SHARE FIRST, SPECIALIZE LATER: TEMPORAL PARAMETER SHARING IN MULTI-AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In multi-agent reinforcement learning (MARL) behavioral diversity among agents is crucial when a task requires them to pursue distinct goals, cover different areas, or take on asymmetric roles. Existing methods induce this diversity by modifying the training objective or the policy architecture, often at the cost of extra structural and hyperparameter complexity. We instead induce diversity through training dynamics alone, via *temporal parameter sharing*: agents first pre-train a single shared, homogeneous policy, then split into independently trained, heterogeneous policies initialized from it. We formalize the choice of how long to pre-train homogeneously as a bilevel optimization problem. Since its non-convex lower level cannot be differentiated through, we solve it with Bayesian optimization over the switch point — an approach we call **BOSS** (Bayesian Optimization for Shared-to-Split training). Across three multi-agent tasks (dispersion, navigation, and predator-prey) with two MARL algorithms (IPPO and MADDPG), BOSS matches or even exceeds DiCo, a state-of-the-art diversity-control baseline evaluated at its own best published configuration without any change to the policy architecture. The variation in the switch point found by BOSS tracks how symmetric each task's underlying coordination problem is, positioning the duration of homogeneous pre-training as an interpretable, previously underappreciated diagnostic of task structure rather than a hyperparameter to tune away.

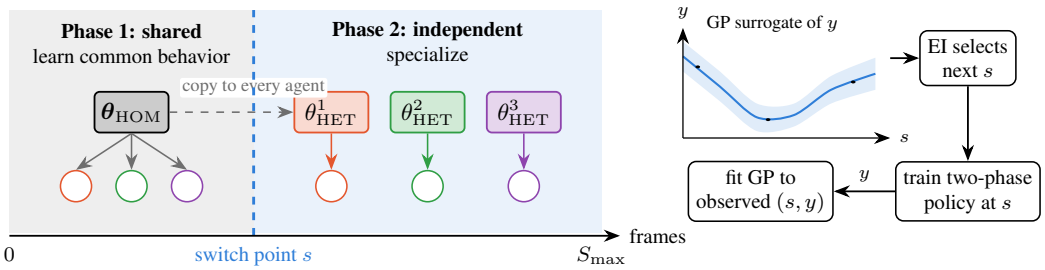
1 INTRODUCTION

Multi-agent reinforcement learning (MARL) studies decision-making problems in which multiple agents learn and act in a shared environment. Training MARL policies is a non-convex optimization problem, and a central challenge is balancing coordination (homogeneity) against specialization (heterogeneity). A common way to exploit homogeneity is parameter sharing, in which all agents use a single policy network (Gupta et al., 2017; Terry et al., 2020): it improves sample efficiency but biases agents toward identical behavior, limiting specialization (Christianos et al., 2021).

Existing approaches encourage diversity through intrinsic rewards (Jaques et al., 2019; Jiang and Lu, 2021; Li et al., 2021), auxiliary losses (Mahajan et al., 2019; Wang et al., 2020), or explicit agent policy decomposition (Bettini et al., 2024a; Fu et al., 2026; Qin et al., 2025; Tessera et al., 2026). Although effective, these methods resolve the homogeneity–heterogeneity trade-off statically: it is fixed in the architecture or the objective and held constant throughout training.

In this work, we take a different perspective: we treat the trade-off as a *temporal* choice. The question is not *whether* agents should share parameters, but *for how long*. We instantiate this view as **temporal parameter sharing**, a two-phase framework of homogeneous pre-training followed by heterogeneous fine-tuning. In the first phase, agents share policy parameters to learn common behaviors; in the second, each agent is initialized from the shared policy and trained independently to develop specialized strategies. We formalize this process as a bilevel optimization problem, in which the shared parameters $\boldsymbol{\theta}_{\mathrm{HOM}}$ learned in the first phase are the upper-level variable: they determine the initialization, and hence the final solution quality, of the lower-level heterogeneous (non-convex) problem. Standard gradient-based bilevel methods (Colson et al., 2007; Sinha et al., 2018) are ill-suited here. The lower level is solved by thousands of stochastic policy-gradient updates on a

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.



