the selection criteria in Equations 5, 6, and 7, respectively. The difference lies in how they are represented: in the theoretical framework, the criteria are defined explicitly over the complete sequence of visited policies, whereas practical implementations only keep track of the best policy observed so far during learning.

Algorithm 2 On-policy STPO with *last*

Input:
$\mathcal{E}$: Environment;
ρ: Random seed;
t_{max}: Maximum learning interaction steps;

Output: Learned policy

$t \leftarrow 0$
$\pi_0 \leftarrow$ Initialize policy with ρ
while $t < t_{max}$ **do**
  $\tau_t \leftarrow$ Interaction of π_t with $\mathcal{E}$ during n steps
  $\widetilde{\pi}_t \leftarrow \pi_t$
  $\pi_t \leftarrow$ Update π_t using τ_t
  $t \leftarrow t + n$
return $\widetilde{\pi}_t$

Our experimental analysis is based on Proximal Policy Optimization (Schulman et al., 2017), which is an interaction-based and, more specifically, an on-policy actor-critic method. On-policy methods update the policy exclusively using trajectories generated by the current policy itself (lines 6, 11, 12, and 17 of Algorithms 2, 3, 4 and 5, respectively). Consequently, in practice, these methods do not need to store all trajectories generated during learning either, meaning that line 5 of Algorithm 1 is not required for the policy update itself.
Algorithm 2, corresponding to the *last* criterion, simply keeps the current policy at each iteration, without storing additional policies or learning trajectories.

Algorithm 3, associated with the *train* criterion, instead depends on an additional input parameter κ and requires storing learning trajectories, since they are used to compute the empirical estimate of the visited policies (lines 6–7). In this case, the stored best policy is updated whenever the current policy achieves a higher empirical estimate than the best one observed so far (lines 8–10).

Algorithm 3 On-policy STPO with *train*

Input:
$\mathcal{E}$: Environment;
ρ: Random seed;
t_{max}: Maximum learning interaction steps;
κ: Number of validation episodes;

Output: Learned policy

1: $t \leftarrow 0$
2: $\pi_0 \leftarrow$ Initialize policy with ρ
3: $D_t \leftarrow \emptyset$, $\widetilde{f}_{max} \leftarrow -\infty$
4: **while** $t < t_{max}$ **do**
5: $\tau_t \leftarrow$ Interaction of π_t with $\mathcal{E}$ during n steps
6: $D_t \leftarrow D_t \cup \{\tau_t\}$
7: $D \leftarrow$ Last κ episodes in D_t
8: **if** $\widetilde{f}(\pi_t; D) > \widetilde{f}_{max}$ **then**
9: $\widetilde{f}_{max} \leftarrow \widetilde{f}(\pi_t; D)$
10: $\widetilde{\pi}_t \leftarrow \pi_t$
11: $\pi_t \leftarrow$ Update π_t using τ_t
12: $t \leftarrow t + n$
13: **return** $\widetilde{\pi}_t$

Algorithm 4 On-policy STPO with *validation*

Input:
$\mathcal{E}$: Environment;
ρ: Random seed;
t_{max}: Maximum learning interaction steps;
φ: Validation period;
κ: Number of validation episodes;

Output: Learned policy

1: $t \leftarrow 0$
2: $\pi_0 \leftarrow$ Initialize policy with ρ
3: $\widetilde{f}_{max} \leftarrow -\infty$
4: $S_I^{test} \leftarrow$ Drawn κ initial states from p_I
5: **while** $t < t_{max}$ **do**
6: $\tau_t \leftarrow$ Interaction of π_t with $\mathcal{E}$ during n steps
7: **if** $\varphi \mid t$ **then**
8: $D \leftarrow \left\{\tau \colon \tau \sim (s_0, \pi_t, p),\ s_0 \in S_I^{test}\right\}$
9: **if** $\widetilde{f}(\pi_t; D) > \widetilde{f}_{max}$ **then**
10: $\widetilde{f}_{max} \leftarrow \widetilde{f}(\pi_t; D)$
11: $\widetilde{\pi}_t \leftarrow \pi_t$
12: $\pi_t \leftarrow$ Update π_t using τ_t
13: $t \leftarrow t + n$
14: **return** $\widetilde{\pi}_t$

Algorithm 4 represents the *validation* criterion. This criterion introduces an additional input parameter, the validation period φ, which determines how often validation is performed. Unlike *train*, this criterion does not store learning trajectories. Instead, whenever the number of interaction steps elapsed becomes a multiple of the validation period φ (line 7), validation episodes generated by the current policy are collected (line 8) starting from the κ validation initial states fixed at the beginning of learning (line 4). Afterwards, the same update operation used in *train* is performed to replace the currently stored best policy whenever a higher empirical estimate is obtained, now using validation episodes to compute the estimate.

Algorithm 5
On-policy STPO with cost-drive *validation*

Input:
$\mathcal{E}$: Environment;
ρ: Random seed;
t_{max}: Maximum learning interaction steps;
κ: Number of validation episodes;
φ_c: Validation-tolearning cost ratio threshold;
Output: Learned policy

1: $t \leftarrow 0$
2: $\pi_0 \leftarrow$ Initialize policy with ρ
3: $\widetilde{f}_{max} \leftarrow -\infty$
4: $c_{learn} \leftarrow 0$, $c_{val} \leftarrow 0$, $\widetilde{c} \leftarrow 0$, $c_\tau \leftarrow -\infty$
5: $S_I^{test} \leftarrow$ Drawn κ initial states from p_I
6: **while** $t < t_{max}$ **do**
7: $\quad \tau_t \leftarrow$ Interaction of π_t with $\mathcal{E}$ during n steps
8: $\quad c_{learn} \leftarrow c_{learn} + c(\tau_t) + c(\pi_t)$
9: $\quad$ **if** $\frac{c_{val} + \widetilde{c}}{c_{learn}} \leq \varphi_c$ **then**
10: $\quad\quad D \leftarrow \{\tau \colon \tau \sim (s_0, \pi_t, p), s_0 \in S_I^{test}\}$
11: $\quad\quad$ **if** $\widetilde{f}(\pi_t; D) > \widetilde{f}_{max}$ **then**
12: $\quad\quad\quad \widetilde{f}_{max} \leftarrow \widetilde{f}(\pi_t; D)$
13: $\quad\quad\quad \widetilde{\pi}_t \leftarrow \pi_t$
14: $\quad\quad c_\tau \leftarrow \max\left\{c_\tau, \max_{\tau \in D} c(\tau)\right\}$
15: $\quad\quad \widetilde{c} \leftarrow c_\tau \cdot \kappa$
16: $\quad\quad c_{val} \leftarrow c_{val} + \sum_{\tau \in D} c(\tau)$
17: $\quad \pi_t \leftarrow$ Update π_t using τ_t
18: $\quad t \leftarrow t + n$
19: **return** $\widetilde{\pi}_t$

Finally, Algorithm 5 defines the proposed slightly modified version of the original *validation* criterion. This criterion is guided by computational cost, rather than by the number of interaction steps t that limits the learning process. During learning, three variables associated with computational cost are updated: the estimated validation cost of κ episodes $\widetilde{c}$, the total cost invested in validation c_{val}, and the total cost invested in learning c_{learn} (initialized in line 4). At each iteration, the learning cost is increased by the cost of the interaction and the policy update (line 8), while the validation cost is increased whenever κ validation episodes are generated (line 16). Consequently, at each iteration, a check is performed to determine whether validating the current policy is expected to incur a computational cost lower than or equal to the predefined threshold (line 9). If this condition holds, the policy is validated (lines 10–13), and the validation cost estimate is updated using a pessimistic estimate based on the maximum validation episode cost observed so far (lines 14–15).
The notation $c(\cdot)$ used in lines 8, 14, and 16 denotes the computational cost required to obtain the corresponding element. For example, $c(\tau_t)$ and $c(\pi_t)$ represent the cost of performing the last interaction of n steps and the last policy update in lines 7 and 17, respectively.

