AMPO: ADAPTIVE MINIMAX POLICY OPTIMIZATION FOR FEDERATED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In federated reinforcement learning (FRL), we aim to learn a shared policy that performs reliably across all local environments. We therefore focus on the worst-case return, rather than the average return optimized by conventional methods. Achieving this is challenging since the worst-case environment is induced by the learned policy itself and shifts during training. We formulate this goal as an adaptive minimax objective and address the resulting non-stationarity through a saddle-point reformulation: we propose *adaptive minimax policy optimization* (AMPO), a primal–dual policy gradient method in which the dual variable tracks the shifting worst-case environment. Our analysis both guarantees convergence—to a stationary point with sublinear dual regret—and informs the algorithm design: it prescribes how fast the dual variable may be updated relative to the policy, and quantifies a robustness–scalability trade-off in which emphasizing difficult environments reduces the effective parallelism gained from federation. Experiments on heterogeneous MuJoCo benchmarks show that AMPO improves worst-case performance over average-return baselines while retaining competitive average performance.

1 INTRODUCTION

Reinforcement learning (RL) typically requires a large number of environment interactions, making *sample efficiency* a major bottleneck in practice. Federated reinforcement learning (FRL) has emerged as a promising paradigm to address this challenge by combining RL with federated learning (FL) (Qi et al., 2021; Khodadadian et al., 2022). A fundamental challenge in FRL is *environment heterogeneity* (Xie & Song, 2023; Jin et al., 2022; Wang et al., 2024; Zhang et al., 2024; Zhu & Gong, 2025; Labbi et al., 2025). Unlike standard RL, which optimizes a policy within a single environment, FRL requires learning a single policy that must operate across environments with distinct transition dynamics. This creates a structural trade-off: improving performance in one environment may degrade performance in others. More fundamentally, the state–action visitation distribution is determined by the policy itself, making the heterogeneity inherently *policy-dependent*—distinguishing FRL not only from standard RL, but also from conventional FL, where heterogeneity is fixed and independent of the learned model.

Existing FRL approaches typically optimize the *average return* across environments, inheriting the empirical risk minimization (ERM) paradigm of FL. We observe, however, that this objective corresponds to a *virtual* environment in which the deployment environment is drawn at random at the start of each episode, so that performance is measured only in expectation over this draw. A deployed policy, by contrast, acts in one fixed local environment and is judged by its performance there alone. Consequently, a policy that performs well on average may perform poorly in individual environments, especially under strong heterogeneity.

We therefore focus on the *worst-case* return across environments—directly targeting the deployment requirement that every local environment be served reliably. This is complicated by a closed loop: the worst-case environment is itself induced by the learned policy, and therefore shifts as training proceeds. While classical robust RL also optimizes a worst case, it does so over a structured uncertainty set of transition dynamics (Iyengar, 2005; Wiesemann et al., 2013; Roy et al., 2017; Ho et al., 2022; Hwang & Hong, 2024). Regularity conditions such as rectangularity yield a robust Bellman equation, so the worst case is resolved within dynamic programming. Here, by contrast, the worst

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

case ranges over a finite collection of *real* environments for which no such rectangular structure is available; the performance-limiting environment must therefore be tracked explicitly during optimization. We cast this as an *adaptive* minimax problem and resolve the resulting non-stationarity through a saddle-point reformulation, in which a dual variable dynamically reweights environments to track the shifting worst case. This adaptivity, however, is not free: concentrating the dual weights on difficult environments reduces the variance-averaging benefit of federated aggregation, and our analysis makes this robustness–scalability trade-off explicit.

Our contributions are summarized as follows:

- We develop *adaptive minimax policy optimization* (AMPO), a primal–dual algorithm whose dual variable tracks the policy-dependent worst-case environment.
- We establish $O(M^{-1/3})$ convergence and sublinear dual regret over M communication rounds under a policy-dependent, non-stationary worst case, and derive a guideline for the relative update rates of the policy and the dual variable.
- We characterize a robustness–scalability trade-off inherent to adaptive reweighting: emphasizing worst-case environments reduces the effective parallelism across agents.
- We instantiate AMPO with a PPO-based implementation and show that it improves worst-case return by up to 80% over uniform aggregation across heterogeneous MuJoCo benchmarks, while retaining competitive average return.

2 Related Work

FRL enables multiple agents to collaboratively learn a shared policy without sharing raw trajectories (Qi et al., 2021; Nadiger et al., 2019; Khodadadian et al., 2022), but early methods generally assume identical or sufficiently similar environments across agents (Fan et al., 2021; Zheng et al., 2024; Lan et al., 2024). Under environment heterogeneity, the average-return objective admits no Bellman equation over the original state space, so value-based methods instead solve a *central* MDP obtained by averaging the local models, with per-agent suboptimality scaling with the degree of heterogeneity (Jin et al., 2022; Wang et al., 2023; Zhang et al., 2024). Policy-gradient methods avoid this detour, since the gradient of a weighted return decomposes across environments (Labbi et al., 2025)—a property our adaptive weighting relies on. In parallel, algorithmic advances mitigate heterogeneity via KL-based policy constraints (Xie & Song, 2023), momentum-based variance reduction (Wang et al., 2024), and single-loop actor–critic updates (Zhu & Gong, 2025). All of these methods, however, target aggregate performance and provide no guarantee for the worst-performing environment. Recent work seeks robustness via a fixed uncertainty set around local transition kernels (Hwang & Hong, 2025), in the spirit of classical robust RL. In contrast, we directly optimize the worst-case return over the K environments, where the performance-limiting environment is itself policy-dependent and shifts during training.

Beyond FRL, adaptive reweighting toward the worst-performing subgroup underlies distributionally robust optimization (DRO) (Sagawa et al., 2020) and agnostic federated learning (AFL) (Mohri et al., 2019), which share our minimax structure but operate on a static loss landscape where the worst group depends only on the model parameters. In FRL, however, the worst-case environment evolves as a consequence of the agent's own behavior rather than remaining fixed, since the state–action visitation distribution under each environment is itself induced by the policy. This closed-loop, policy-dependent non-stationarity has no counterpart in standard DRO or AFL and is the central challenge our saddle-point analysis addresses.

Our regret analysis differs from that of online FL (OFL) (Kwon et al., 2023; Hong & Chae, 2022; Sahu et al., 2026), where the loss sequence is exogenous—determined by external data streams or adversaries, independently of the model being learned. In our setting, the loss sequence faced by the dual variable is endogenous: it is generated by the policy under optimization and thus evolves with the primal updates. This coupling, together with the fact that returns are observed only through noisy Monte Carlo estimates, precludes a direct application of existing OFL regret analysis techniques.

3 AMPO: ADAPTIVE MINIMAX POLICY OPTIMIZATION

3.1 PROBLEM SETUP

We consider an FRL setting with K environments (one per agent) that share a common state–action space, reward function, and initial-state distribution ρ_0, but differ in their transition kernels P_k. The goal is to learn a *single* global policy π that performs reliably across all local environments. Rewards are uniformly bounded, $|r(s,a)| \le r_{\max}$, and each episode has horizon T. The discounted return of π under environment P_k is

$$J_k(\pi) := \mathbb{E}_{\pi,P_k}\left[\sum_{t=0}^{T-1} \gamma^t r(s_t,a_t)\right], \tag{1}$$

where $\gamma \in (0,1)$ is the discount factor, $s_0 \sim \rho_0$, $a_t \sim \pi(\cdot \mid s_t)$, and $s_{t+1} \sim P_k(\cdot \mid s_t, a_t)$.

Conventional FRL targets the average return $\frac{1}{K}\sum_{k=1}^{K} J_k(\pi)$ (Jin et al., 2022; Wang et al., 2024; Labbi et al., 2025), which, as discussed in Section 1, measures performance in expectation over a randomly drawn deployment environment. Such an objective does not directly control the performance of any particular local environment and, in particular, provides no guarantee on the worst-performing environment. We therefore consider the minimax objective

$$\max_{\pi} \min_{k\in[K]} J_k(\pi), \tag{2}$$

which directly targets worst-case performance across the K local environments. The main difficulty is that the minimizing environment $k^\star(\pi) = \arg\min_{k\in[K]} J_k(\pi)$ is policy-dependent and may shift as the policy evolves. Consequently, the environment that limits performance cannot be fixed in advance.

3.2 SADDLE-POINT REFORMULATION

The objective in equation 2 is non-smooth in π due to the inner minimization. Introducing an auxiliary variable v, we write the problem in hypograph form as

$$\max_{\pi,v} v \qquad \text{s.t.} \qquad J_k(\pi) \ge v, \quad k \in [K]. \tag{3}$$

Applying Lagrangian relaxation with multipliers $\lambda_k \ge 0$ gives

$$\mathcal{L}(\pi, v, \lambda) = \sum_{k=1}^{K} \lambda_k J_k(\pi) + \left(1 - \sum_{k=1}^{K} \lambda_k\right) v. \tag{4}$$

For the maximization over v to remain bounded, the multipliers must satisfy $\lambda \in \Delta_K := \{\lambda \in \mathbb{R}_+^K : \sum_k \lambda_k = 1\}$, yielding the saddle-point formulation

$$\min_{\lambda\in\Delta_K} \max_{\pi} \sum_{k=1}^{K} \lambda_k J_k(\pi). \tag{5}$$

The dual variable λ defines a distribution over environments: for fixed λ, the policy is optimized under the corresponding weighted objective, while minimizing over λ increases emphasis on environments with lower returns. By weak duality, equation 5 upper-bounds equation 2; equality need not hold in general, since the set of return vectors $(J_1(\pi), \ldots, J_K(\pi))$ attainable by a single shared policy is non-convex even in tabular settings. We therefore treat equation 5 as a tractable relaxation of equation 2.

3.3 POLICY GRADIENT FOR THE WEIGHTED OBJECTIVE

For a fixed $\lambda \in \Delta_K$, define the weighted objective

$$J_\lambda(\pi) := \sum_{k=1}^{K} \lambda_k J_k(\pi). \tag{6}$$

Because the shared policy does not observe which environment it faces, $J_\lambda(\pi)$ corresponds to a partially observed latent MDP on $\mathcal{S} \times [K]$, in which the environment index is drawn from λ at $t = 0$ and held fixed thereafter; in general, s_t alone is not Markov, and the problem admits no standard Bellman formulation over $\mathcal{S}$. Policy-gradient methods, by contrast, operate directly on environment-specific trajectories and allow the gradient of equation 6 to decompose across agents.

We consider a parameterized stochastic policy π_θ with $\theta \in \mathbb{R}^d$.

Proposition 1 (Policy Gradient of the Weighted Objective)**.** *Let $\lambda \in \Delta_K$ be fixed, and let $\pi_\theta(a \mid s)$ be continuously differentiable with full support. Under standard regularity conditions,*

$$\nabla_\theta J_\lambda(\pi_\theta) = \sum_{k=1}^{K} \lambda_k \, \mathbb{E}_{\tau \sim (\pi_\theta, P_k)} \left[\sum_{t=0}^{T-1} \gamma^t \, \nabla_\theta \log \pi_\theta(a_t \mid s_t) \, G_t^{(k)} \right], \tag{7}$$

where $G_t^{(k)} := \sum_{t'=t}^{T-1} \gamma^{t'-t} r_{t'}$ is the return-to-go at time t under environment k.

Proposition 1 shows that the gradient of the weighted objective decomposes into a weighted sum of environment-wise policy gradients. This decomposition aligns naturally with the federated setting: each agent estimates a gradient using trajectories from its own dynamics P_k, while the server combines these estimates according to the adaptive weights λ. This observation provides the basis for AMPO.

3.4 AMPO: Adaptive Minimax Policy Optimization

AMPO alternates between local computation at the agents and global updates at the server.