(a) Two-phase training with temporal parameter sharing

(b) BOSS: Bayesian optimization of s

Figure 1: Overview of BOSS. (a) Temporal parameter sharing: all agents share one policy $\boldsymbol{\theta}_{\text{HOM}}$ for the first s frames; then each agent is initialized from it and trained independently for the remaining $S_{\max} - s$ frames. (b) Bayesian optimization over the switch point: each training run at s yields an observed objective y (the negative final return; Section 4.2), and a GP surrogate of y guides Expected Improvement to the next s to evaluate.

non-convex objective, so it neither reaches a stationary point suitable for implicit differentiation nor admits tractable differentiation through its unrolled updates. We therefore parameterize the upper level by a single scalar: the *switch point* s, i.e., the number of training frames spent in pre-training before switching to independent fine-tuning. We treat the choice of s as a black-box optimization problem and solve it with Bayesian optimization (BO) using a Gaussian-process surrogate. We call the resulting method **BOSS** (Bayesian Optimization for Shared-to-Split training; Figure 1).

We compare BOSS against DiCo (Bettini et al., 2024a) on three multi-agent tasks (dispersion, navigation, and predator-prey), asking how many frames of shared, homogeneous learning each task needs before independent, heterogeneous learning should take over. We find that this balance is decisive: it is not a minor tuning knob, but the difference between matching a state-of-the-art diversity-control baseline and falling well short of it.

With a well-chosen switch point, BOSS matches or outperforms DiCo, which relies on a dedicated policy architecture, on every task, without any architectural change of our own. Shared pre-training is thus an effective training mechanism in its own right. We further find that the optimal switch point is strongly task-dependent. When agents pursue individually distinct goals from the outset (navigation), switching almost immediately is best (0.5% of the training budget). When agents must break symmetry over shared targets (dispersion), a longer shared phase pays off (21% of the budget). The switch point is therefore not merely a hyperparameter to tune away: it is an interpretable diagnostic of how much of a task's optimal behavior is shared across agents.

We summarize our contributions as follows:

- **Reframing the homogeneity–heterogeneity trade-off as a temporal choice.** Unlike existing approaches, which fix the trade-off in the learning objective (auxiliary diversity losses, intrinsic rewards) or the policy architecture (explicit shared/individual parameter decompositions), we treat *how long* agents train with shared parameters before specializing as the quantity to optimize.
- **A bilevel-optimization formulation of the shared/independent training balance.** We formalize two-phase training as a bilevel optimization problem (Definition 1): the upper level selects the shared pre-trained parameters, which initialize the lower-level, non-convex training of independent heterogeneous policies. We show why standard gradient-based bilevel methods are ill-suited to this setting and parameterize the upper level by a single scalar, the switch point s, which motivates a black-box treatment.
- **Bayesian optimization as a practical solver.** We treat switch-point selection as black-box optimization over $s \in [0, S_{\max}]$ and solve it with a Gaussian-process surrogate and Expected Improvement (Section 4.2), locating a strong switch point per task within a budget of eight training runs.
- **Empirical evidence that the switch point matters and is interpretable.** Across three multi-agent tasks, BOSS matches or outperforms DiCo (Bettini et al., 2024a) without any

architectural change. The optimal switch point varies widely across tasks, from almost immediate splitting in navigation (0.5% of the budget) to a substantial shared phase in dispersion (21%), and reflects how much of each task's optimal behavior is shared across agents.

The paper is organized as follows. Section 2 introduces background on reinforcement learning and multi-agent reinforcement learning. Section 3 reviews related work on behavior diversity in MARL. Section 4 formalizes our two-phase training scheme as a bilevel optimization problem and presents our Bayesian optimization solution. Section 5 reports our empirical study, and Section 6 concludes.