D Configuration of *train* and *validation* criteria

In this appendix, we explicitly describe the selection of the *train* and *validation* criteria configurations considered in the experiments of Section 5. In addition, we include an analysis of the validation cost associated with the different policies in the sequences of policies generated during a learning process, and how this influences the validation-to-learning cost ratio for both the default *validation* criterion and the cost-driven *validation* criterion proposed in this work.

D.1 Criteria configuration for experimentation

In Stable-Baselines3 library (Raffin et al., 2021), several default values are defined that determine the configuration of the *train* and *validation* criteria. On the one hand, the parameter `stats_window_size`, by default set to 100 for any environment within the PPO class associated with the on-policy Proximal Policy Optimization method (Schulman et al., 2017), plays the role of the parameter κ defined in this paper for the *train* criterion, as the number of training episodes considered for the empirical estimation. On the other

hand, in the `EvalCallback` class, which allows periodic validation of the visited policies, the parameters `n_eval_episodes` and `eval_freq`, with default values of 5 and 10000 respectively, play the role of κ and φ defined in this paper for the *validation* criterion. In this latter case, the evaluation period `eval_freq` is measured in interaction steps, which corresponds to different numbers of iterations (see column 4 of Table 4), since for each environment RL Baselines3 Zoo library (Raffin, 2020) uses different learning interaction lengths (represented by n in Algorithm 1). These default configurations are used for the *train* and *validation* criteria in the Figures 21 and 22 of Appendix F.

Table 4: Default and selected configurations for the *train* and *validation* criteria across environments.

$\mathcal{E}$	*Train* default κ	*Validation* default κ	*Validation* default φ 10000 steps $\approx$ iter.	*Train* selected κ	Cost-driven *validation* selected (κ, φ_c)
LLC	100	5	10	50	(5,0.2)
BW			5		
S			10		
H			20		
A			5		
HC			20		
W2d			20		

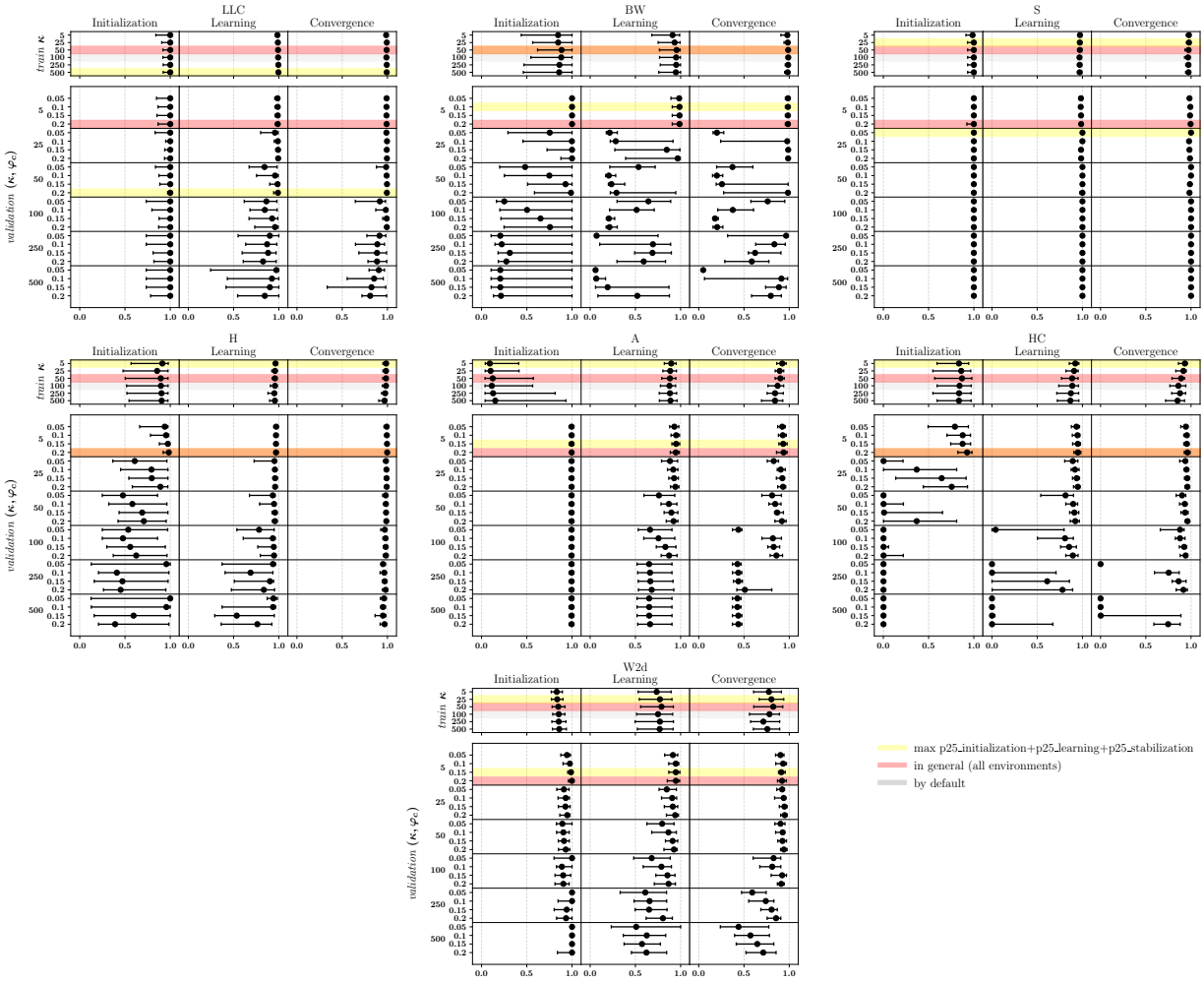

Figure 13: Distribution of selection accuracy for different configurations of the *train* and cost-driven *validation* criteria across environments and regions. Each distribution is represented by its interquartile range, with the median marked within it. The most accurate configurations overall across all regions and environments are highlighted in red, those that are most accurate for each individual environment are highlighted in yellow, and the default configurations are highlighted in grey.