Local computation (agents). At iteration m, the server broadcasts the current policy parameter $\theta^{(m)}$ to all agents. Each agent k collects trajectories $\mathcal{D}_k^{(m)} = \{\tau_{k,1}, \ldots, \tau_{k,N}\}$ with $\tau_{k,i} \sim (\pi_{\theta^{(m)}}, P_k)$ and computes local gradient and return estimates

$$g_k^{(m)} := \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T-1} \gamma^t \, \nabla_\theta \log \pi_{\theta^{(m)}}(a_t^{k,i} \mid s_t^{k,i}) \, \hat{G}_t^{(k,i)}, \qquad J_k^{(m)} := \frac{1}{N} \sum_{i=1}^{N} R(\tau_{k,i}), \tag{8}$$

where $\hat{G}_t^{(k,i)}$ denotes the sample return-to-go from time step t and $R(\tau_{k,i}) := \sum_{t=0}^{T-1} \gamma^t r_t^{k,i}$ denotes the discounted return of trajectory $\tau_{k,i}$. Since the trajectories are sampled on-policy, $J_k^{(m)}$ is an unbiased estimate of $J_k(\pi_{\theta^{(m)}})$ by definition, and Proposition 1 implies that $g_k^{(m)}$ is an unbiased estimate of $\nabla_\theta J_k(\pi_{\theta^{(m)}})$. Each agent then transmits $\big(g_k^{(m)}, J_k^{(m)}\big)$ to the server.

This is the idealized update analyzed in Section 4; the PPO-based implementation in Section 5 likewise evaluates all local gradients at the broadcast parameter $\theta^{(m)}$, but replaces the return-to-go with GAE advantages over multiple minibatches.

Global update (server). Upon receiving $\{g_k^{(m)}, J_k^{(m)}\}_{k=1}^K$, the server aggregates the local policy gradients using the current dual weights and performs gradient ascent,

$$g^{(m)} = \sum_{k=1}^{K} \lambda_k^{(m)} g_k^{(m)}, \qquad \theta^{(m+1)} = \theta^{(m)} + \eta_\pi g^{(m)}. \tag{9}$$

The server then updates the environment weights by projected gradient descent,

$$\lambda^{(m+1)} = \Pi_{\Delta_K} \left(\lambda^{(m)} - \eta_\lambda \big[J_1^{(m)}, \ldots, J_K^{(m)} \big]^\top \right), \tag{10}$$

where Π_{Δ_K} denotes Euclidean projection onto the probability simplex. The weights λ are optimization variables of the saddle-point problem in equation 5, rather than fixed aggregation coefficients. Equations 9 and 10 therefore correspond to the primal and dual updates of AMPO, respectively.

4 Theoretical Analysis of AMPO

We establish the theoretical guarantees of AMPO: convergence of the primal variable and no-regret of the dual variable (Section 4.1), followed by a characterization of how adaptive reweighting affects the scalability of federated policy optimization (Section 4.2). All proofs are deferred to the appendix.

4.1 Convergence and Regret Analysis

We analyze AMPO from the perspective of stochastic saddle-point optimization, in which the algorithm performs stochastic gradient ascent on θ and projected gradient descent on λ. For notational convenience, we write the objective in equation 5 as $F(\theta, \lambda) := \sum_{k=1}^{K} \lambda_k J_k(\pi_\theta)$ for $\lambda \in \Delta_K$, and let $\mathcal{H}_m$ denote the filtration containing all randomness up to the beginning of iteration m.

Assumption 1 (Smoothness)**.** *Each $J_k(\pi_\theta)$ is L-smooth in θ.*

Assumption 2 (Unbiased Gradient with Bounded Variance)**.** *The policy gradient estimator $g^{(m)}$ is conditionally unbiased with bounded variance: $\mathbb{E}[g^{(m)} \mid \mathcal{H}_m] = \nabla_\theta F(\theta^{(m)}, \lambda^{(m)})$ and $\mathbb{E}[\|g^{(m)} - \nabla_\theta F(\theta^{(m)}, \lambda^{(m)})\|^2 \mid \mathcal{H}_m] \le \sigma^2$.*

Assumption 3 (Unbiased and Uniformly Bounded Return Estimator)**.** *The return estimator $J_k^{(m)}$ is conditionally unbiased, $\mathbb{E}[J_k^{(m)} \mid \mathcal{H}_m] = J_k(\pi_{\theta^{(m)}})$, and there exists a constant $J_{\max} > 0$ such that $|J_k^{(m)}| \le J_{\max}$ and $|J_k(\pi_\theta)| \le J_{\max}$ almost surely, for all $k \in [K]$, $m \ge 0$, and θ.*

Assumptions 1 and 2 are standard in the analysis of stochastic policy optimization (Labbi et al., 2025; Wang et al., 2024). The unbiasedness parts of Assumptions 2 and 3 hold by construction: $F(\theta, \lambda)$ is a convex combination of $\{J_k(\pi_\theta)\}_{k=1}^K$, and since $\lambda^{(m)}$ is $\mathcal{H}_m$-measurable, the aggregated gradient inherits conditional unbiasedness from the local estimators in equation 8 by Proposition 1; F likewise inherits L-smoothness uniformly over $\lambda \in \Delta_K$. Assumption 3 strengthens the usual bounded-second-moment condition to almost-sure boundedness, as required by the martingale concentration argument underlying the high-probability guarantee in Theorem 1; this is essentially without loss of generality, since bounded rewards and a finite horizon imply $|J_k^{(m)}| \le r_{\max}(1-\gamma^T)/(1-\gamma)$.

Theorem 1 (Primal Convergence and Dual No-Regret)**.** *Suppose Assumptions 1–3 hold. Choose the step sizes as $\eta_\pi = c_\pi M^{-1/3}$ and $\eta_\lambda = c_\lambda M^{-2/3}$ for constants $c_\pi, c_\lambda > 0$ with $\eta_\pi \le 1/L$. Then the iterates $\{(\theta^{(m)}, \lambda^{(m)})\}_{m=0}^{M-1}$ generated by AMPO satisfy the following.*

(i) Primal convergence:

$$\frac{1}{M}\sum_{m=0}^{M-1} \mathbb{E}\Big[\big\|\nabla_\theta F(\theta^{(m)}, \lambda^{(m)})\big\|^2\Big] = \mathcal{O}\big(M^{-1/3}\big).$$

(ii) Dual no-regret: For any $\delta \in (0,1)$, with probability at least $1-\delta$,

$$\sum_{m=0}^{M-1} F(\theta^{(m)}, \lambda^{(m)}) - \min_{\lambda \in \Delta_K} \sum_{m=0}^{M-1} F(\theta^{(m)}, \lambda) \;\le\; \mathcal{O}\big(M^{2/3} + J_{\max}\sqrt{M\log(K/\delta)}\big) = \mathcal{O}\big(M^{2/3}\big).$$

Result (i) shows that the policy converges to a first-order stationary point of the dynamically reweighted objective $F(\theta, \lambda^{(m)})$: although this objective is non-stationary as the dual weights evolve, the time-averaged gradient norm vanishes. Result (ii) establishes that the adaptive weights are nearly as adversarial as the best fixed weighting in hindsight, which places all mass on the environment with the lowest cumulative return; the dual variable thus tracks the performance-limiting environment even as it shifts. Since (ii) holds with high probability rather than in expectation, it controls the regret of individual training runs. The rate $\mathcal{O}(M^{2/3})$, slower than the classical $\mathcal{O}(\sqrt{M})$ of online convex optimization, reflects a two-timescale trade-off: the step sizes prioritize the primal rate, and choosing a larger η_λ would improve the dual regret at the expense of slower primal convergence. Together, (i) and (ii) characterize the reweighted objective $F(\theta, \lambda^{(m)})$; relating it to the original minimax objective in equation 2 requires additional structure in the non-convex setting.

Practical choice of learning rates. Theorem 1 requires $\eta_\lambda/\eta_\pi \to 0$: if η_λ is too large, the weights chase noisy return estimates and the worst-case environment oscillates; if too small, λ remains close to its initialization and, from a uniform start, AMPO reduces to uniform aggregation. We thus treat η_λ/η_π as the primary hyperparameter.

4.2 Robustness–Scalability Trade-off

Theorem 1 was established under a conservative, iteration-independent bound σ^2 on the gradient noise; we now sharpen its primal rate by exploiting the variance-reduction effect of weighted aggre-

gation. In standard FRL methods that optimize an average-return objective, local gradients are uniformly averaged across K agents, achieving variance reduction proportional to $1/K$ and enabling linear speedup in K. In contrast, AMPO aggregates gradients using adaptive weights $\lambda \in \Delta_K$, which changes the effective degree of parallelism.

Proposition 2 (Effective Parallelism under Adaptive Weighting)**.** *Suppose the local gradient estimators $\{g_k\}_{k=1}^K$ are conditionally independent given $\mathcal{H}_m$, and each is unbiased with per-agent variance bounded by σ_u^2. Then, for any $\lambda \in \Delta_K$,*

$$\mathbb{E}\Big[\big\|\textstyle\sum_{k=1}^K \lambda_k (g_k - \nabla_\theta J_k)\big\|^2 \,\Big|\, \mathcal{H}_m\Big] \;\le\; \tfrac{\sigma_u^2}{K_{\text{eff}}}, \qquad K_{\text{eff}} := \tfrac{1}{\sum_{k=1}^K \lambda_k^2} \in [1, K]. \tag{11}$$

The quantity K_{eff} characterizes the effective degree of variance reduction in the aggregated gradient: it equals K when λ is uniform, corresponding to classical linear speedup, and decreases toward 1 as λ concentrates on a subset of environments. Applying Proposition 2 at round m with $K_{\text{eff}}^{(m)} := 1/\sum_k (\lambda_k^{(m)})^2$ yields $\mathbb{E}[\|g^{(m)} - \nabla_\theta F_m\|^2 \mid \mathcal{H}_m] \le \sigma_u^2 / K_{\text{eff}}^{(m)} \le \sigma_u^2 =: \sigma^2$, where $F_m := F(\theta^{(m)}, \lambda^{(m)})$. The second inequality confirms that Assumption 2 holds with $\sigma^2 = \sigma_u^2$, while the first shows that the aggregated gradient retains variance reduction proportional to $K_{\text{eff}}^{(m)}$. Retaining this sharper bound yields the following refinement.

Corollary 1 (Effective Linear Speedup)**.** *Suppose Assumptions 1–3 and the conditions of Proposition 2 hold, and let $\bar{K}_{\text{eff}}$ denote the harmonic mean of $\{K_{\text{eff}}^{(m)}\}_{m=0}^{M-1}$. If there exists $\kappa \in [1, K]$ such that $\bar{K}_{\text{eff}} \ge \kappa$ almost surely, then choosing $\eta_\pi = \Theta(\kappa^{2/3} M^{-1/3})$ and $\eta_\lambda = \Theta(\kappa^{1/3} K^{-1} M^{-2/3})$, subject to $\eta_\pi \le 1/L$, gives*

$$\frac{1}{M} \sum_{m=0}^{M-1} \mathbb{E}\|\nabla_\theta F_m\|^2 = \mathcal{O}\big((\kappa M)^{-1/3}\big). \tag{12}$$

Consequently, achieving $\frac{1}{M}\sum_{m=0}^{M-1} \mathbb{E}\|\nabla_\theta F_m\|^2 \le \epsilon$ requires $M = \mathcal{O}(1/(\kappa\epsilon^3))$ communication rounds.

Corollary 1 establishes a linear speedup in the effective number κ of environments contributing to gradient-noise reduction. In the uniform case, $\kappa = K$, recovering linear speedup in the number of agents; if instead the dual weights concentrate on a single environment, then $\kappa = 1$ and equation 12 reduces to the single-environment rate $\mathcal{O}(M^{-1/3})$. This reveals a trade-off inherent to adaptive reweighting: emphasizing challenging environments improves robustness but reduces effective parallelism, so AMPO achieves only a *partial* speedup governed by κ. The ratio $K/\kappa \ge 1$ thus quantifies the price of robustness in FRL; see Section 5.2 for empirical values.

5 Practical Implementation of AMPO

We instantiate AMPO with a PPO-based actor–critic implementation (Schulman et al., 2017; Raffin et al., 2021). We adopt standard PPO components for trajectory collection, generalized advantage estimation (GAE), minibatch processing, and value-function learning. Unlike standard PPO, however, the shared actor is not updated locally across PPO minibatches. Instead, each agent uses its local rollout to construct an actor-gradient direction at the broadcast policy $\theta^{(m)}$, which is then aggregated and applied only at the server.