2 Background

2.1 Reinforcement Learning

In reinforcement learning (RL), an agent learns by trial and error to act so as to maximize its cumulative reward. Its behavior is given by a policy π_θ, typically a neural network with parameters θ, and learning solves the non-convex problem

$$\max_{\theta} \quad \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{H} R(s_t, a_t)\right], \tag{1}$$

where τ is a trajectory sampled by running π_θ, s_t and a_t are the state and action at time step t, H is the episode horizon, and R is the reward function. (In practice, returns are estimated with a discount factor $\gamma \in (0, 1]$.) Policy-gradient methods (Williams, 1992) solve this problem by stochastic gradient ascent on sampled trajectories. Proximal Policy Optimization (PPO) (Schulman et al., 2017), the most widely used of these, maximizes a clipped surrogate objective that limits how far each update can move the policy, together with a learned critic that estimates advantages.

2.2 Multi-Agent Reinforcement Learning

In multi-agent reinforcement learning (MARL), multiple agents learn in a shared environment; because the other agents' policies change as they learn, each agent faces a non-stationary environment. We model the problem as a Partially Observable Markov Game (POMG) (Littman, 1994) $\langle \mathcal{N}, \mathcal{S}, \{\mathcal{O}_i\}_{i\in\mathcal{N}}, \{\mathcal{A}_i\}_{i\in\mathcal{N}}, \{\mathcal{R}_i\}_{i\in\mathcal{N}}, \mathcal{T}, \gamma\rangle$, where $\mathcal{N} = \{1, \dots, n\}$ is the set of agents, $\mathcal{S}$ the global state space, $\mathcal{O}_i$, $\mathcal{A}_i$ and $\mathcal{R}_i$ agent i's observation space, action space and reward function, $\mathcal{T}$ the stochastic transition model over the joint action $\mathbf{a}_t = (a_t^1, \dots, a_t^n)$, and γ the discount factor. Each agent i acts according to a policy $\pi_i(a_i \mid o_i)$ with parameters $\theta_i \in \Theta$, where Θ is the parameter space of a single agent's policy network, common to all agents because they share the same architecture. In sampling and in the adversary group of predator-prey (Section 5), all agents in a group receive the same reward, $\mathcal{R}_i = \mathcal{R}$, which we write as $R(s_t, \mathbf{a}_t)$; in navigation and dispersion, each agent receives its own reward $\mathcal{R}_i(s_t, \mathbf{a}_t)$.

Because every agent's policy lives in the same space Θ, agents can be trained either with a single set of shared parameters ($\theta_1 = \cdots = \theta_n$) or with independent, per-agent parameters (Gupta et al., 2017; Terry et al., 2020; Christianos et al., 2021). Sharing improves sample efficiency, since experience from any agent updates the same weights, but biases all agents toward the same behavior; independent parameters let agents specialize, at the cost of that efficiency. This tension is the central design axis of this paper: Section 4 formalizes it as a bilevel optimization problem over *when*, rather than *whether*, agents share parameters.

Following DiCo (Bettini et al., 2024a), which chose each task's algorithm according to the task's needs (e.g., reward sparsity), we use IPPO (de Witt et al., 2020) for navigation and predator-prey, MADDPG (Lowe et al., 2017) for dispersion, and IDDPG for sampling. IPPO applies PPO to each agent independently, with stochastic policies. MADDPG and IDDPG extend DDPG (Lillicrap et al., 2016), an off-policy actor-critic method with deterministic policies, to multiple agents: MADDPG trains each agent's critic centrally on all agents' observations and actions, while IDDPG trains each agent fully independently. Whether agents share parameters is orthogonal to the choice of algorithm. This setup keeps the comparison with DiCo like-for-like and spans on-policy and off-policy algorithms as well as individual and shared rewards, so our findings are not an artifact of any one algorithm or reward design.

3 Related Work: Behavior Diversity in MARL

Many multi-agent tasks require agents to behave differently in order to cooperate effectively and maximize the team reward. Parameter sharing can produce some behavioral variation by conditioning a shared policy on agent-specific inputs such as agent IDs (Terry et al., 2020), but a single shared network still biases agents toward similar behavior, and selectively sharing parameters among groups of agents can improve on full sharing (Christianos et al., 2021). The methods below promote diversity by changing either the learning objective or the policy architecture.