For the main analysis of the criteria in Section 5 instead, we fix configurations for the *train* and cost-driven *validation* criteria that perform well across all environments. To this end, we define a grid of possible parameter values and analyze the selection accuracies (see Equation 10) of each configuration across the three regions. Specifically, we consider $\kappa \in \{5, 25, 50, 100, 250, 500\}$ and $\varphi_c \in \{0.05, 0.1, 0.15, 0.2\}$, and compare the distribution of accuracy associated with each configuration across regions and environments, as shown in Figure 13. Finally, we select the configuration with the highest sum of 25th percentiles across all regions and environments, which we highlight in red and specify in the last columns of Table 4. In Figure 13, we also highlight in grey the default *train* configuration, and in yellow the best configurations when considering the three regions of each environment individually. Overall, there are no significant differences between the segments of the best configurations per environment and the globally selected one, which indicates consistency and low sensitivity of selection accuracy across environments for the selected configurations.

Figure 13 also reveals several interesting patterns. First, the *train* criterion is less sensitive to parameter changes than the cost-driven *validation* criterion, with segments across the three regions varying very little for different values of κ. However, some deviation is observed in the learning and convergence regions as κ increases, as expected due to the inclusion of more episodes generated by policies further away from the current one in the empirical estimation (see Equation 4). This deviation is more pronounced for the cost-driven *validation* criterion. In this case instead, the reason is not that more validation episodes lead to higher inaccuracies, but rather that a larger number of episodes increases the validation cost, so that for a fixed validation-to-learning cost threshold fewer policies are evaluated, with longer periods between validations that more easily overlook good policies.

D.2 Validation cost within *validation* criteria

Given a sequence of policies visited during a learning process, validation with the same number of episodes for different policies does not necessarily imply the same cost. This makes it difficult to define a common *validation* configuration that ensures a similar cost across all cases.

D.2.1 Irregular validation cost within sequences

Figure 14 shows the validation-to-learning cost ratio associated with each individual policy in a sequence for each environment. The last six rows of each plot illustrate these cost ratios for different numbers of episodes, with a color scale ranging from dark green for ratios close to 0, to lighter green and white as the ratio approaches 0.3, and from white to red as it reaches 1, with ratios clipped at 1. Thus, when the validation cost of a policy is less than 30% of the time spent on its learning interaction and update, it is shown in green (darker green meaning lower cost), whereas higher costs are shown in red (darker red meaning higher cost). In addition, the first two rows represent, in grayscale, the normalized ground truth and normalized average episode length of each policy throughout the sequence. Darker values indicate higher ground truth and longer episodes.

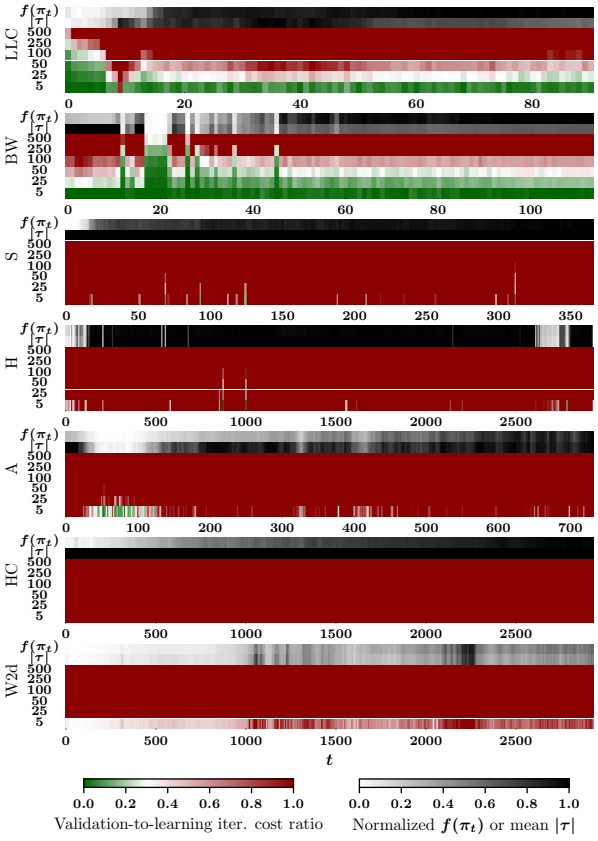

Figure 14: In each plot, the ground truth (first row), the average length of generated episodes (second row), and the validation-to-learning cost ratio per iteration (last six rows) for the policies in a sequence generated under each environment.

It can be observed that the rows are not homogeneous in color, meaning that validating different policies leads to different costs. Moreover, this pattern is not consistent across environments, suggesting that validation cost also varies across environments. Furthermore, the smaller the number of episodes used for validation, the lower the validation-to-learning cost ratio. Notice that the lowest rows show clearer and greener patterns.

It is worth noting that, across each sequence, for all environments except S and HC, the policies with lower cost ratios coincide with those with lower ground truth, which in turn correspond to those generating shorter episodes. These regions are typically located at the beginning of the sequence, indicating that worse policies are cheaper to validate because they generate shorter episodes. In contrast, for S and HC, all policies generate episodes of the same length, which makes validation cost more homogeneous. This is because in S and HC episode duration is fixed to 1000 interaction steps, whereas in the other environments episodes may terminate earlier upon reaching an unhealthy state, which is a terminal state from which no further interaction is possible, e.g., in A or W2d when the torso coordinates are very close to the ground, indicating that either the ant is upside down or the biped has fallen over, respectively.

D.2.2 Validation cost across different criteria configurations

This irregularity in validation cost across and within policy sequences is reflected in the distribution of validation costs for different configurations of the *validation* criterion. Figure 15 shows a boxplot of the validation-to-learning cost ratio across learning processes defined by each different configuration. It can be observed that, except for the simpler environments LLC and BW, most boxes lie above the cost ratio threshold of 0.2 illustrated by the green band. This highlights the difficulty of selecting a configuration with controlled validation cost. From the entire grid analysis of different possible configurations, it can be concluded that for a larger validation period $\varphi = 20$, the validation cost becomes more manageable, especially when combined with a small number of validation episodes such as $\kappa = 5$. However, increasing the validation period affects selection accuracy, which is illustrated by the blue boxes showing interquartile range and median accuracy. The case of BW is particularly notable.