Local critics and global actor. AMPO-PPO maintains a single server-owned actor π_θ, while each agent k maintains an environment-specific critic V_{ψ_k} that remains local throughout training. This design is motivated by the environment heterogeneity in our formulation: since each critic estimates values under the agent-specific transition dynamics P_k, aggregating critic parameters across heterogeneous environments may produce a value estimator that is not well matched to any individual local environment. Maintaining local critics instead allows each agent's value and advantage estimates to reflect its own dynamics. This separation is also consistent with Proposition 1, which decomposes the global policy gradient into a weighted sum of environment-wise policy gradients without requiring a shared value function.

Local PPO computation. At communication round m, the server broadcasts $\theta^{(m)}$ to all agents, and agent k collects a local rollout under P_k. GAE advantages are computed from the collected rollout,

after which the rollout is processed over multiple PPO epochs and minibatches. For each minibatch, the agent updates only its local critic using the standard PPO value-function objective. For the actor, it computes a policy-gradient contribution at the fixed broadcast parameter $\theta^{(m)}$ without taking a local actor step:

$$\hat{g}_{k,b}^{(m)} = \widehat{\mathbb{E}}_{\mathcal{B}_{k,b}^{(m)}}\left[\nabla_\theta \log \pi_{\theta^{(m)}}(a_t \mid s_t)\, \hat{A}_{k,t}^{(m)}\right], \qquad g_k^{(m)} = \sum_{b=1}^{B_k^{(m)}} \hat{g}_{k,b}^{(m)}, \tag{13}$$

where $\hat{A}_{k,t}^{(m)}$ denotes the GAE advantage computed from the rollout collected under $\pi_{\theta^{(m)}}$ and $\mathcal{B}_{k,b}^{(m)}$ denotes the b-th minibatch across all epochs. The resulting $g_k^{(m)}$ is transmitted to the server, which applies the weighted actor update in equation 9.

Practical dual update. The practical dual update addresses two issues. First, since the PPO rollout length need not coincide with the episode horizon, rollout fragments would yield truncated returns; we therefore accumulate rewards across rollout and communication-round boundaries and use only completed episodic returns, computed without discounting to match the evaluation metric. Second, returns from the training stream are noisy and can destabilize the adaptive weights; we therefore smooth them with an exponential moving average, standardize them across agents, and update λ only every five communication rounds, i.e., on a slower timescale than the actor. Details are provided in Appendix E.2.

5.1 Experimental Setup

We evaluate AMPO on four MuJoCo continuous-control tasks from Gymnasium (Brockman et al., 2016; Towers et al., 2024). All methods use the same Stable-Baselines3 PPO backbone (Raffin et al., 2021) and hyperparameter configuration, thereby isolating the effect of the federated optimization strategy. Results are averaged over five training seeds, with each evaluation conducted over ten episodes; we report the mean and standard deviation across seeds (details in Appendix F).

We compare AMPO-PPO(A), which adaptively updates the dual weights, against three baselines: **PPOAvg**, a PPO instantiation of the averaging scheme of Jin et al. (2022), where agents perform local PPO updates that are averaged uniformly; **FedSVRPG-M-PPO**, which adds the momentum-based variance reduction of Wang et al. (2024) to mitigate client drift; and **AMPO-PPO(U)**, which fixes $\lambda_k = 1/K$. Unlike PPOAvg, AMPO-PPO(U) applies the actor update only at the server (Appendix E.3), so comparing it with AMPO-PPO(A) isolates the effect of adaptive weighting.

To construct heterogeneous environments, we vary a single physical parameter (gravity or friction) in each setting: environment k uses $q_k = q(1+\delta_k)$ with $\delta_k = -p + 2p(k-1)/(K-1)$, where q is the nominal value. We use $K = 5$ and $p = 0.3$, giving $\delta_k \in \{-0.3, -0.15, 0, 0.15, 0.3\}$. All other settings are unchanged, so the local MDPs share the same task and observation–action spaces but differ in their transition dynamics.

We report the *average local return* $J_{\text{avg}}(\pi) := \frac{1}{K}\sum_k J_k(\pi)$ and the *worst-case local return* $J_{\text{worst}}(\pi) := \min_k J_k(\pi)$, both computed from undiscounted episodic returns; the latter directly measures the robust objective of AMPO. Since the worst-performing environment is policy-dependent, the minimum is recomputed at each evaluation point.

5.2 Local Return Imbalance and Adaptive Weight Concentration

Table 1 examines how adaptive weighting responds to performance imbalance across heterogeneous local environments. For each environment–perturbation setting, we report the relative return spread (Rel. Spread) of AMPO-PPO(U) and AMPO-PPO(A), which quantifies the variation in local performance, together with the adaptive weights λ and the corresponding effective number of environments K_{eff}, where a smaller K_{eff} indicates stronger concentration of the aggregation weights. Full per-environment return profiles are given in Appendix F.3.

A consistent pattern appears in the five settings with relatively large return spreads under AMPO-PPO(U): Ant with friction and gravity, HalfCheetah with gravity, and Hopper with friction and gravity. In these settings, AMPO-PPO(A) reduces the spread while concentrating larger weights on lower-performing environments, yielding K_{eff} of 2.56–3.39 compared with 5 under uniform

Table 1: Local return imbalance and adaptive aggregation weights across heterogeneous MuJoCo environments. Rel. Spread measures the return variation across the five local environments under AMPO-PPO(U) and AMPO-PPO(A). The adaptive weights λ correspond to the environment-specific perturbation levels, and K_{eff} measures their concentration, with smaller values indicating stronger focusing on a subset of environments. AMPO-PPO(U) uses uniform weights and therefore has $K_{\text{eff}} = K = 5$ by construction. Statistics are averaged over the final ten evaluation points.

Env.	Perturbation	Rel. Spread (%)		Adaptive weights λ					K_{eff}
		AMPO-PPO(U)	AMPO-PPO(A)	-0.30	-0.15	0	$+0.15$	$+0.30$	
Ant	Friction	106.0	48.7	0.000	0.036	0.176	0.330	0.458	2.85
	Gravity	73.3	39.3	0.520	0.308	0.140	0.031	0.001	2.59
HalfCheetah	Friction	6.6	12.5	0.235	0.247	0.210	0.179	0.130	4.47
	Gravity	35.1	19.5	0.487	0.280	0.095	0.055	0.083	2.96
Hopper	Friction	76.5	51.9	0.445	0.081	0.002	0.044	0.428	2.56
	Gravity	32.8	15.0	0.246	0.099	0.120	0.174	0.362	3.39
Walker2d	Friction	8.7	7.4	0.348	0.222	0.140	0.127	0.164	3.95
	Gravity	18.9	16.7	0.273	0.163	0.087	0.115	0.362	3.81

weighting. These same settings also exhibit substantial worst-case improvements in Table 2, ranging from 20.6% to 80.4%, indicating that the reduction in local performance imbalance is accompanied by improved worst-case performance.

By contrast, in settings where the return profile is already relatively balanced under AMPO-PPO(U), AMPO-PPO(A) exhibits weaker weight concentration and only limited reductions in spread. This behavior is consistent with the modest worst-case improvement observed for Walker2d. HalfCheetah with friction is an exception: its spread under AMPO-PPO(U) is already small and the adaptive weights remain close to uniform, yet the entire return profile of AMPO-PPO(A) shifts upward, so the worst-case return improves substantially without a reduction in relative spread. This gain is therefore not attributable to weight concentration.

Table 2: Worst-case local return of AMPO-PPO(U) and AMPO-PPO(A) averaged over the final ten evaluation points.

Env.	Perturbation	AMPO-PPO(U)	AMPO-PPO(A)	Improvement
Ant	Friction	1826.3 ± 144.4	3295.5 ± 317.2	$+\mathbf{80.4}\%$
	Gravity	2334.5 ± 419.0	3021.1 ± 316.2	$+\mathbf{29.4}\%$
HalfCheetah	Friction	3496.4 ± 1740.7	4580.7 ± 731.7	$+\mathbf{31.0}\%$
	Gravity	3249.2 ± 788.7	3918.3 ± 736.2	$+\mathbf{20.6}\%$
Hopper	Friction	1248.4 ± 113.7	1636.8 ± 271.5	$+\mathbf{31.1}\%$
	Gravity	1911.5 ± 212.0	2479.1 ± 331.1	$+\mathbf{29.7}\%$
Walker2d	Friction	3097.5 ± 586.9	3393.7 ± 335.9	$+\mathbf{9.6}\%$
	Gravity	2978.4 ± 506.6	2930.0 ± 426.8	$-\mathbf{1.6}\%$

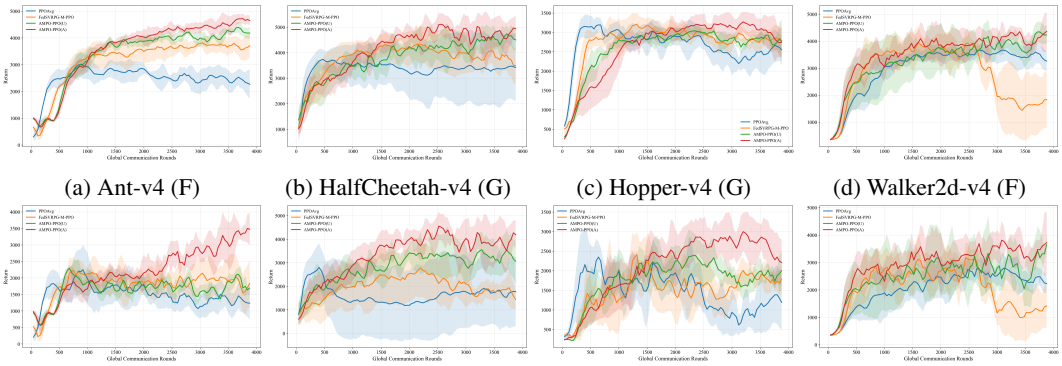
5.3 Policy-Dependent Worst-Environment Dynamics

Table 3 examines how the identity of the worst-performing environment evolves during training of AMPO-PPO(A). The *Worst Switch* rate is the fraction of consecutive evaluation points at which the worst-performing environment changes, and the *Dominant Worst Share* is the fraction of evaluation points occupied by the most frequently worst-performing environment. Worst Switch rates are nonzero in all settings and reach up to 0.66, consistent with our formulation in which the worst environment is policy-dependent and shifts during training.

Table 3: Worst-environment dynamics computed over all evaluation points.

Env.	Perturbation	Worst Switch	Dom. Worst Share
Ant	Friction	0.18 ± 0.03	0.86 ± 0.01
	Gravity	0.18 ± 0.08	0.87 ± 0.03
HalfCheetah	Friction	0.66 ± 0.05	0.38 ± 0.11
	Gravity	0.49 ± 0.10	0.57 ± 0.11
Hopper	Friction	0.38 ± 0.11	0.51 ± 0.05
	Gravity	0.38 ± 0.17	0.66 ± 0.18
Walker2d	Friction	0.66 ± 0.08	0.38 ± 0.07
	Gravity	0.65 ± 0.09	0.39 ± 0.13

In the same five settings where the previous analysis identifies substantial return imbalance, the worst environment is relatively persistent, as indicated by lower switch rates and higher dominant-worst shares than in the remaining settings. Combined with the stronger weight concentration observed in



(a) Ant-v4 (F) (b) HalfCheetah-v4 (G) (c) Hopper-v4 (G) (d) Walker2d-v4 (F)

(e) Ant-v4 (F) (f) HalfCheetah-v4 (G) (g) Hopper-v4 (G) (h) Walker2d-v4 (F)

Figure 1: **Average and worst-case local evaluation performance across heterogeneous MuJoCo environments.** The top row shows the average local return, while the bottom row shows the worst-case local return. Here, (F) and (G) denote friction and gravity perturbations, respectively.

Table 1, this suggests that AMPO-PPO(A) focuses on persistent performance-limiting environments when such environments emerge.

5.4 Comparison with Federated Baselines

Fig. 1 compares the average (top row) and worst-case (bottom row) local returns during training for all methods. In the settings shown, AMPO-PPO(A) achieves competitive average performance while delivering more pronounced gains in worst-case performance. Results for the remaining environment–perturbation settings are provided in Appendix G.

5.5 Dual Learning-Rate Sensitivity

As discussed in Section 4.1, the ratio η_λ/η_π governs how aggressively the dual weights react to the observed returns. We examine its effect by varying η_λ while holding η_π fixed. Fig. 2 reports the average worst-case return over the final ten evaluation points for different values of η_λ/η_π, with AMPO-PPO(U) included as a reference. The results show that excessively large ratios substantially degrade worst-case performance, indicating that overly aggressive dual updates can be detrimental to policy optimization. This behavior is consistent with our theoretical step-size design, which requires the dual adaptation to proceed more slowly than the policy update.