Diversity through the objective. One line of work adds intrinsic rewards for diverse behavior, such as rewards for social influence between agents' actions (Jaques et al., 2019) or for how identifiable an agent is from its observations (Jiang and Lu, 2021). Li et al. (2021) combine shared and agent-specific network components with a mutual-information objective between an agent's identity and its trajectory. Another line adds auxiliary losses: MAVEN (Mahajan et al., 2019) maximizes mutual information between trajectories and a latent variable, and ROMA (Wang et al., 2020) regularizes agents toward learning specialized roles. Because these diversity terms are added to the task objective, they can conflict with it and typically require task-specific tuning of their weights.

Diversity through the architecture. DiCo (Bettini et al., 2024a), our main baseline, represents each agent's policy as the sum of a shared (homogeneous) component π_h and an agent-specific (heterogeneous) deviation $\pi_{h,i}$, rescaled to a desired level of diversity:

$$\pi_i(o) = \pi_h(o) + \frac{\mathrm{SND}_{\mathrm{des}}}{\widehat{\mathrm{SND}}}\,\pi_{h,i}(o), \tag{2}$$

where $\widehat{\mathrm{SND}}$ is the measured System Neural Diversity of the deviations $\{\pi_{h,i}\}_{i\in\mathcal{N}}$ (the average pairwise Wasserstein distance between agents' action distributions) and $\mathrm{SND}_{\mathrm{des}}$ is the target. DiCo thus controls diversity exactly without modifying the learning objective, but requires choosing $\mathrm{SND}_{\mathrm{des}}$ for each task. GradPS (Qin et al., 2025) instead addresses gradient conflict: it identifies neurons whose updates are largely cancelled by conflicting gradients across agents and clones them, assigning each clone to a group of agents with low mutual conflict, so that diversity arises as a by-product of resolving conflict. HyperMARL (Tessera et al., 2026) keeps full sharing but generates agent-specific actor and critic weights with an agent-conditioned hypernetwork, which decouples observation- and agent-conditioned gradients; homogeneous, heterogeneous, or mixed behaviors can emerge, but the level of diversity cannot be specified directly. HPS (Fu et al., 2026) observes that agents' gradients tend to agree in direction but conflict in scale; it constrains a shared backbone to the unit hypersphere and restores heterogeneous response magnitudes with agent-specific scale-and-shift factors.

Positioning of our work. All of the methods above fix the homogeneity–heterogeneity trade-off in the objective or the architecture, and keep that choice constant throughout training. We leave the objective and the architecture unchanged and instead vary *when* agents stop sharing parameters: agents train with full sharing up to a switch point and independently afterwards. Choosing this switch point is a problem of optimizing a training schedule, which relates our approach to hyperparameter optimization for deep learning, including Bayesian optimization (Snoek et al., 2012) and population-based training (Jaderberg et al., 2017), which adapts hyperparameters during training. We use Bayesian optimization because each evaluation is a full, expensive MARL training run and the search space is a single scalar (Section 4.2).

4 Temporal Parameter Sharing as Bilevel Optimization

DiCo (Bettini et al., 2024a) combines homogeneity and heterogeneity *architecturally*, splitting each agent's policy into a shared component and an agent-specific one. We combine them *temporally* instead: the policy architecture is left unchanged, and agents first share parameters and then train independently. This section formalizes this two-phase scheme as a bilevel optimization problem; Section 4.2 presents the Bayesian optimization search that solves it.