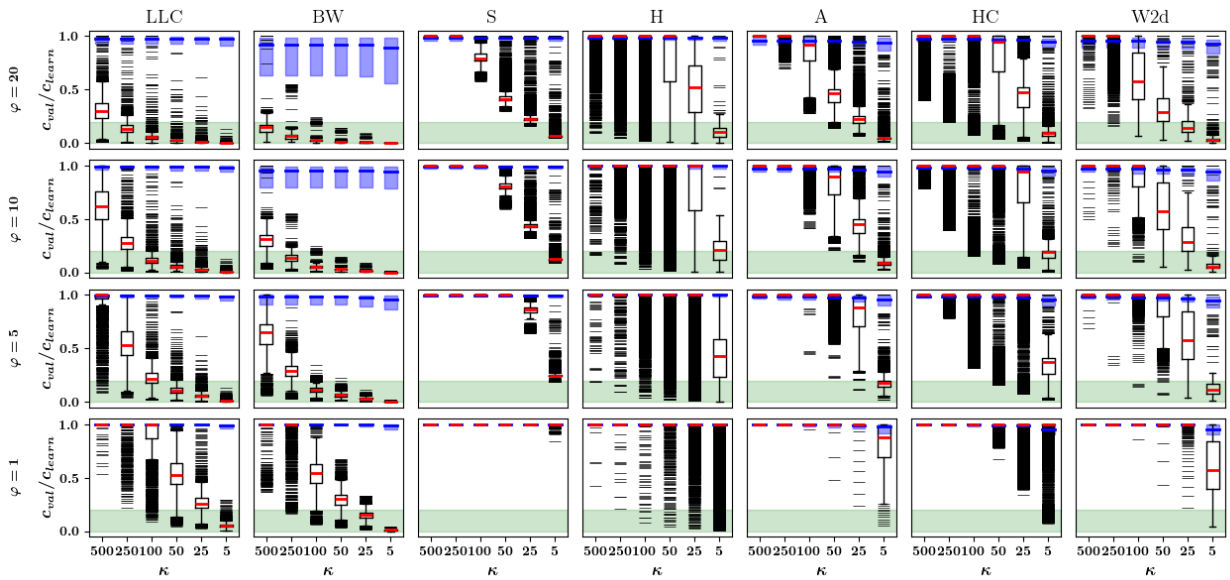

Figure 15: Boxplots of the validation-to-learning cost ratio for the default *validation* criterion across the 30 generated learning processes, considering different validation periods (columns) and numbers of validation episodes (columns within each plot), for each environment (columns). The green band highlights the admissible validation-to-learning cost ratio range between 0 and 0.2, while the blue boxes in the corresponding columns show the interquartile range and median of the selection accuracy.

Figure 16 shows the same analysis for the cost-driven *validation* criterion. In this case, validation cost is much better controlled, with most boxplots lying within the green range delimited by the threshold φ_c. When surpasses occur, they correspond to cases with a large number of validation episodes κ. This leads to longer validation periods, making the validation cost estimator (denoted by $\widetilde{c}$ in Algorithm 5) less accurate due to larger differences between validated policies, and consequently resulting in higher-than-expected validation costs. In addition, the distribution of selection accuracies for a small number of validation episodes $\kappa = 5$ is more stable across environments than in Figure 15.

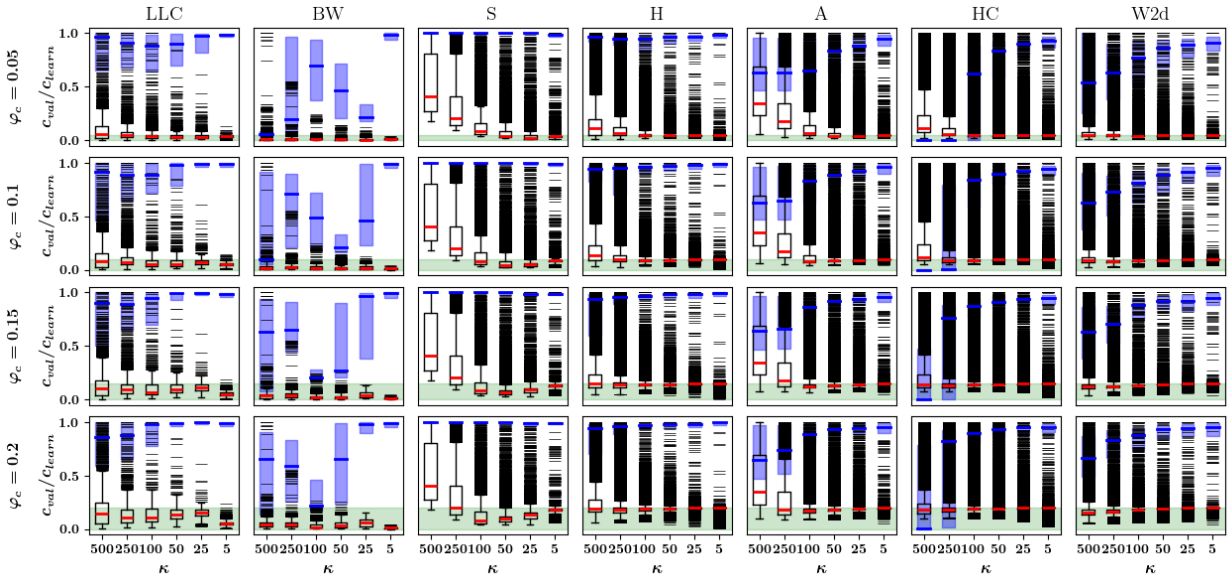

Figure 16: Boxplots of the validation-to-learning cost ratio for the cost-driven *validation* criterion across the 30 generated learning processes, considering different cost thresholds (rows) and numbers of validation episodes (columns within each plot), for each environment (rows). The green band highlights the range delimited by the corresponding cost threshold, while the blue boxes in the matching columns show the interquartile range and median of the selection accuracy.

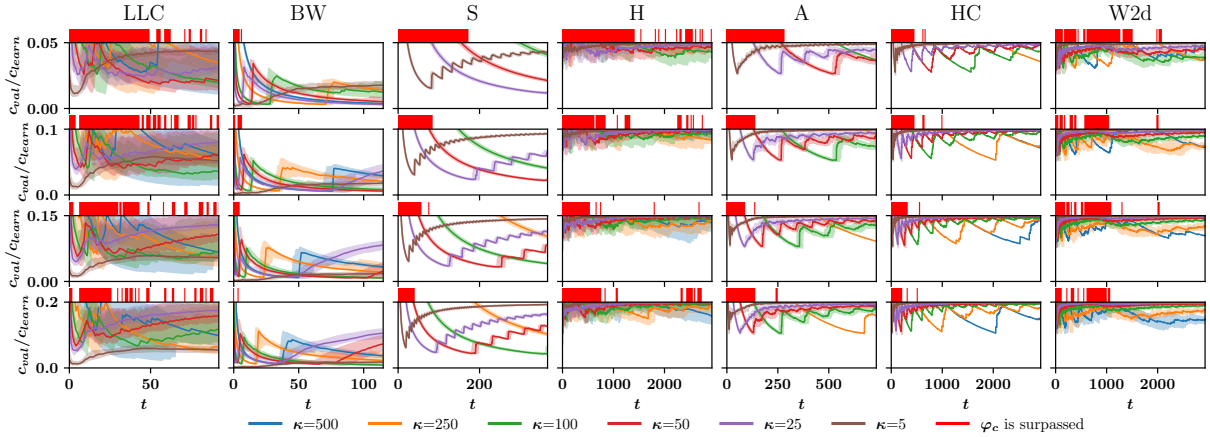

Figure 17: Evolution of the validation-to-learning cost ratio for the cost-driven *validation* criterion across the 30 learning processes, considering different cost thresholds (rows) and numbers of validation episodes (different curves), for each environment (columns). Red markers above each plot indicate iterations in which at least one exceedance of the predefined validation-to-learning cost threshold occurred.

Taking a closer look at the cost-driven *validation* criterion, Figure 17 explicitly shows the evolution of the validation-to-learning cost ratio across iterations for different configurations. Above each plot, red markers indicate iterations where the cost threshold is exceeded. Most of them occur at the beginning of training,

where the cost estimator is less accurate due to limited validation observations. However, these cases are outliers, as shown by the boxplots in Figure 16, whose interquartile range is in most cases fully contained within the threshold green band.