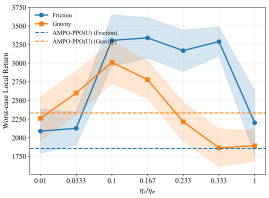


Figure 2: Sensitivity of worst-case local performance to the dual-to-policy learning-rate ratio η_λ/η_π on Ant environment.

6 Conclusion

We proposed a robust FRL framework that departs from the standard average-return objective by directly optimizing worst-case performance across heterogeneous environments. By reformulating the minimax objective as a saddle-point problem, we developed AMPO, an adaptive minimax policy optimization algorithm in which the dual variable dynamically reweights environments to track the policy-dependent worst-case environment. We established, in expectation, convergence of the policy to a first-order stationary point, and, with high probability, sublinear regret for the dual variable, together with a robustness–scalability trade-off characterizing the effective speedup of federated aggregation under adaptive reweighting. A PPO-based instantiation of AMPO demonstrated improved worst-case performance while maintaining competitive average performance across heterogeneous benchmark environments. Future directions include exploring alternative dual update rules, such as exponentiated-gradient methods over the simplex.

AI USE STATEMENT

We used generative AI tools to assist with polishing the writing and improving the clarity of the manuscript, and for retrieval and discovery of potentially relevant prior work. All AI-assisted edits were reviewed by the authors, and any retrieved or suggested references were checked against the original sources before being included in the paper. The authors take full responsibility for the final content of the manuscript, including all claims, citations, and conclusions.

REFERENCES

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Xiaofeng Fan, Yining Ma, Zhongxiang Dai, Wei Jing, Cheston Tan, and Bryan Kian Hsiang Low. Fault-tolerant federated reinforcement learning with theoretical guarantee. *Advances in neural information processing systems*, 34:1007–1021, 2021.

Chin Pang Ho, Marek Petrik, and Wolfram Wiesemann. Robust $\backslash phi$-divergence mdps. *Advances in Neural Information Processing Systems*, 35:32680–32693, 2022.

Songnam Hong and Jeongmin Chae. Communication-efficient randomized algorithm for multi-kernel online federated learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):9872–9886, 2022. doi: 10.1109/TPAMI.2021.3131552.

Ukjo Hwang and Songnam Hong. On practical robust reinforcement learning: Adjacent uncertainty set and double-agent algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 36(4):7696–7710, 2024.

Ukjo Hwang and Songnam Hong. Federated reinforcement learning in heterogeneous environments. *arXiv preprint arXiv:2507.14487*, 2025.

Garud N Iyengar. Robust dynamic programming. *Mathematics of Operations Research*, 30(2): 257–280, 2005.

Hao Jin, Yang Peng, Wenhao Yang, Shusen Wang, and Zhihua Zhang. Federated reinforcement learning with environment heterogeneity. In *International conference on artificial intelligence and statistics*, pp. 18–37. PMLR, 2022.

Sajad Khodadadian, Pranay Sharma, Gauri Joshi, and Siva Theja Maguluri. Federated reinforcement learning: Linear speedup under markovian sampling. In *International conference on machine learning*, pp. 10997–11057. PMLR, 2022.

Dohyeok Kwon, Jinwoo Park, and Songnam Hong. Tighter regret analysis and optimization of online federated learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (12):15772–15789, 2023. doi: 10.1109/TPAMI.2023.3316672.

Safwan Labbi, Paul Mangold, Daniil Tiapkin, and Eric Moulines. On global convergence rates for federated softmax policy gradient under heterogeneous environments. *arXiv preprint arXiv:2505.23459*, 2025.

Guangchen Lan, Dong-Jun Han, Abolfazl Hashemi, Vaneet Aggarwal, and Christopher G Brinton. Asynchronous federated reinforcement learning with policy gradient updates: Algorithm design and convergence analysis. *arXiv preprint arXiv:2404.08003*, 2024.

Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. In *International conference on machine learning*, pp. 4615–4625. PMLR, 2019.

Chetan Nadiger, Anil Kumar, and Sherine Abdelhak. Federated reinforcement learning for fast personalization. In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pp. 123–127. IEEE, 2019.

Jiaju Qi, Qihao Zhou, Lei Lei, and Kan Zheng. Federated reinforcement learning: Techniques, applications, and open challenges. *arXiv preprint arXiv:2108.11887*, 2021.

Antonin Raffin. Rl baselines3 zoo. https://github.com/DLR-RM/rl-baselines3-zoo, 2020.

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(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Aurko Roy, Huan Xu, and Sebastian Pokutta. Reinforcement learning under model mismatch. *Advances in neural information processing systems*, 30, 2017.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *International Conference on Learning Representations (ICLR)*, 2020.

Harekrushna Sahu, Pratik Jawanpuria, and Pranay Sharma. Fedsea: Achieving benefit of parallelization in federated online learning. *arXiv preprint arXiv:2604.19336*, 2026.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Han Wang, Aritra Mitra, Hamed Hassani, George J Pappas, and James Anderson. Federated temporal difference learning with linear function approximation under environmental heterogeneity. *arXiv preprint arXiv:2302.02212*, 2023.

Han Wang, Sihong He, Zhili Zhang, Fei Miao, and James Anderson. Momentum for the win: Collaborative federated reinforcement learning across heterogeneous environments. *arXiv preprint arXiv:2405.19499*, 2024.

Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust markov decision processes. *Mathematics of Operations Research*, 38(1):153–183, 2013.

Zhijie Xie and Shenghui Song. Fedkl: Tackling data heterogeneity in federated reinforcement learning by penalizing kl divergence. *IEEE Journal on Selected Areas in Communications*, 41(4): 1227–1242, 2023.

Chenyu Zhang, Han Wang, Aritra Mitra, and James Anderson. Finite-time analysis of on-policy heterogeneous federated reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pp. 501–545, 2024.

Zhong Zheng, Fengyu Gao, Lingzhou Xue, and Jing Yang. Federated q-learning: Linear regret speedup with low communication cost. In *International Conference on Learning Representations*, volume 2024, pp. 57399–57443, 2024.

Ye Zhu and Xiaowen Gong. Single-loop federated actor-critic across heterogeneous environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 23054–23062, 2025.

A Proof of Proposition 1

Recall that the weighted objective is defined as

$$J_\lambda(\pi_\theta) = \sum_{k=1}^{K} \lambda_k J_k(\pi_\theta),$$

where $J_k(\pi_\theta) = \mathbb{E}_{\tau\sim(\pi_\theta,P_k)}[R(\tau)]$ with $R(\tau) := \sum_{t=0}^{T-1} \gamma^t r_t$. Since $\lambda \in \Delta_K$ is fixed and independent of θ, differentiation is linear in λ, and we obtain

$$\nabla_\theta J_\lambda(\pi_\theta) = \sum_{k=1}^{K} \lambda_k \nabla_\theta J_k(\pi_\theta). \tag{14}$$

It remains to compute $\nabla_\theta J_k(\pi_\theta)$ for each environment k. We assume standard regularity conditions: bounded rewards, a finite horizon T, and a policy π_θ that is continuously differentiable in θ with full support and $\mathbb{E}\big[\|\nabla_\theta \log \pi_\theta(a_t \mid s_t)\|\big] < \infty$. Under these conditions, the dominated convergence theorem justifies exchanging the gradient and the expectation. The trajectory density under environment k factorizes as

$$p_\theta^{(k)}(\tau) = \rho_0(s_0) \prod_{t=0}^{T-1} \pi_\theta(a_t \mid s_t)\, P_k(s_{t+1} \mid s_t, a_t),$$

and since ρ_0 and P_k do not depend on θ, we have $\nabla_\theta \log p_\theta^{(k)}(\tau) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t)$. The likelihood-ratio identity then gives

$$\nabla_\theta J_k(\pi_\theta) = \mathbb{E}_{\tau\sim(\pi_\theta,P_k)} \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \sum_{t'=0}^{T-1} \gamma^{t'} r_{t'} \right].$$

For $t' < t$, the reward $r_{t'}$ is determined before a_t is drawn, so $\mathbb{E}\big[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\, r_{t'}\big] = 0$ and only the terms with $t' \geq t$ remain. Writing $\sum_{t'=t}^{T-1} \gamma^{t'} r_{t'} = \gamma^t G_t^{(k)}$, we obtain

$$\nabla_\theta J_k(\pi_\theta) = \mathbb{E}_{\tau\sim(\pi_\theta,P_k)} \left[\sum_{t=0}^{T-1} \gamma^t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, G_t^{(k)} \right]. \tag{15}$$

Substituting equation 15 into equation 14 yields the gradient expression in Proposition 1, completing the proof. ■

B Proof of Theorem 1

We prove the primal convergence and dual no-regret results separately.

B.1 Primal Convergence

Let $\mathcal{H}_m$ denote the filtration containing all randomness up to the beginning of iteration m, and define $F_m := F(\theta^{(m)}, \lambda^{(m)})$. Since $F(\theta, \lambda)$ is L-smooth in θ (Assumption 1), the policy update $\theta^{(m+1)} = \theta^{(m)} + \eta_\pi g^{(m)}$ gives

$$F(\theta^{(m+1)}, \lambda^{(m)}) \;\geq\; F_m + \eta_\pi \langle \nabla_\theta F_m,\, g^{(m)} \rangle - \frac{L\eta_\pi^2}{2} \|g^{(m)}\|^2.$$

Taking the conditional expectation given $\mathcal{H}_m$ followed by the total expectation and using Assumption 2—which implies $\mathbb{E}[\|g^{(m)}\|^2 \mid \mathcal{H}_m] \leq \|\nabla_\theta F_m\|^2 + \sigma^2$ by the standard bias–variance decomposition—we obtain

$$\mathbb{E}\big[F(\theta^{(m+1)}, \lambda^{(m)})\big] \geq \mathbb{E}[F_m] + \frac{\eta_\pi}{2} \mathbb{E}\|\nabla_\theta F_m\|^2 - \frac{L\eta_\pi^2}{2}\sigma^2. \tag{16}$$

Here, we used the step-size condition $\eta_\pi \leq 1/L$ such that $\eta_\pi - L\eta_\pi^2/2 \geq \eta_\pi/2$.

We next account for the change induced by the dual update. Since $F(\theta, \lambda) = \sum_{k=1}^{K} \lambda_k J_k(\pi_\theta)$ is linear in λ and $|J_k(\pi_\theta)| \le J_{\max}$ by Assumption 3, we obtain

$$\left|F(\theta^{(m+1)}, \lambda^{(m+1)}) - F(\theta^{(m+1)}, \lambda^{(m)})\right| \le J_{\max} \|\lambda^{(m+1)} - \lambda^{(m)}\|_1.$$

Let $g_\lambda^{(m)} = [J_1^{(m)}, \dots, J_K^{(m)}]^\top$ denote the return vector used in the dual update equation 10. Moreover, since the projection Π_{Δ_K} is non-expansive and $\|g_\lambda^{(m)}\|_2 \le \sqrt{K}\, J_{\max}$ almost surely (by Assumption 3), the dual step satisfies, on every sample path,

$$\|\lambda^{(m+1)} - \lambda^{(m)}\|_1 \le \sqrt{K}\, \|\lambda^{(m+1)} - \lambda^{(m)}\|_2 \le \sqrt{K} \cdot \eta_\lambda \|g_\lambda^{(m)}\|_2 \le K J_{\max}\, \eta_\lambda =: C_\lambda\, \eta_\lambda,$$

and therefore

$$\mathbb{E}\big[F(\theta^{(m+1)}, \lambda^{(m+1)})\big] \ge \mathbb{E}\big[F(\theta^{(m+1)}, \lambda^{(m)})\big] - J_{\max} C_\lambda\, \eta_\lambda. \tag{17}$$

Adding equation 16 and equation 17 yields

$$\mathbb{E}\big[F(\theta^{(m+1)}, \lambda^{(m+1)})\big] \ge \mathbb{E}[F_m] + \frac{\eta_\pi}{2} \mathbb{E}\|\nabla_\theta F_m\|^2 - \frac{L\eta_\pi^2}{2}\sigma^2 - J_{\max} C_\lambda\, \eta_\lambda. \tag{18}$$