4.1 TWO-PHASE TRAINING VIA TEMPORAL PARAMETER SHARING

We divide training into two phases. In the first *homogeneous* phase, all agents share a single set of policy parameters and jointly learn common behavior; we denote the resulting shared parameters $\boldsymbol{\theta}_{\mathrm{HOM}}$. In the second *heterogeneous* phase, each agent's policy is initialized from $\boldsymbol{\theta}_{\mathrm{HOM}}$ and then trained independently, allowing agent-specific behavior to emerge; we denote the resulting per-agent parameters $\boldsymbol{\theta}^*_{\mathrm{HET}}$. Because heterogeneous training begins from $\boldsymbol{\theta}_{\mathrm{HOM}}$, the quality of $\boldsymbol{\theta}^*_{\mathrm{HET}}$ is shaped by the outcome of the homogeneous phase. We formalize this dependency as a bilevel optimization problem.

Setup. As in Section 2, let $\mathcal{N} = \{1, \dots, n\}$ index the agents, Θ the parameter space of a single agent's policy, H the episode horizon, and $\mathcal{R}_i$ agent i's reward. A joint parameter vector $\boldsymbol{\theta} = (\theta^1, \dots, \theta^n) \in \Theta^n$ induces the joint policy $\boldsymbol{\pi}_{\boldsymbol{\theta}} = (\pi_{\theta^1}, \dots, \pi_{\theta^n})$, and $S_{\max}$ denotes the total training budget in environment frames. The quantity we ultimately care about is the average return of the agents,

$$L_U(\boldsymbol{\theta}) := -\,\mathbb{E}_{\tau \sim \boldsymbol{\pi}_{\boldsymbol{\theta}}}\left[\frac{1}{n}\sum_{i\in\mathcal{N}}\sum_{t=0}^{H}\mathcal{R}_i(s_t, \mathbf{a}_t)\right], \qquad L_U : \Theta^n \to \mathbb{R}, \tag{3}$$

which reduces to the shared return $R(s_t, \mathbf{a}_t)$ when all agents share a reward. During training, each agent i minimizes its own training loss $\ell_i(\theta^i; \boldsymbol{\theta}^{-i})$, given by whichever RL algorithm the task uses (the PPO clipped surrogate for IPPO, or a deterministic policy-gradient loss for MADDPG and IDDPG); ℓ_i depends on the other agents' parameters $\boldsymbol{\theta}^{-i}$ through the environment and, for MADDPG, the centralized critic. Nothing in this section depends on the particular choice of ℓ_i.

We describe the two phases as training operators. Let $\Theta_{\mathrm{HOM}} := \{\boldsymbol{\theta} \in \Theta^n : \theta^1 = \cdots = \theta^n\} \cong \Theta$ be the set of fully shared parameter assignments. The *shared-training operator* $\mathcal{A}_{\mathrm{sh}}(\boldsymbol{\theta}_0; s) \in \Theta_{\mathrm{HOM}}$ runs the task's RL algorithm for s frames from a random initialization $\boldsymbol{\theta}_0 \in \Theta_{\mathrm{HOM}}$ with parameters tied across agents, so that every update keeps the parameters in Θ_{HOM}. The *independent-training operator* $\mathcal{A}_{\mathrm{ind}}(\boldsymbol{\theta}; B) \in \Theta^n$ starts every agent at its component of $\boldsymbol{\theta}$ and runs B frames of per-agent updates,

$$\theta^i_{\mathrm{HET}} \;\leftarrow\; \text{RL-update}\left(\theta^i_{\mathrm{HET}};\; \ell_i(\theta^i_{\mathrm{HET}}; \boldsymbol{\theta}^{-i}_{\mathrm{HET}})\right), \qquad \forall i \in \mathcal{N}, \tag{4}$$

with parameters untied across agents, so that agents are free to diverge.

Definition 1 (Two-Phase Bilevel Training Problem)**.** *Given a lower-level budget B, the* two-phase bilevel training problem *is*

$$\min_{\boldsymbol{\theta}_{\mathrm{HOM}} \in \Theta_{\mathrm{HOM}}} \; L_U\big(\boldsymbol{\theta}^*_{\mathrm{HET}}(\boldsymbol{\theta}_{\mathrm{HOM}})\big) \tag{5}$$

$$\textit{s.t.} \quad \boldsymbol{\theta}^*_{\mathrm{HET}}(\boldsymbol{\theta}_{\mathrm{HOM}}) \;=\; \mathcal{A}_{\mathrm{ind}}(\boldsymbol{\theta}_{\mathrm{HOM}}; B). \tag{6}$$

The upper-level variable $\boldsymbol{\theta}_{\mathrm{HOM}}$ is restricted to the shared set Θ_{HOM}, while the lower-level solution $\boldsymbol{\theta}^*_{\mathrm{HET}}$ is initialized at $\boldsymbol{\theta}_{\mathrm{HOM}}$ but is free to leave Θ_{HOM}.