D.2.3 Benefits of adaptive validation period

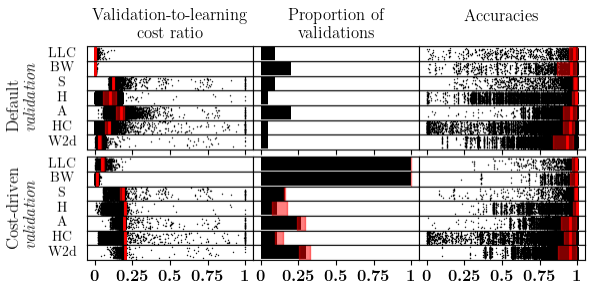

Figure 18: In the first and last columns, strip plots of validation-to-learning cost ratios truncated at 1 and accuracies across the 30 policy sequences per environment, together with the interquartile range and median in red. In the central column, the median percentage of validated policies, together with the interquartile range in red.

Up to this point, we have shown that the proposed cost-driven *validation* criterion allows, for different values of κ, a controlled cost under a fixed φ_c. However, this comes at the expense of a validation frequency that becomes variable not only across environments but also across individual processes within an environment. Figure 19 shows validation patterns across the 30 sequences used per environment under the selected configuration. Each line corresponds to a seed, where black marks indicate validation iterations. In simpler cases such as LLC and BW, almost every policy is validated, whereas in more complex environments validation becomes more sparse and irregular. These patterns also vary across different sequences within the same environment.

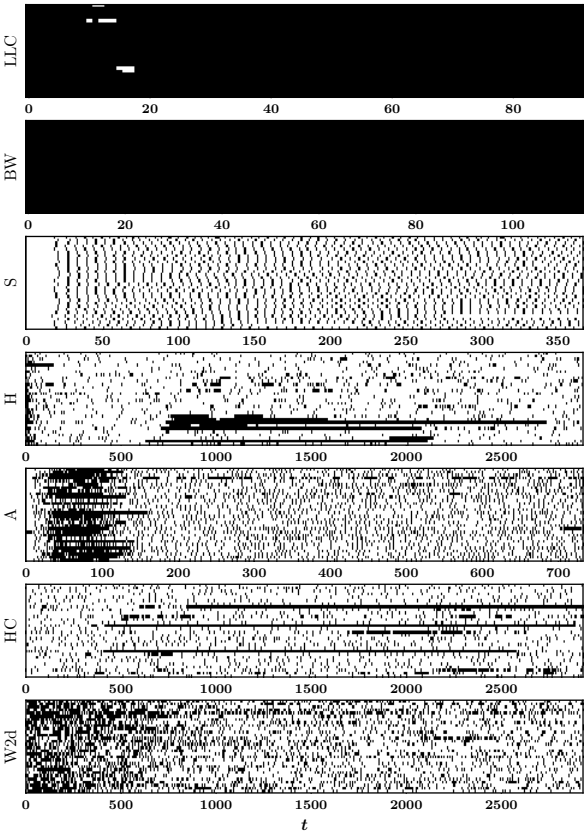

Figure 19: Validated policies (marked in black) throughout the 30 policy sequences (one row per sequence in each plot) using the cost-driven *validation* criterion across the different environments.

The adaptive validation exhibits certain advantages compared with the fixed validation period of the default *validation* criterion. In relatively simple environments with lower overall interaction budgets, such as LLC and BW, the default validation period results in too few policy evaluations during training, leading to a decrease in best-policy tracking accuracy. Specifically, high-performing policies are overlooked, despite the fact that validating all policies would not have incurred a substantial computational overhead (compare the first two subrows associated with both the default and cost-driven *validation* criteria in Figure 18). In contrast, accuracy in these cases is improved by the proposed cost-driven *validation* criterion, which adapts the validation period according to a predefined validation-to-learning cost ratio.

In summary, the cost-driven *validation* criterion proposed in this work defines the validation frequency such that, for a fixed configuration, it yields a similar maximum validation cost across processes. However, for fixed κ and a validation period controlled by a cost threshold φ_c, the resulting *validation* criterion induces irregular validation frequencies both within and across processes. However, a similar effect is observed in the default *validation* criterion, where the frequency is fixed in terms of interaction steps (see the fourth column

Table 4). This leads to different numbers of evaluations across environments due to varying interaction lengths (Chan et al., 2020), although without preserving any frequency-related invariance across processes, unlike our cost-driven *validation* criterion.

E Regions of learning processes

To conduct the experimental analysis in Section 5, we use a total of 30 sequence of policies per environment, each with a fixed but environment-dependent length. However, for simplicity, the results are presented in a compact form, summarizing seed variability using statistics such as the interquartile range and characterizing temporal variability of the metrics across three regions. In this appendix, we explicitly show in Figure 20 the characterization of the regions (see Equation 12) defined over individual sequence per environment that are internally considered in our analysis, together with the evolution of the associated degradation (see Equation 9) for each case.

Table 5: Mean and standard deviation of the percentage of the learning start and convergence start with respect to the total execution horizon for each environment, considering the 30 policy sequences generated in each case.

$\mathcal{E}$	$\frac{a}{T} \cdot 100$	$\frac{b}{T} \cdot 100$
LLC	9.78 ± 0.00	85.33 ± 9.20
BW	14.49 ± 8.28	70.52 ± 21.48
S	9.81 ± 0.00	70.02 ± 23.01
H	12.3 ± 5.01	31.48 ± 23.04
A	16.58 ± 2.50	79.69 ± 10.58
HC	10.79 ± 3.62	68.44 ± 19.74
W2d	10.00 ± 0.00	77.09 ± 16.24

Figure 20 shows only 5 seeds per environment, which are sufficient to illustrate how the region definition adapts to different scenarios to correctly characterize the three phases of initialization, learning, and convergence. For instance, while in LLC learning processes are relatively stable with generally low degradation (lighter top bars) and small initialization and convergence regions, in BW we observe more unstable sequences with higher degradation (darker top bars), where the region characterization adapts to each case. In particular, the learning-start limit a tends to occur later when there is higher initial instability (last row), while the convergence-start limit b appears earlier when reward improvement stagnates (third row). Table 5 summarizes the position of the limits a and b for each environment in numerical form, considering all 30 seeds.