Rearranging equation 18 and summing over $m = 0, \dots, M-1$, we obtain

$$\frac{\eta_\pi}{2} \sum_{m=0}^{M-1} \mathbb{E}\|\nabla_\theta F_m\|^2 \le \sum_{m=0}^{M-1} \mathbb{E}\big[F(\theta^{(m+1)}, \lambda^{(m+1)}) - F_m\big] + \frac{L\eta_\pi^2 \sigma^2 M}{2} + J_{\max} C_\lambda\, \eta_\lambda M.$$

The first sum on the right-hand side telescopes:

$$\sum_{m=0}^{M-1} \mathbb{E}\big[F(\theta^{(m+1)}, \lambda^{(m+1)}) - F(\theta^{(m)}, \lambda^{(m)})\big] = \mathbb{E}[F(\theta^{(M)}, \lambda^{(M)})] - F(\theta^{(0)}, \lambda^{(0)}) \le C_2,$$

where the last inequality uses $|F(\theta, \lambda)| \le J_{\max}$ for all (θ, λ), which follows from Assumption 3 since $\lambda \in \Delta_K$; hence we may take $C_2 = 2J_{\max}$. Dividing both sides by $\eta_\pi M/2$, we have

$$\begin{aligned} \frac{1}{M} \sum_{m=0}^{M-1} \mathbb{E}\|\nabla_\theta F_m\|^2 &\le \frac{2C_2}{M\eta_\pi} + L\eta_\pi \sigma^2 + \frac{2J_{\max} C_\lambda\, \eta_\lambda}{\eta_\pi} \\ &= \mathcal{O}\left(\frac{1}{M\eta_\pi} + \eta_\pi + \frac{\eta_\lambda}{\eta_\pi}\right). \end{aligned} \tag{19}$$

Choosing $\eta_\pi = c_\pi M^{-1/3}$ and $\eta_\lambda = c_\lambda M^{-2/3}$ balances the three terms: $1/(M\eta_\pi) = \mathcal{O}(M^{-2/3})$, $\eta_\pi = \mathcal{O}(M^{-1/3})$, and $\eta_\lambda/\eta_\pi = \mathcal{O}(M^{-1/3})$, which gives

$$\frac{1}{M} \sum_{m=0}^{M-1} \mathbb{E}\|\nabla_\theta F_m\|^2 = \mathcal{O}\big(M^{-1/3}\big).$$

This establishes the primal convergence claim (i).

B.2 Dual No-Regret with High Probability

For each round m, define the per-round loss as

$$f_m(\lambda) := F(\theta^{(m)}, \lambda) = \sum_{k=1}^{K} \lambda_k J_k(\pi_{\theta^{(m)}}).$$

Since $\theta^{(m)}$ is $\mathcal{H}_m$-measurable, $f_m(\lambda)$ is a linear (hence convex) function of λ over Δ_K with gradient

$$\nabla_\lambda f_m = [J_1(\pi_{\theta^{(m)}}), \dots, J_K(\pi_{\theta^{(m)}})]^\top,$$

whereas the dual update is performed with the noisy estimate $g_\lambda^{(m)} = [J_1^{(m)}, \dots, J_K^{(m)}]^\top$, which satisfies $\mathbb{E}[g_\lambda^{(m)} \mid \mathcal{H}_m] = \nabla_\lambda f_m$ and $\|g_\lambda^{(m)}\|_2 \le \sqrt{K}\, J_{\max}$ almost surely by Assumption 3. Fix any $\lambda \in \Delta_K$. By the non-expansiveness of Π_{Δ_K},

$$\|\lambda^{(m+1)} - \lambda\|^2 \le \|\lambda^{(m)} - \eta_\lambda g_\lambda^{(m)} - \lambda\|^2 = \|\lambda^{(m)} - \lambda\|^2 - 2\eta_\lambda \langle g_\lambda^{(m)}, \lambda^{(m)} - \lambda\rangle + \eta_\lambda^2 \|g_\lambda^{(m)}\|^2.$$

Rearranging and summing over $m = 0, \ldots, M-1$ yields, on every sample path,

$$\sum_{m=0}^{M-1} \langle g_\lambda^{(m)},\, \lambda^{(m)} - \lambda \rangle \le \frac{\|\lambda^{(0)} - \lambda\|^2}{2\eta_\lambda} + \frac{\eta_\lambda}{2} \sum_{m=0}^{M-1} \|g_\lambda^{(m)}\|^2 \le \frac{D^2}{2\eta_\lambda} + \frac{\eta_\lambda K J_{\max}^2 M}{2}, \tag{20}$$

where $D := \max_{\lambda,\lambda' \in \Delta_K} \|\lambda - \lambda'\|_2 = \sqrt{2}$ is the Euclidean diameter of Δ_K. We emphasize that equation 20 is a deterministic inequality and valid for any realization of the noise.

Next, we relate the surrogate bound to the true losses. By linearity of f_m,

$$f_m(\lambda^{(m)}) - f_m(\lambda) = \langle \nabla_\lambda f_m,\, \lambda^{(m)} - \lambda \rangle = \langle g_\lambda^{(m)},\, \lambda^{(m)} - \lambda \rangle + X_m(\lambda),$$

where the noise term is defined as

$$X_m(\lambda) := \langle \nabla_\lambda f_m - g_\lambda^{(m)},\, \lambda^{(m)} - \lambda \rangle.$$

Summing over m and applying equation 20, for any fixed $\lambda \in \Delta_K$,

$$\sum_{m=0}^{M-1} \big(f_m(\lambda^{(m)}) - f_m(\lambda)\big) \le \frac{D^2}{2\eta_\lambda} + \frac{\eta_\lambda K J_{\max}^2 M}{2} + \sum_{m=0}^{M-1} X_m(\lambda). \tag{21}$$

We now control the noise term. Fix $\lambda \in \Delta_K$ (deterministic). Since $\lambda^{(m)}$ and $\theta^{(m)}$ are $\mathcal{H}_m$-measurable and $\mathbb{E}[g_\lambda^{(m)} \mid \mathcal{H}_m] = \nabla_\lambda f_m$, the sequence $\{X_m(\lambda)\}_{m=0}^{M-1}$ is a martingale difference sequence with respect to $\{\mathcal{H}_m\}$:

$$\mathbb{E}[X_m(\lambda) \mid \mathcal{H}_m] = 0.$$

Moreover, by Hölder's inequality, Assumption 3, and the fact that $\|\lambda^{(m)} - \lambda\|_1 \le 2$ on the simplex,

$$|X_m(\lambda)| \le \|\nabla_\lambda f_m - g_\lambda^{(m)}\|_\infty \cdot \|\lambda^{(m)} - \lambda\|_1 \le (2J_{\max}) \cdot 2 = 4J_{\max}$$

almost surely. By the Azuma–Hoeffding inequality, for any $t > 0$,

$$\mathbb{P}\left(\sum_{m=0}^{M-1} X_m(\lambda) \ge t\right) \le \exp\left(-\frac{t^2}{32\, J_{\max}^2 M}\right).$$

Setting $t = 4J_{\max}\sqrt{2M \log(K/\delta)}$, we obtain

$$\mathbb{P}\left(\sum_{m=0}^{M-1} X_m(\lambda) \ge 4J_{\max}\sqrt{2M \log(K/\delta)}\right) \le \frac{\delta}{K}. \tag{22}$$

Since each f_m is linear in λ, the sum $\sum_{m=0}^{M-1} f_m(\lambda)$ is also linear, and hence its minimum over Δ_K is attained at a vertex:

$$\min_{\lambda \in \Delta_K} \sum_{m=0}^{M-1} f_m(\lambda) = \min_{k \in [K]} \sum_{m=0}^{M-1} f_m(e_k),$$

where $e_1, \ldots, e_K$ denote the standard basis vectors. Each vertex e_k is a fixed (deterministic) comparator, so equation 22 applies to each $\lambda = e_k$. By a union bound over $k \in [K]$, with probability at least $1 - \delta$,

$$\max_{k \in [K]} \sum_{m=0}^{M-1} X_m(e_k) \le 4J_{\max}\sqrt{2M \log(K/\delta)}. \tag{23}$$

Combining equation 21 with equation 23: on the event in equation 23, applying equation 21 with $\lambda = e_k$ for each k and taking the maximum over k yields, with probability at least $1 - \delta$,

$$\sum_{m=0}^{M-1} f_m(\lambda^{(m)}) - \min_{\lambda \in \Delta_K} \sum_{m=0}^{M-1} f_m(\lambda) = \max_{k \in [K]} \sum_{m=0}^{M-1} \big(f_m(\lambda^{(m)}) - f_m(e_k)\big) \le \frac{D^2}{2\eta_\lambda} + \frac{\eta_\lambda K J_{\max}^2 M}{2} + 4J_{\max}\sqrt{2M \log(K/\delta)}.$$

Substituting $\eta_\lambda = c_\lambda M^{-2/3}$, the three terms scale as $\frac{D^2}{2\eta_\lambda} = \mathcal{O}(M^{2/3})$, $\frac{\eta_\lambda K J_{\max}^2 M}{2} = \mathcal{O}(M^{1/3})$, and $4J_{\max}\sqrt{2M\log(K/\delta)} = \mathcal{O}\big(J_{\max}\sqrt{M\log(K/\delta)}\big)$. Since $\sqrt{M} = o(M^{2/3})$, the first term dominates for any fixed K and δ. Substituting $f_m(\lambda) = F(\theta^{(m)}, \lambda)$, we conclude that with probability at least $1-\delta$,

$$\sum_{m=0}^{M-1} F(\theta^{(m)}, \lambda^{(m)}) - \min_{\lambda\in\Delta_K} \sum_{m=0}^{M-1} F(\theta^{(m)}, \lambda) \le \mathcal{O}\Big(M^{2/3} + J_{\max}\sqrt{M\log(K/\delta)}\Big) = \mathcal{O}\big(M^{2/3}\big).$$

This establishes the dual no-regret claim (ii) and completes the proof. ■

C Proof of Proposition 2

Let $\xi_k := g_k - \nabla_\theta J_k(\pi_\theta)$. Under the per-agent variance assumption, we obtain

$$\mathbb{E}[\xi_k \mid \mathcal{H}_m] = 0 \text{ and } \mathbb{E}\big[\|\xi_k\|^2 \mid \mathcal{H}_m\big] \le \sigma_u^2.$$

Expanding the squared norm yields

$$\begin{aligned}\mathbb{E}\left[\left\|\sum_{k=1}^K \lambda_k\xi_k\right\|^2 \,\middle|\, \mathcal{H}_m\right] &= \mathbb{E}\left[\left\langle \sum_{i=1}^K \lambda_i\xi_i, \sum_{j=1}^K \lambda_j\xi_j\right\rangle \,\middle|\, \mathcal{H}_m\right] \\ &= \sum_{k=1}^K \lambda_k^2 \mathbb{E}\big[\|\xi_k\|^2 \mid \mathcal{H}_m\big] + \sum_{i\neq j} \lambda_i\lambda_j \mathbb{E}[\langle\xi_i,\xi_j\rangle \mid \mathcal{H}_m]. \end{aligned} \tag{24}$$

Under the conditional independence assumption on $\{\xi_k\}_{k=1}^K$ given $\mathcal{H}_m$, the cross terms vanish, i.e., $\mathbb{E}[\langle\xi_i,\xi_j\rangle \mid \mathcal{H}_m] = 0$ for $i \neq j$. Therefore, we have

$$\begin{aligned}\mathbb{E}\left[\left\|\sum_{k=1}^K \lambda_k\xi_k\right\|^2 \,\middle|\, \mathcal{H}_m\right] &= \sum_{k=1}^K \lambda_k^2 \mathbb{E}\big[\|\xi_k\|^2 \mid \mathcal{H}_m\big] \\ &\le \sigma_u^2 \sum_{k=1}^K \lambda_k^2 = \frac{\sigma_u^2}{K_{\text{eff}}}, \end{aligned} \tag{25}$$

where the last equality follows from the definition $K_{\text{eff}} := 1/\sum_{k=1}^K \lambda_k^2$ in Proposition 2. This completes the proof. ■