Following bilevel formulations of meta-learning (Finn et al., 2017; Rajeswaran et al., 2019), we define the lower-level solution as the *output of a training procedure*, not as an exact minimizer $\arg\min_{\boldsymbol{\theta}_{\mathrm{HET}}} \sum_i \ell_i$. This distinction matters here. The lower-level objective does not depend on $\boldsymbol{\theta}_{\mathrm{HOM}}$ at all, so its set of exact minimizers would be the same for every upper-level choice. $\boldsymbol{\theta}_{\mathrm{HOM}}$ influences the outcome only by determining where non-convex training starts, and hence which solution it reaches. Solving Equation 5 therefore amounts to finding the initialization from which independent training reaches a good solution.

Why gradient-based bilevel methods do not apply. The dominant solution methods for bilevel optimization are gradient-based (Colson et al., 2007; Sinha et al., 2018): they differentiate the upper-level objective through the lower-level solution, either by unrolling the lower-level optimization or by applying the implicit function theorem at a lower-level stationary point. Neither route is practical here. The lower level consists of thousands of stochastic policy-gradient updates on a non-convex, non-smooth objective (PPO's clipped surrogate under stochastic environment sampling), so it does not settle at a stationary point suitable for implicit differentiation, and its trajectory is far too long to differentiate through.

Reducing the search to the switch point. Searching over Θ_{HOM} directly is also infeasible: it is the full, high-dimensional space of network parameters, and each evaluation of Equation 5 costs a

full training run. We therefore restrict the upper level to the candidates that shared training actually produces, $\boldsymbol{\theta}_{\mathrm{HOM}}(s) := \mathcal{A}_{\mathrm{sh}}(\boldsymbol{\theta}_0; s)$ for $s \in [0, S_{\max}]$, and give the lower level the remaining budget $B = S_{\max} - s$, so that every candidate uses the same total budget. The upper-level search then collapses to a single scalar, the *switch point* s. Section 4.2 searches over s with Bayesian optimization, and Algorithm 1 in Appendix A gives the full two-phase training procedure for a given s.

4.2 Solving the Bilevel Problem via Bayesian Optimization

Because gradient-based bilevel methods do not apply (Section 4), we treat the search over switch points as black-box optimization and solve it with Bayesian optimization (BO) (Snoek et al., 2012). BO only queries the *outcome* of a full two-phase training run, treating everything inside the run as an opaque function evaluation, so it never needs to differentiate through the lower level. Combined with the two-phase training scheme of Section 4, this search completes BOSS.

The switch-point objective. With the upper level restricted to the shared checkpoints $\boldsymbol{\theta}_{\mathrm{HOM}}(s)$ and the lower-level budget set to $B = S_{\max} - s$ (Section 4), Equation 5 becomes

$$\min_{s \in [0,\, S_{\max}]} f(s), \qquad f(s) \;:=\; L_U\big(\mathcal{A}_{\mathrm{ind}}\big(\boldsymbol{\theta}_{\mathrm{HOM}}(s);\, S_{\max} - s\big)\big)\,, \tag{7}$$

where each evaluation of f costs one full two-phase training run.

Gaussian process surrogate. BO maintains a probabilistic surrogate of f; we use a Gaussian process (GP) (Rasmussen and Williams, 2006),

$$f(s) \;\sim\; \mathcal{GP}\big(\mu(s),\, k(s, s')\big), \tag{8}$$

with mean function μ and covariance kernel k, and update its posterior after each new observation. Because each observation comes from a single training seed, we model observations as noisy, $y_i = f(s_i) + \epsilon_i$ with $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$.