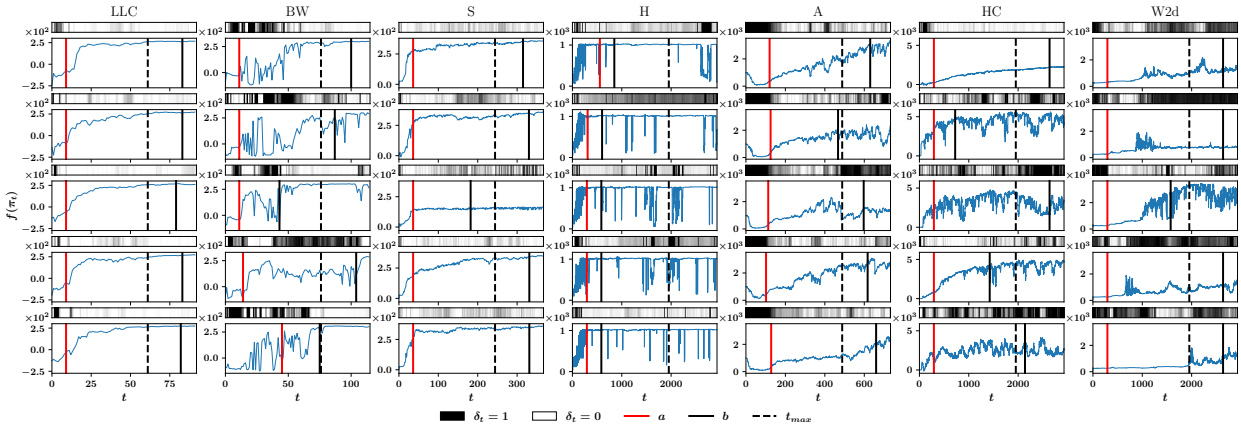

Figure 20: Evolution of the ground truth (blue curve) across the generated sequences for each enviroment (by column) and for 5 different seeds (by row), together with the start of learning (red line) and convergence (black line), which delimit the three regions. Above each curve, the evolution of degradation is shown along the sequence, where darker values indicate higher degradation.

F Selection criteria in early-stopping and resource-allocation techniques

In some practical settings, the objective is to obtain the best possible policy for a given task under a fixed computational budget. Such a budget may be large enough to allow multiple independent learning processes to be executed (Echevarrieta et al., 2026), for example by running the same method with different random

seeds. However, the learning curves generated by the same method-environment pair can vary substantially across seeds (Cheng et al., 2019; Henderson et al., 2018), with some runs leading to highly successful policies while others stagnate at relatively poor reward levels.

Obtaining a sufficiently large set of learning processes to reliably estimate the best achievable performance of a method can be computationally expensive (Jordan et al., 2024; Agarwal et al., 2021). In many situations, the available budget may not be sufficient to fully execute all candidate processes. For this reason, early-stopping and adaptive resource-allocation techniques (Li et al., 2018; Bohdal et al., 2023) are particularly valuable, as they discarded early the poorly performing processes, allowing more computational resources to be allocated to more promising ones. A classical approach is successive halving (Jamieson & Talwalkar, 2016), which iteratively discards the worst-performing half of the candidate processes to double the resources allocated to the remaining ones. This procedure is repeated until only a single process remains. Since these decisions are based on comparisons between learning curves, the selection criterion used to represent each learning process may influence the resulting resource-allocation decisions.

In this appendix, we provide a preliminary study of the influence of selection criteria on successive halving. Figure 21 illustrates the application of successive halving to the 30 policy sequences generated for each environment, considering different selection criteria for constructing the corresponding learning curves.

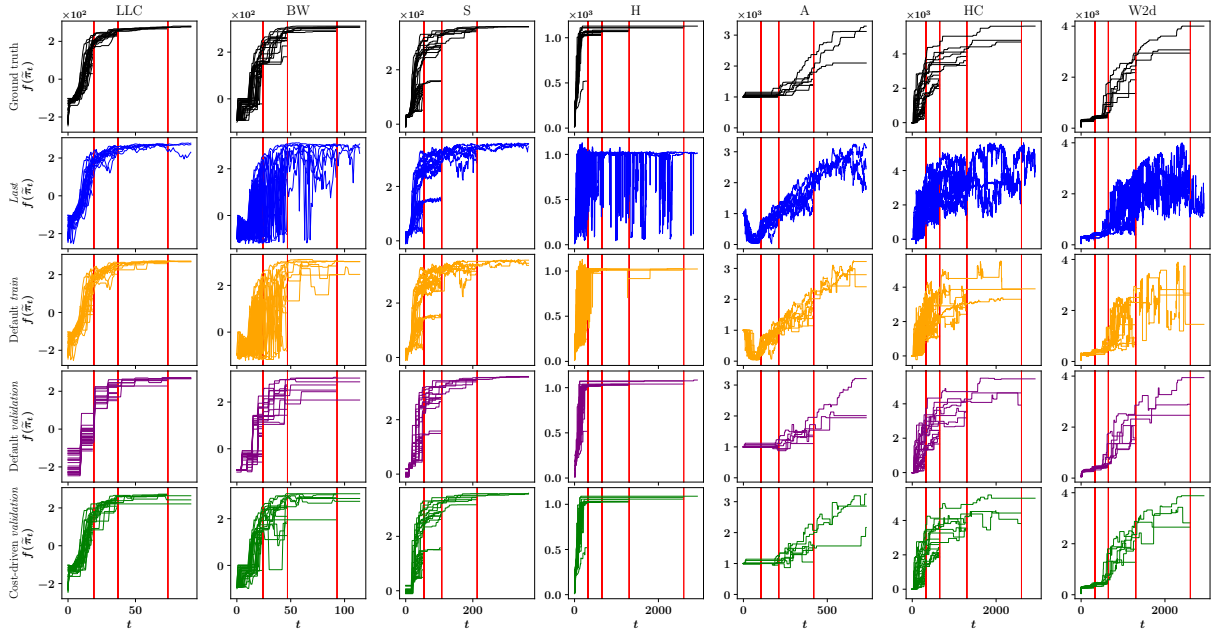

Figure 21: For each environment (columns) and selection criterion (rows), application of successive halving to the 30 policy sequences represented as learning curves. Each curve shows the ground-truth reward of the best policy identified by the corresponding selection criterion. Red vertical lines indicate successive elimination stages, where half of the surviving processes are discarded according to the estimated reward of their currently selected best policies[4].

F.1 Inefficiencies in resource allocation

Figure 21 reveals differences in the behaviour of successive halving depending on the selection criterion used to represent the learning curves. In particular, the curves associated with the *last* and *train* criteria (second and third rows) tend to occupy lower reward levels at later iterations, leaving a larger portion of the area

[4]For the *last* criterion, which does not rely on reward estimates, elimination decisions are based on the default *train* estimator applied to the last policy of each surviving sequences, using the most recent $\kappa = 100$ episodes generated during learning.

below the diagonal of each plot more densely covered. This suggests that these criteria may spend a larger fraction of the available budget evaluating policy sequences that ultimately achieve relatively poor reward.

To further analyze this observation, Figure 22 shows the distribution of computational resources allocated across different reward levels. The observed reward range is divided into three regions, using 25% and 75% of the maximum observed reward as thresholds. We then compute, for each selection criterion, the distribution of evaluations assigned to policies within each reward region and visualize the result using separate histograms. We observe that, the distributions most closely matching the ideal resource allocation correspond to the cost-driven *validation* criterion, where the green bars closely follow the reference distribution shown in black. In contrast, the *last* and *train* criteria exhibit the largest deviations. In these cases, the blue and orange bars tend to exceed the reference in low-reward levels, while falling below it in high-reward levels. This suggests that the *last* and *train* criteria allocate more resources to poorly performing policies and fewer resources to high-performing ones, resulting in a less efficient allocation of computational resources.