D Proof of Corollary 1

We substitute the per-iteration variance bound $\sigma_u^2/K_{\text{eff}}^{(m)}$ from Proposition 2 into the proof of Theorem 1. The noise term $L\eta_\pi\sigma^2$ in equation 19 is then refined as

$$\frac{L\eta_\pi\sigma_u^2}{M}\sum_{m=0}^{M-1}\frac{1}{K_{\text{eff}}^{(m)}} = \frac{L\eta_\pi\sigma_u^2}{\bar{K}_{\text{eff}}}, \tag{26}$$

where

$$\bar{K}_{\text{eff}} := \left(\frac{1}{M}\sum_{m=0}^{M-1}\frac{1}{K_{\text{eff}}^{(m)}}\right)^{-1} \tag{27}$$

denotes the harmonic mean of $\{K_{\text{eff}}^{(m)}\}_{m=0}^{M-1}$ over iterations, and satisfies $1 \le \bar{K}_{\text{eff}} \le K$. We note that $\bar{K}_{\text{eff}}$ is determined by the realized trajectory of the dual weights and is therefore random; the refinement below is stated under the almost-sure lower bound $\bar{K}_{\text{eff}} \ge \kappa$ assumed in the corollary. Consequently, the refined primal convergence bound is

$$\frac{1}{M}\sum_{m=0}^{M-1}\mathbb{E}\|\nabla_\theta F_m\|^2 \le \frac{2C_2}{M\eta_\pi} + \frac{L\eta_\pi\sigma_u^2}{\bar{K}_{\text{eff}}} + \frac{2J_{\max}C_\lambda\eta_\lambda}{\eta_\pi}. \tag{28}$$

Recall from the proof of Theorem 1 that $C_\lambda = KJ_{\max}$. Under the almost-sure bound $\bar{K}_{\text{eff}} \geq \kappa$, inequality equation 28 implies

$$\frac{1}{M}\sum_{m=0}^{M-1}\mathbb{E}\|\nabla_\theta F_m\|^2 \leq \frac{2C_2}{M\eta_\pi} + \frac{L\eta_\pi\sigma_u^2}{\kappa} + \frac{2KJ_{\max}^2\eta_\lambda}{\eta_\pi}. \tag{29}$$

With the step sizes $\eta_\pi = \Theta(\kappa^{2/3}M^{-1/3})$ and $\eta_\lambda = \Theta(\kappa^{1/3}K^{-1}M^{-2/3})$, subject to $\eta_\pi \leq 1/L$, the three terms on the right-hand side of equation 29 scale respectively as $\mathcal{O}\big(\kappa^{-2/3}M^{-2/3}\big)$, $\mathcal{O}\big(\kappa^{-1/3}M^{-1/3}\big)$, and $\mathcal{O}\big(\kappa^{-1/3}M^{-1/3}\big)$. It follows that

$$\frac{1}{M}\sum_{m=0}^{M-1}\mathbb{E}\|\nabla_\theta F_m\|^2 = \mathcal{O}\Big((\kappa M)^{-1/3}\Big).$$

Setting the right-hand side to ϵ and solving for M yields $M = \mathcal{O}\big(1/(\kappa\epsilon^3)\big)$, which completes the proof. ■

E Algorithmic Details

This section provides the implementation details of the PPO-based realization of AMPO described in Section 5. The shared actor π_θ is maintained by the server, whereas each agent k maintains its own critic V_{ψ_k}. During local computation, the actor remains fixed at the broadcast parameter $\theta^{(m)}$: agents compute actor-gradient contributions but do not take local actor steps. In contrast, the critic is updated locally over PPO minibatches and is not communicated to the server.

E.1 Local PPO Computation

At communication round m, agent k receives the global actor $\theta^{(m)}$ and collects a rollout under its local transition dynamics P_k. GAE advantages and value targets are computed from the collected rollout and stored in the rollout buffer. These stored quantities are then reused over the subsequent PPO epochs and minibatches.

For each minibatch $\mathcal{B}_{k,b}^{(m)}$, the agent computes an actor-gradient contribution at the fixed broadcast parameter $\theta^{(m)}$ as

$$\hat{g}_{k,b}^{(m)} := \widehat{\mathbb{E}}_{\mathcal{B}_{k,b}^{(m)}}\left[\nabla_\theta \log \pi_{\theta^{(m)}}(a_t \mid s_t)\,\widetilde{A}_{k,t}^{(m)} + c_{\text{ent}}\nabla_\theta \mathcal{H}(\pi_{\theta^{(m)}}(\cdot \mid s_t))\right], \tag{30}$$

where $\widetilde{A}_{k,t}^{(m)}$ denotes the GAE advantage used for the actor update, and c_{ent} is the entropy coefficient. The local actor-gradient direction is accumulated over the PPO minibatches as

$$g_k^{(m)} := \sum_{b=1}^{B_k^{(m)}} \hat{g}_{k,b}^{(m)}. \tag{31}$$

The actor parameters are not modified during this local computation. Hence, the minibatches are used only to construct $g_k^{(m)}$, and the actual actor update is performed at the server through equation 9. Equation 30 gives the practical counterpart of the advantage-weighted gradient in equation 13.

The critic is updated separately using the value targets stored during rollout collection. Let $\hat{V}_{k,t}^{\text{targ},(m)}$ denote the stored value target. For minibatch $\mathcal{B}_{k,b}^{(m)}$, we define

$$\ell_{V,k,b}^{(m)}(\psi_k) := \widehat{\mathbb{E}}_{\mathcal{B}_{k,b}^{(m)}}\left[\Big(V_{\psi_k}(s_t) - \hat{V}_{k,t}^{\text{targ},(m)}\Big)^2\right], \tag{32}$$

and update

$$\psi_k \leftarrow \psi_k - \eta_V \nabla_{\psi_k}\ell_{V,k,b}^{(m)}(\psi_k). \tag{33}$$

Unlike the actor, the critic therefore takes an optimizer step at every local minibatch using the value targets stored in the rollout buffer.

Algorithm 1 Practical AMPO-PPO with Local Critics

Require: Initial actor $\theta^{(0)}$, local critics $\{\psi_k^{(0)}\}_{k=1}^K$, dual weights $\lambda^{(0)} \in \Delta_K$, communication rounds M, local PPO epochs E, server actor step size η_π, critic step size η_V, dual step size η_λ

for $m = 0, 1, \ldots, M-1$ **do**
  Server: broadcast $\theta^{(m)}$ to all agents
  for each agent $k \in [K]$ **in parallel do**
    Collect a local rollout using $\pi_{\theta^{(m)}}$ in environment P_k
    Compute and store GAE advantages and value targets
    Update the persistent episodic-return accumulator using equation 34
    $g_k^{(m)} \leftarrow 0$
    for $e = 1, \ldots, E$ **do**
      for each minibatch $\mathcal{B}_{k,b}^{(m)}$ **do**
        Compute $\hat{g}_{k,b}^{(m)}$ using equation 30
        $g_k^{(m)} \leftarrow g_k^{(m)} + \hat{g}_{k,b}^{(m)}$ ▷ no local actor update
        Update ψ_k using equation 33 ▷ local critic update
      end for
    end for
    Update $\widetilde{J}_k^{(m)}$ from completed episodes using equation 35–equation 36
    Send $\big(g_k^{(m)}, \widetilde{J}_k^{(m)}\big)$ to the server
  end for
  Server: update $\theta^{(m+1)}$ using equation 9 with $\lambda^{(m)}$
  if the dual-update interval is reached and the return warm-up is complete **then**
    Compute $z^{(m)}$ using equation 39
    $\lambda^{(m+1)} \leftarrow \Pi_{\Delta_K}\big(\lambda^{(m)} - \eta_\lambda z^{(m)}\big)$ ▷ using equation 40
  else
    $\lambda^{(m+1)} \leftarrow \lambda^{(m)}$
  end if
end for
return $\theta^{(M)}$

E.2 Practical Dual-Return Construction

The return signal used for dual adaptation is constructed separately from the GAE advantages and value targets used by PPO. For a completed episode $\tau = (s_0, a_0, r_0, \ldots, s_{T_\tau-1}, a_{T_\tau-1}, r_{T_\tau-1})$, we define

$$R_{\mathrm{dual}}(\tau) := \sum_{t=0}^{T_\tau-1} r_t. \tag{34}$$

Because the PPO rollout length need not coincide with an episode boundary, the implementation does not treat a rollout fragment as a complete trajectory. Instead, each agent maintains a persistent return accumulator across both rollout and communication-round boundaries. Rewards are added to the accumulator until a terminal transition is observed, at which point the complete episodic return in equation 34 is recorded and the accumulator is reset.

Let $\mathcal{C}_k^{(m)}$ denote the set of episodes completed by agent k during communication round m. The mean completed-episode return is

$$\bar{R}_k^{(m)} := \frac{1}{|\mathcal{C}_k^{(m)}|} \sum_{\tau \in \mathcal{C}_k^{(m)}} R_{\mathrm{dual}}(\tau). \tag{35}$$

Episodes may span multiple communication rounds; in that case, $R_{\mathrm{dual}}(\tau)$ includes all rewards accumulated over the complete episode and is recorded in the round in which the episode terminates. The return signal communicated to the server is maintained through an exponential moving average:

$$\widetilde{J}_k^{(m)} = \beta_J \widetilde{J}_k^{(m-1)} + (1 - \beta_J)\bar{R}_k^{(m)}. \tag{36}$$

Equation 36 is applied only when at least one episode terminates during the round. The first available completed-episode mean initializes $\widetilde{J}_k$ directly, and if no episode is completed during a round, the previous value is retained.

Before a dual update, the server computes the cross-environment return spread

$$\sigma_J^{(m)} := \mathrm{Std}\left(\widetilde{J}_1^{(m)}, \ldots, \widetilde{J}_K^{(m)}\right) \tag{37}$$

and maintains its EMA-smoothed scale

$$s^{(m)} = \beta_s s^{(m-1)} + (1-\beta_s)\sigma_J^{(m)}. \tag{38}$$

The resulting standardized dual signal is

$$z_k^{(m)} := \frac{\widetilde{J}_k^{(m)} - \frac{1}{K}\sum_{j=1}^{K} \widetilde{J}_j^{(m)}}{s^{(m)}}. \tag{39}$$

At a scheduled dual-update round, after every agent has accumulated the prescribed minimum number of completed episodes, the server applies

$$\lambda^{(m+1)} = \Pi_{\Delta_K}\left(\lambda^{(m)} - \eta_\lambda z^{(m)}\right). \tag{40}$$

The dual variable is therefore updated on a slower timescale than the actor and reacts to persistent cross-environment performance differences rather than individual episodic fluctuations.

The practical signal $\widetilde{J}_k^{(m)}$ should be distinguished from the instantaneous Monte Carlo estimator $J_k^{(m)}$ used in the idealized analysis. The EMA introduces temporal smoothing, and an episode can span multiple communication rounds while the global actor changes. Moreover, in the present implementation the dual signal is based on undiscounted episodic returns. Consequently, the theoretical results in Section 4 characterize the idealized AMPO update, whereas the implementation above uses a stabilized training-stream proxy for dual adaptation and, as is standard in practice, omits the γ^t weighting of the policy-gradient terms in equation 7. Algorithm 1 summarizes the complete training procedure.

E.3 PPOAvg versus AMPO-PPO(U)

Although both PPOAvg and AMPO-PPO(U) assign equal importance to all agents, they differ in how local policy optimization is performed. AMPO-PPO(U) is the uniform-weight variant of AMPO-PPO, obtained by fixing $\lambda_k = 1/K$ throughout training. Each agent keeps the broadcast actor $\theta^{(m)}$ fixed during local computation and evaluates all actor-gradient contributions at this common parameter. The resulting server update is therefore

$$\theta^{(m+1)} = \theta^{(m)} + \eta_\pi \frac{1}{K}\sum_{k=1}^{K} g_k^{(m)}, \tag{41}$$

where every $g_k^{(m)}$ is constructed at the same broadcast actor $\theta^{(m)}$. Thus, AMPO-PPO(U) removes adaptive dual weighting while preserving the server-owned actor and local-gradient computation of AMPO-PPO.

PPOAvg instead follows the conventional federated PPO procedure in which each agent performs local PPO actor optimization after receiving the global policy. Consequently, the actor parameters evolve locally across PPO minibatches and epochs, so later local gradients are evaluated at agent-specific policy parameters rather than at the common broadcast parameter $\theta^{(m)}$. The resulting local updates are then aggregated uniformly at the server. PPO clipping therefore plays its standard role in PPOAvg by constraining local policy changes, whereas AMPO-PPO(U) does not rely on clipping to control local actor drift because no local actor update is performed.

The distinction is important for interpreting our ablations. PPOAvg serves as a conventional average-return federated PPO baseline, whereas AMPO-PPO(U) controls specifically for the adaptive weighting mechanism within the AMPO-PPO architecture. Accordingly, comparing AMPO-PPO(U) with AMPO-PPO(A) isolates the effect of replacing the fixed weights $\lambda_k = 1/K$ with the adaptive dual weights, while comparison with PPOAvg additionally reflects the difference between server-side gradient application and conventional local PPO actor optimization.

Table 4: Local evaluation return profiles and aggregation-weight concentration across heterogeneous MuJoCo environments. The columns -0.30 to $+0.30$ denote the environment-specific perturbation levels. Mean, Spread, and Rel. Spread summarize the return profile across local environments, while K_{eff} measures the concentration of the aggregation weights, with smaller values indicating stronger focusing on a subset of environments. All return statistics are averaged over the final ten evaluation points.

Env.	Perturbation	Method	-0.30	-0.15	0	$+0.15$	$+0.30$	Mean	Spread	Rel. Spread (%)	K_{eff}
Ant	Friction	AMPO-PPO(U)	6341.1	5634.7	4483.2	2919.4	1842.9	4244.3	4498.1	106.0	5.00
		AMPO-PPO(A)	5558.5	5309.0	4841.3	4189.2	3299.5	4639.5	2259.0	48.7	2.85
		λ	0.000	0.036	0.176	0.330	0.458	–	–	–	–
	Gravity	AMPO-PPO(U)	2343.8	3598.3	4611.1	5170.3	5448.3	4234.4	3104.4	73.3	5.00
		AMPO-PPO(A)	3109.0	4209.2	4717.3	4787.7	4512.7	4267.2	1678.7	39.3	2.59
		λ	0.520	0.308	0.140	0.031	0.001	–	–	–	–
HalfCheetah	Friction	AMPO-PPO(U)	3696.5	3898.1	3861.3	3924.6	3953.1	3866.7	256.6	6.6	5.00
		AMPO-PPO(A)	4855.7	5070.6	5268.7	5509.8	5498.6	5240.7	654.1	12.5	4.47
		λ	0.235	0.247	0.210	0.179	0.130	–	–	–	–
	Gravity	AMPO-PPO(U)	3489.8	4344.5	4995.6	5083.8	4779.9	4538.7	1594.0	35.1	5.00
		AMPO-PPO(A)	4153.9	4927.6	5086.1	4968.3	4735.8	4774.3	932.3	19.5	2.96
		λ	0.487	0.280	0.095	0.055	0.083	–	–	–	–
Hopper	Friction	AMPO-PPO(U)	1308.6	2568.9	3141.0	3087.3	1871.0	2395.4	1832.4	76.5	5.00
		AMPO-PPO(A)	1837.2	2829.0	2958.7	3233.3	2592.2	2690.1	1396.1	51.9	2.56
		λ	0.445	0.081	0.002	0.044	0.428	–	–	–	–
	Gravity	AMPO-PPO(U)	2105.5	2997.4	3028.9	3012.4	2949.7	2818.8	923.4	32.8	5.00
		AMPO-PPO(A)	2643.2	2824.0	3047.1	3059.3	3082.4	2931.2	439.2	15.0	3.39
		λ	0.246	0.099	0.120	0.174	0.362	–	–	–	–
Walker2d	Friction	AMPO-PPO(U)	3910.2	4156.6	4266.7	4177.1	3977.6	4097.7	356.4	8.7	5.00
		AMPO-PPO(A)	3924.8	4103.0	4229.3	4222.6	4039.4	4103.8	304.5	7.4	3.95
		λ	0.348	0.222	0.140	0.127	0.164	–	–	–	–
	Gravity	AMPO-PPO(U)	3785.4	3936.2	4231.7	4147.1	3492.7	3918.6	739.0	18.9	5.00
		AMPO-PPO(A)	3476.2	3901.6	3902.0	4124.4	3974.8	3875.8	648.1	16.7	3.81
		λ	0.273	0.163	0.087	0.115	0.362	–	–	–	–

F Experimental Details

This section provides additional details for the experimental setup and the derived statistics used in the main text. In particular, we describe the construction of the heterogeneous MuJoCo environments, specify how the local-return imbalance and worst-environment dynamics statistics are computed, and provide the complete per-environment return profiles that are omitted from the main text for space.

F.1 Environment Perturbations

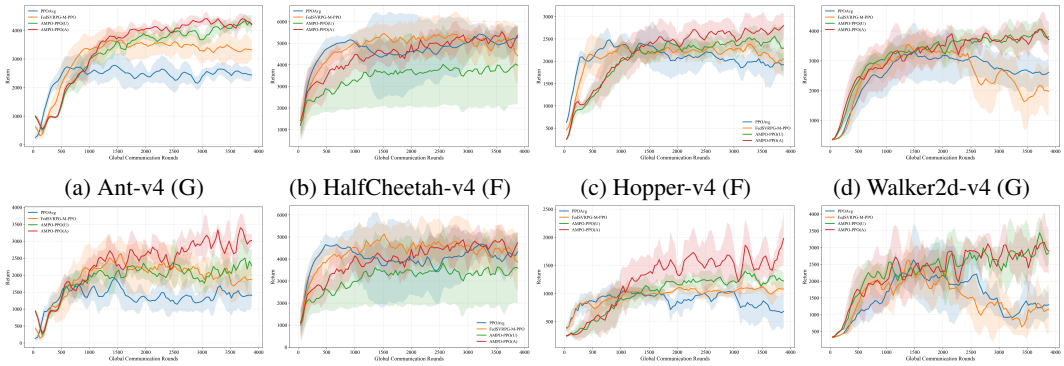
We provide additional details on the physical parameters varied in the MuJoCo simulator. For the *gravity* perturbation, we scale the nominal gravitational acceleration according to the environment-specific perturbation level. Negative perturbations reduce the magnitude of gravity, whereas positive perturbations increase it, thereby changing the forces acting on the agent and the resulting locomotion dynamics.

For the *friction* perturbation, we analogously scale the nominal contact-friction parameter. Negative perturbations correspond to lower friction and positive perturbations to higher friction, modifying the contact dynamics between the agent and the environment.

For each benchmark–perturbation setting, only the selected physical parameter is varied across environments, and the same collection of perturbed environments is used for all compared methods.

F.2 Evaluation Statistics and Reporting Protocol

At each evaluation point, the current global policy is evaluated independently in every local environment over ten episodes. For environment k, we denote the resulting mean episodic return by J_k^{eval}.



(a) Ant-v4 (G) (b) HalfCheetah-v4 (F) (c) Hopper-v4 (F) (d) Walker2d-v4 (G)

(e) Ant-v4 (G) (f) HalfCheetah-v4 (F) (g) Hopper-v4 (F) (h) Walker2d-v4 (G)

Figure 3: **Additional average and worst-case local evaluation learning curves.** These environment–perturbation combinations are omitted from the main figures for space. The top row shows the average local return, while the bottom row shows the worst-case local return. Here, (F) and (G) denote friction and gravity perturbations, respectively.

Each episodic return is the cumulative reward collected over one evaluation episode, and J_k^{eval} is obtained by averaging these returns over the ten evaluation episodes.

For the final-stage statistics reported in the tables, we average the local evaluation returns over the final ten evaluation points and then across the five training seeds. Let $\bar{J}_k$ denote the resulting final-stage return of environment k. We then define the mean local return as

$$\bar{J}_{\text{mean}} := \frac{1}{K} \sum_{k=1}^{K} \bar{J}_k. \tag{42}$$

To quantify performance imbalance across the heterogeneous local environments, we define the absolute return spread as

$$\text{Spread} := \max_{k \in [K]} \bar{J}_k - \min_{k \in [K]} \bar{J}_k, \tag{43}$$

and its scale-normalized counterpart as

$$\text{RelSpread} := 100 \times \frac{\text{Spread}}{|\bar{J}_{\text{mean}}|}. \tag{44}$$

Thus, Spread measures the absolute gap between the best- and worst-performing local environments, whereas Rel. Spread normalizes this gap by the magnitude of the mean local return, facilitating comparison across tasks with different return scales.

For AMPO-PPO(A), the aggregation weights reported together with these return profiles are averaged over the same final evaluation window and across seeds. Let $\bar{\lambda} = (\bar{\lambda}_1, \ldots, \bar{\lambda}_K)$ denote this averaged weight profile. We summarize its concentration using the effective number of environments

$$K_{\text{eff}} := \frac{1}{\sum_{k=1}^{K} \bar{\lambda}_k^2}. \tag{45}$$

For AMPO-PPO(U), $\lambda_k = 1/K$ and thus $K_{\text{eff}} = K$ by construction. In AMPO-PPO(A), stronger concentration of the adaptive weights on a subset of environments results in a smaller K_{eff}.

We separately characterize how the identity of the worst-performing environment evolves over the entire training trajectory. For each seed s and evaluation point t, define

$$k_{t,s}^{\text{worst}} := \arg\min_{k \in [K]} J_{k,t,s}^{\text{eval}}. \tag{46}$$

The *Worst Switch* rate is the fraction of consecutive evaluation points at which this identity changes:

$$\text{WorstSwitch}_s := \frac{1}{T_{\text{eval}} - 1} \sum_{t=2}^{T_{\text{eval}}} \mathbf{1}\left\{k_{t,s}^{\text{worst}} \neq k_{t-1,s}^{\text{worst}}\right\}. \tag{47}$$

Table 5: Computational cost and memory usage measured on Ant-v4 with friction perturbation. Wall-clock time is normalized per 10 communication rounds.

Method	Time / 10 rounds (s)	Peak RSS (MB)
PPOAvg	15.18	371.4
FedSVRPG-M-PPO	13.79	379.3
AMPO-PPO(U)	16.21	376.3
AMPO-PPO(A)	16.45	377.1

A smaller value indicates a more persistent performance bottleneck, whereas a larger value indicates that the worst-performing environment changes more frequently during training.

The *Dominant Worst Share* measures the fraction of evaluation points occupied by the environment that is worst most frequently:

$$\mathrm{DomWorstShare}_s := \max_{k\in[K]} \frac{1}{T_{\mathrm{eval}}} \sum_{t=1}^{T_{\mathrm{eval}}} \mathbf{1}\left\{k_{t,s}^{\mathrm{worst}} = k\right\}. \tag{48}$$

A value close to one indicates that a single environment remains the dominant bottleneck over most of training, whereas a smaller value indicates that the worst-performing environment is distributed across multiple environments. The Worst Switch and Dominant Worst Share values reported in the main text are computed separately for each training seed over all evaluation points and are reported as the mean and standard deviation across seeds.

F.3 Complete Local Return Profiles and Aggregation Weights

The main text reports compact summary statistics to emphasize local-return imbalance, adaptive weight concentration, and worst-case performance. For completeness, Table 4 reports the full per-environment return profiles for all benchmark–perturbation settings. In addition to the quantities retained in the main-text summary, the table includes the return obtained at each environment-specific perturbation level, the mean and absolute spread of the local return profile, the relative spread defined in equation 44, the adaptive aggregation weights, and the corresponding K_{eff}.

F.4 Computational Cost and Resource Usage

We additionally measure the computational cost and memory usage of the compared methods under a common resource benchmark. The measurements are obtained on a M5 MacBook Pro with 24 GB memory. All methods are executed on CPU using a single PyTorch thread. We use Ant-v4 with friction perturbation, $K = 5$ clients, and 512 local interaction steps per communication round. Evaluation is performed every 40 communication rounds. To facilitate comparison, Table 5 reports wall-clock time normalized per 10 communication rounds together with peak resident-set size (RSS).