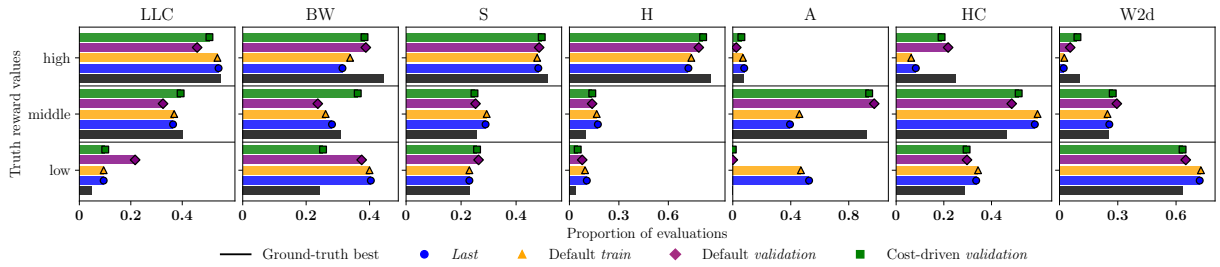

Figure 22: Distribution of evaluations allocated by successive halving across different policy reward levels for each environment and selection criterion. The observed reward range is partitioned into three regions (low, middle and high) using 25% and 75% of the maximum observed reward as thresholds. Histograms show the proportion of evaluations assigned to policies within each reward range. The black bars correspond to the ideal resource-allocation distribution obtained using ground-truth policy reward, while the coloured bars show the distributions induced by each selection criterion. Deviations from the reference distribution indicate differences in how computational resources are allocated across low, medium, and high performing policies.

F.2 Inaccuracies in best-policy tracking

The change induced by selection criteria in the allocation of resources across the 30 processes can also affect the tracking of the best policy among processes. During the application of successive halving, at each execution time we have as many candidate best policies as surviving processes. Each of these policies is assigned a reward estimate by the selection criterion, from which the policy with the highest estimated reward is selected as the best policy learned up to that moment. By comparing the reward of this policy with the true best policy, a tracking accuracy for early-stopping techniques applied to a set of processes under a given selection criterion can be defined, similar to how we define selection accuracy over a single learning sequence. For the preliminary experiment, we propose as a definition of tracking accuracy for a criterion under successive halving the following adaptation of Equation 10

$$\alpha_t^{SH} = \frac{\min\left\{ f\left(\arg\max_{\rho,\widetilde{\pi}_t} \widetilde{f}(\widetilde{\pi}_t) \right) - \min_{0,t},\ \max_{\rho,\pi_t^*} f(\pi_t^*) - \min_{0,t} \right\}}{\max_{\rho,\pi_t^*} f(\pi_t^*) - \min_{0,t}} \tag{13}$$

where

$$\min_{0,t} = \min\left\{ \min_{\rho} \left\{ f(\widetilde{\pi}_0), \ldots, f(\widetilde{\pi}_t) \right\}, \min_{\rho} \left\{ f(\pi_0^*), \ldots, f(\pi_t^*) \right\} \right\}$$

is the minimum reward across all learning curves defined by the selection criterion and by the ground truth. In the numerator, the minimum prevents the accuracy from exceeding 1, since when applying successive

halving with a given criterion, it may happen that sequences that are discarded early under the ground truth are not discarded, but eventually lead to higher rewards.

In Figure 23, we show, for each environment and criterion pair, the distribution of tracking accuracy using the interquartile range represented by the segment and the median indicated by the marker. We observe that the tracking accuracy of the *last* and *train* criteria is lower in more complex environments such as HC and W2d, with median values below 0.75 in the latter case. In contrast, validation-based criteria achieve higher tracking accuracy, with narrow interquartile ranges and values close to 1 in these environments. It is worth noting that the cost-driven *validation* criterion maintains consistently high tracking accuracy across environments, unlike default *validation*, which shows drops in accuracy in certain environments such as BW and A, with values close to 0.75. These observations suggest that the cost-driven *validation* criterion could accurately track the best solution in successive halving processes.

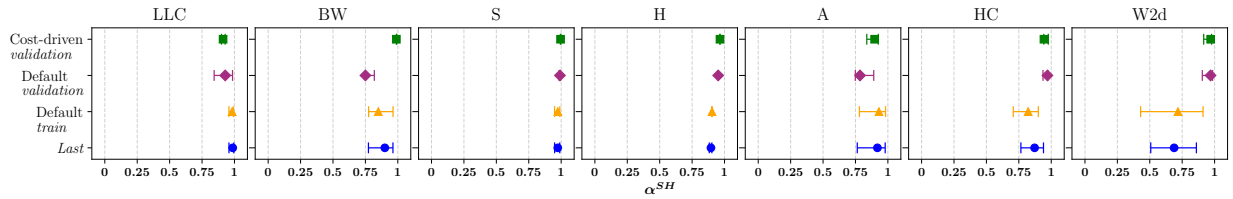

Figure 23: Tracking accuracy of successive halving under different selection criteria, computed across 30 policy sequences per environment. For each environment (columns) and selection criterion (rows), segments show the interquartile range and markers indicate median of the tracking accuracy defined in Equation 13.

F.3 Finding trade-off validation-to-learning cost ratio threshold

The cost-driven *validation* criterion has shown strong results across the different analyses conducted in this paper. In particular, it has demonstrated consistent accuracy and efficiency across environments and training regions that induce policy sequences with varying levels of degradation. However, this criterion depends on the parameters κ and φ_c, which determine the number of validation episodes and the validation-to-learning cost threshold used to adapt the validation frequency during learning. We have conducted our analyses using the configuration $\kappa = 5$ and $\varphi_c = 0.2$, under which we generally observe high selection accuracy across environments and regions (see Appendix D.1). As a final experiment, in Figure 24, we analyze the influence of the parameter φ_c on the trade-off between efficient resource allocation in successive halving and accurate tracking of the best policy.

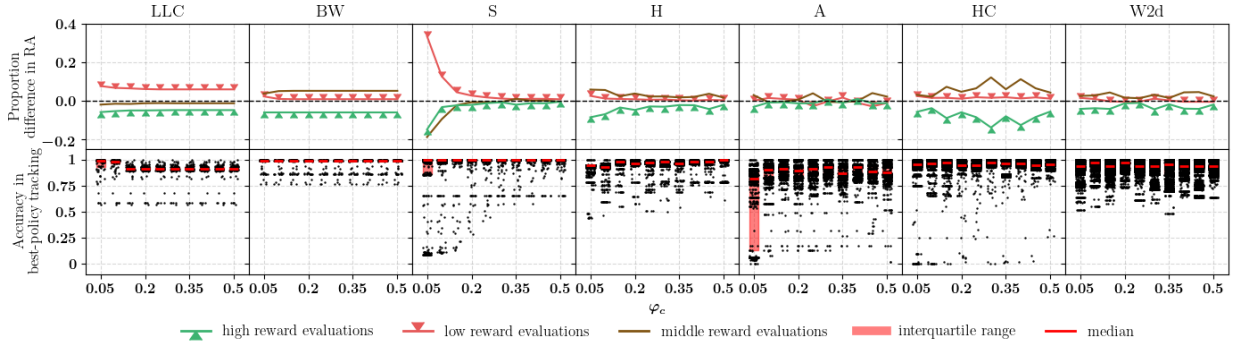

Figure 24: Trade-off analysis between Resource Allocation (RA) efficiency and best-policy tracking accuracy as a function of the cost threshold φ_c. In the top row, differences in the distribution of evaluations across low, medium, and high reward ranges after applying successive halving, measured relative to the ground-truth allocation. In the bottom row, the distribution of tracking accuracy for each value of φ_c, showing interquartile range and median in red.