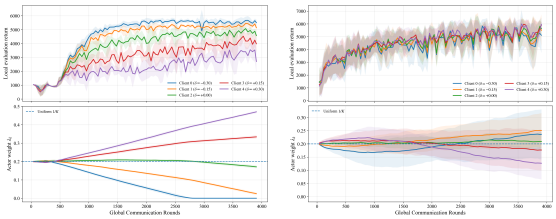
G Additional Experiments

For completeness, Fig. 3 presents the environment–perturbation combinations omitted from the main figures for space: Ant with gravity, HalfCheetah with friction, Hopper with friction, and Walker2d with gravity. The top row reports the average local return and the bottom row the worst-case local return, following the same evaluation protocol as in the main text.

G.1 Adaptive Weight Dynamics Across Heterogeneous Environments

To examine how adaptive reweighting is realized during training, Fig. 4 jointly visualizes the local evaluation returns and the corresponding dual weights λ_k for representative Ant and HalfCheetah friction settings. For each local environment, the upper panel shows its evaluation return, while the lower panel shows the weight assigned to that environment during global aggregation.

The trajectories show that the dual weights are not static throughout training. Instead, different environments receive different levels of emphasis as their relative performance changes. Environments that remain comparatively difficult tend to receive larger weights, whereas relatively well-performing environments receive less aggregation mass. Importantly, the adaptive weights evolve on a smoother timescale than the evaluation returns themselves, rather than reacting to every short-term fluctuation. This behavior is consistent with the intended role of the dual variables: λ_k acts as an accumulated measure of environment difficulty that biases the global policy update toward persistent performance bottlenecks rather than simply selecting the instantaneously worst return at each evaluation point.



(a) Ant: Friction (b) HalfCheetah: Friction

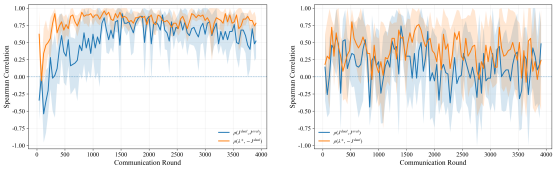
Figure 4: The upper panels show the evaluation return of each heterogeneous local environment, while the lower panels show the corresponding adaptive weights λ_k over global communication rounds.

G.2 Dual-Signal Tracking During Training

We next examine whether the dual signal and adaptive weights evolve consistently with the relative difficulty of the heterogeneous environments. Here, J^{dual} denotes the environment-wise dual-return signal, J^{eval} the corresponding environment-wise evaluation returns, and ρ the Spearman rank correlation across environments. Figure 5 tracks two such correlations over communication rounds. The first,

$$\rho\big(J^{\mathrm{dual}}, J^{\mathrm{eval}}\big)\,, \tag{49}$$

measures how well the return signal used for the dual update preserves the environment ranking observed during evaluation. The second,



(a) Hopper: Friction (b) Walker2d: Friction

Figure 5: Each panel shows the Spearman rank correlations between the dual return and evaluation return, $\rho(J^{\mathrm{dual}}, J^{\mathrm{eval}})$, and between the updated dual weight and negative dual return, $\rho(\lambda^{+}, -J^{\mathrm{dual}})$, over global communication rounds.

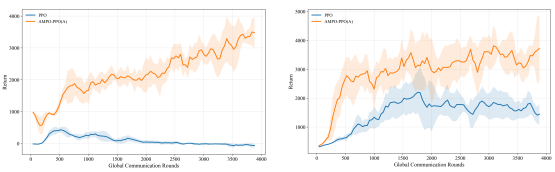
$$\rho\big(\lambda^{+}, -J^{\mathrm{dual}}\big)\,, \tag{50}$$

measures whether environments with lower dual returns receive larger updated weights, where λ^{+} denotes the weights after the dual update.

These two correlations characterize complementary stages of the adaptive mechanism: the first measures the fidelity of the dual return signal, while the second measures how strongly the updated weights align with that signal. The Hopper and Walker2d trajectories in Fig. 5 show how these relationships evolve throughout training. Variations in the first correlation indicate changes in the agreement between training and evaluation returns, whereas the second reflects how consistently the dual weights prioritize lower-return environments. Overall, the results provide direct evidence of the internal tracking behavior underlying AMPO-PPO's adaptive reweighting mechanism.

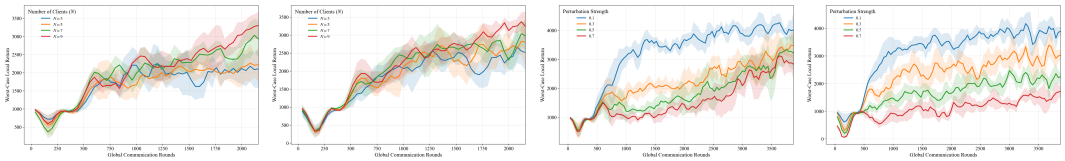
G.3 Comparison with Local PPO

To further examine the benefit of federated learning under environment heterogeneity, we compare AMPO-PPO(A) with a PPO baseline trained only with local environment interactions. Figure 6 reports the worst-case local return for the Ant and Walker2d tasks under friction perturbations. While the local PPO baseline optimizes each policy using only locally collected experience, AMPO-PPO(A) incorporates information across the



(a) Ant: Friction (b) Walker2d: Friction

Figure 6: Worst-case local performance comparison between PPO and AMPO-PPO(A) under friction perturbations.



(a) Clients: Friction (b) Clients: Gravity (c) Strength: Friction (d) Strength: Gravity

Figure 7: **Sensitivity of AMPO-PPO(A) to the number of clients (environments) and perturbation strength on Ant-v4.** The first two panels show the worst-case local return for different numbers of clients under friction and gravity perturbations, respectively, while the last two panels show the corresponding performance for different perturbation strengths.

heterogeneous environments through federated aggregation. The results show that AMPO-PPO(A) achieves improved worst-case performance compared with the local PPO baseline, indicating that federated aggregation can improve the reliability of the learned policy across heterogeneous environments.

G.4 Sensitivity to Client Population and Heterogeneity Strength

We further examine the sensitivity of AMPO-PPO(A) to two key factors in heterogeneous FRL: the number of participating clients and the severity of environment heterogeneity. Figure 7 reports the worst-case local return on Ant-v4 under both friction and gravity perturbations. In the first two panels, we vary the number of clients K while keeping the perturbation range fixed at $[-0.3,\ 0.3]$; for each value of K, the client-specific perturbation levels are evenly spaced over this interval. In the last two panels, we vary the perturbation strength while keeping the number of clients fixed. Specifically, for perturbation strength p, the client-specific perturbations are evenly spaced over $[-p,\ p]$; for example, $p = 0.7$ corresponds to perturbation levels distributed uniformly over $[-0.7,\ 0.7]$.

These experiments complement the main performance evaluation by examining AMPO-PPO(A) beyond a single choice of K and heterogeneity level. The client-population experiment therefore changes the number of heterogeneous environments while holding the overall perturbation range fixed, whereas the perturbation-strength experiment changes the severity of heterogeneity by expanding or contracting this range. We use the worst-case local return in both cases, consistent with the minimax objective of AMPO.

H Hyperparameter Settings

We use a common federated training protocol across all four MuJoCo benchmarks, while retaining environment-specific PPO hyperparameters for the underlying continuous-control tasks. The PPO backbone and its default hyperparameter settings are based on the standard implementations and configuration files provided by Stable-Baselines3 and RL Baselines3 Zoo (Raffin et al., 2021; Raffin, 2020). We retain these defaults when no task-specific override is applied and use environment-specific PPO settings where appropriate. The same PPO configuration is used for the gravity and friction perturbations of a given benchmark. Table 6 summarizes the common experimental protocol, Table 7 reports the environment-specific PPO hyperparameters, and Table 8 summarizes the algorithm-specific federated parameters and tuning ranges.

H.1 Environment-Specific PPO Configuration

The PPO hyperparameters are held fixed across the compared federated methods for a given benchmark unless otherwise noted. For Hopper, we observed unstable training with the default RL Baselines3 Zoo configuration under PPOAvg and therefore tuned the PPO hyperparameters for this benchmark; the resulting configuration is shared across all compared methods. For AMPO-PPO, the PPO optimizer learning rate below is used for the local critic optimization, while the actor is updated separately on the server using the server step size η_π reported in Table 8.

Table 6: Common experimental settings used in the main MuJoCo experiments.

Parameter	Value
Number of clients (environments) K	5
Perturbation strength p	0.3
Perturbation levels	$\{-0.30, -0.15, 0, 0.15, 0.30\}$
Local interaction steps per round	512
Rollout length n_{steps}	512
Parallel environments per client	1
Training budget	10^7 environment steps
Evaluation frequency	Every 40 communication rounds
Number of training seeds	5
Evaluation episodes per environment	10

Table 7: Environment-specific PPO hyperparameters used for the MuJoCo benchmarks.

Parameter	Ant	HalfCheetah	Hopper	Walker2d
PPO optimizer learning rate	3×10^{-4}	2.0633×10^{-5}	1×10^{-4}	5.05041×10^{-5}
Discount factor γ	0.99	0.98	0.99	0.99
GAE parameter λ_{GAE}	0.95	0.92	0.95	0.95
PPO clipping parameter	0.2	0.1	0.2	0.1
PPO epochs	10	20	10	20
Minibatch size	16	16	128	16
Entropy coefficient	0	4.01762×10^{-4}	0	5.85045×10^{-4}
Value-loss coefficient	0.5	0.58096	0.5	0.871923
Maximum gradient norm	0.5	0.8	0.5	1.0
Policy/value architecture	$[64, 64]$, Tanh	$[256, 256]$, ReLU	$[256, 256]$, ReLU	$[64, 64]$, Tanh

For HalfCheetah and Hopper, the policy and value networks use two hidden layers of 256 units with ReLU activations and disabled orthogonal initialization. HalfCheetah uses an initial policy log-standard-deviation of -2, whereas Hopper uses 0. Ant and Walker2d use the default MLP policy architecture of Stable-Baselines3 framework (Raffin et al., 2021).

H.2 Algorithm-Specific Hyperparameters

The federated methods share the PPO configurations above but differ in the hyperparameters governing server-side aggregation and adaptation. For PPOAvg, the server mixing weight is fixed to 1.0. For AMPO-PPO, we tune the server actor step size over $\eta_\pi \in \{1 \times 10^{-3}, 3 \times 10^{-3}\}$ and, for the adaptive variant AMPO-PPO(A), the dual step size over $\eta_\lambda \in \{1 \times 10^{-4}, 3 \times 10^{-4}\}$. The selected actor step sizes are 3×10^{-3} for Ant and Hopper and 1×10^{-3} for HalfCheetah and Walker2d. The main AMPO-PPO(A) results use $\eta_\lambda = 3 \times 10^{-4}$ for all environments.

For FedSVRPG-M-PPO, we tune the server update weight over $\{0.5, 0.7, 1.0\}$ and the momentum coefficient over $\beta_{\text{mom}} \in \{0.85, 0.90, 0.95\}$. These parameters control the magnitude of the server update and the strength of the momentum-based variance-reduction mechanism, respectively. The selected server update weight is 1.0 for Walker2d and 0.5 for the other benchmarks, while the selected momentum coefficient is 0.9 for Ant and 0.95 for HalfCheetah, Hopper, and Walker2d.

For the practical dual adaptation in AMPO-PPO(A), we use a return-EMA coefficient of $\beta_J = 0.9$ and a scale-EMA coefficient of $\beta_s = 0.95$. The dual weights are updated every five communication rounds after all clients have satisfied the dual warm-up condition. The dual signal is constructed from undiscounted completed-episode returns and standardized using the smoothed cross-environment return scale, as described in Section E.2.

AMPO-PPO(U) uses the same local PPO configuration and server actor step size as AMPO-PPO(A), but fixes $\lambda_k = 1/K$ throughout training. It therefore does not use the adaptive dual step size η_λ or the dual-specific EMA parameters.

Table 8: Algorithm-specific hyperparameters and tuning ranges.

Parameter	PPOAvg	FedSVRPG-M-PPO	AMPO-PPO
Server update weight	1.0	$\{0.5, 0.7, 1.0\}$	–
Momentum coefficient β_{mom}	–	$\{0.85, 0.90, 0.95\}$	–
Server actor step size η_π	–	–	$\{10^{-3}, 3 \times 10^{-3}\}$
Dual step size η_λ	–	–	$\{10^{-4}, 3 \times 10^{-4}\}$