Concretely, in the first row of Figure 24 we show, for each value of φ_c, the percentage difference in evaluations allocated to high, medium, and low reward ranges after applying successive halving. These values illustrate the deviations between the green and black bars in Figure 22. Then, in the second row, for each value of φ_c, a strip plot shows the tracking accuracies obtained using Equation 13, together with the interquartile range and median in red.

In general, we observe that for different values of φ_c, fewer evaluations tend to be allocated to high-reward regions (the green curve lies below the horizontal zero line), while more evaluations are allocated to low-reward levels (the pink curve lies above it), consistent with the general trends observed in Figure 22. However, in most cases these curves remain close to the horizontal baseline, suggesting that the deviations from the ground-truth resource allocation are not large. Additionally, although no strong trend is observed across values of φ_c in most environments, the case of S is notable. In this environment, for values of φ_c below 0.2, resource allocation becomes significantly less efficient, while inaccuracies in best-policy tracking also begin to appear. The pink and green curves diverge, while cases with tracking accuracies below 0.25 become more frequent. This suggests that, in this setting, using a threshold above 0.2 does not provide significant gains in efficiency or accuracy, while smaller values may lead to inefficiencies and inaccuracies. Therefore, the value $\varphi_c = 0.2$ appears to define a good trade-off. However, in other settings where results are less sensitive to φ_c, a smaller value could achieve similar accuracy and efficiency while reducing validation cost, leaving more budget available to execute additional processes.

G Construction of the figures for the experimental analysis

A method $\mathcal{M}$, an environment $\mathcal{E}$, a random seed ρ, and an execution time t determine a learning process, which we denoted as $\mathcal{M}(\mathcal{E}, \rho, t) \rightarrow \widetilde{\pi}_t$ (see Section 3 and Algorithm 1). During this process, a sequence of policies Π_t is generated and the policy $\widetilde{\pi}_t$ returned as learned by a selection criterion belongs to this sequence.

Our experimental analysis is based on a single on-policy Proximal Policy Optimization (Schulman et al., 2017) method and seven Gymnasium (Towers et al., 2025) environments $\mathcal{E} \in \{\text{LLC}, \text{BW}, \text{S}, \text{H}, \text{A}, \text{HC}, \text{W2d}\}$, as we described in Section 4. For each environment, we generate 30 policy sequences by considering 30 different random seeds and a maximum execution time T. Each sequence of T policies is characterized through the three regions defined in Equation 12. This allows us to partition a complete sequence of T policies into subsequences of length t, where $t \in R$ beeing $R \in \{[0, a], [a, b], [b, T]\}$ one of the intervals defining the regions. More formally, given a sequence $\Pi_T = (\pi_0, ..., \pi_T)$, the region R have associated the following set of subsequences:

$$\Pi_{\text{sub}} = \{\Pi_t : t \in R\}. \tag{14}$$

The figures presented throughout Section 5 are constructed from the values of the proposed metrics computed over these subsequences. In this appendix, we complement the descriptions provided in Section 5 by explicitly detailing how each figure is constructed.

G.1 Figures based on the complete set of subsequences

The metrics considered throughout the analysis are degradation, selection accuracy, and selection efficiency, defined in Equations 9, 10 and 11, respectively. The corresponding samples used for each figure are described below:

First plot in Figure 6. Given an environment and a learning region, we consider the sample of degradation values associated with all subsequences whose lengths belong to the interval R, formally defined as

$$X_\delta = \bigcup_{\rho \in \{1,\ldots,30\}} \{\delta_t : t \in R\}. \tag{15}$$

This sample is used to construct a histogram with 10 bins over the degradation range $[0, 1]$. Instead of representing frequencies through bar heights, darker colors indicate a larger number of degradation observations within the corresponding interval. The sample median is additionally represented by a red vertical line.

Second and third plots in Figure 6. Given an environment, a learning region, and a selection criterion, we consider the samples of selection accuracy and selection efficiency associated with all subsequences whose lengths belong to the interval R, formally defined as

$$X_\alpha = \bigcup_{\rho \in \{1,\ldots,30\}} \{\alpha_t : t \in R\}, \tag{16}$$

$$X_\varepsilon = \bigcup_{\rho \in \{1,\ldots,30\}} \{\varepsilon_t : t \in R\}. \tag{17}$$

The distributions of accuracy and efficiency are represented through the interquartile range together with the median of these samples.

Figure 7. Given an environment, for each pair consisting of a selection criterion and a learning region, we represent the point $\big(\text{Median}(X_\delta), \text{Median}(X_\alpha)\big)$ to study the relationship between degradation and selection accuracy, and $\big(\text{Median}(X_\delta), \text{Median}(X_\varepsilon)\big)$ to study the relationship between degradation and selection efficiency.

G.2 Figures based on pairwise criterion comparisons

For the comparative analysis of selection criteria presented in Section 5, the previous samples are restricted to the subsequences for which one criterion outperforms another. Specifically, given an environment, a learning region, and two selection criteria, we focus on the subsequences $\Pi_t \in \Pi_{\text{sub}}$ such that t belongs to

$$T_{1,2} = \bigcup_{\rho \in \{1,\ldots,30\}} \left\{ t \in R : f\left(\widetilde{\pi}_t^1\right) > f\left(\widetilde{\pi}_t^2\right) \right\}, \tag{18}$$

where $\widetilde{\pi}_t^1$ and $\widetilde{\pi}_t^2$ denote the policies selected by the first and second criteria, respectively. The set $T_{1,2}$ therefore represents all subsequences for which the first criterion selects a policy with higher ground-truth reward than the second criterion.

The corresponding figures are constructed as follows:

Figure 8. Given an environment, two selection criteria, and a learning region, we compare the cardinals $|T_{1,2}|$ and $|T_{2,1}|$ with respect to the total number of subsequences, including in the total also ties.

Figure 9. Given an environment, two selection criteria, and a learning region, we consider the degradation samples associated with the subsequences for which each criterion outperforms the other:

$$X_\delta^1 = \bigcup_{\rho \in \{1,\ldots,30\}} \{\delta_t : t \in T_{1,2}\}, \tag{19}$$

$$X_\delta^2 = \bigcup_{\rho \in \{1,\ldots,30\}} \{\delta_t : t \in T_{2,1}\}. \tag{20}$$

These samples are used to construct two 10-bin histograms following the same procedure described for Figure 6. Finally, both histograms are jointly normalized using the maximum frequency observed across the two samples, and the normalized frequencies are represented through color intensity. Additionally, the median of each subset is shown with a marker.