Acquisition function. Given the current posterior, an acquisition function scores each candidate switch point by how worthwhile it is to evaluate next, trading off exploitation of points the model believes are good against exploration of points it is uncertain about. We use Expected Improvement (EI) (Jones et al., 1998),

$$\mathrm{EI}(s) \;=\; \mathbb{E}\big[\max(f_{\text{best}} - f(s),\, 0)\big], \tag{9}$$

where f_{best} is the best (lowest) value observed so far, and query next the switch point

$$s_{n+1} \;=\; \arg\max_{s \in [0,\, S_{\max}]} \mathrm{EI}(s). \tag{10}$$

Procedure. Figure 1(b) illustrates the search, and Algorithm 1 in Appendix A gives it in full. The inner subroutine TwoPhaseTrain evaluates f at one switch point by running shared training for s frames and independent training for the remaining $S_{\max} - s$ frames. The outer loop starts from a small random initial design and then queries the EI maximizer until a fixed budget of training runs is spent; in our experiments we use three random initial points followed by five EI-guided queries, i.e., eight training runs per task (Section 5). Because the shared phase does not depend on s beyond its length, shared-phase checkpoints can be cached and reused across queries: a query at s can resume from any cached checkpoint at $s' \le s$, which reduces the cost of each evaluation.

Stopping. We use a fixed budget of training runs because each evaluation of f is expensive and the budget can be matched to available compute. An adaptive alternative is to stop once $\max_s \mathrm{EI}(s)$ falls below a small threshold, indicating that no untried switch point is likely to improve meaningfully on the current best.

5 Experiments

We evaluate BOSS on three tasks from the VMAS simulator (Bettini et al., 2022): dispersion, navigation, and predator-prey (Appendix B); results on a fourth task, sampling, are reported in Appendix D. Training uses TorchRL (Bou et al., 2023) and BenchMARL (Bettini et al., 2024b). We compare against DiCo (Bettini et al., 2024a) at each task's best published diversity target (`desired_snd`), and where available, against two naive controls: full parameter sharing (FuPS),

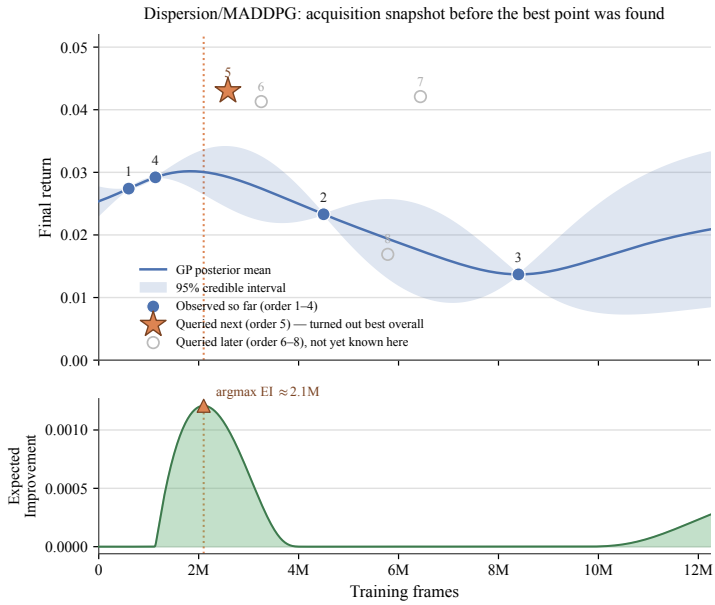


Figure 2: Dispersion (MADDPG): BOSS's GP surrogate (Matérn-5/2 kernel) after the first four evaluations (solid circles). The bottom panel shows the resulting Expected Improvement, which peaks near 2,586,000 frames, the point BOSS queried next (star) and the best switch point found. Points 6–8 (hollow circles), queried later, are shown for context only.

where all agents share one policy for the whole budget, and no parameter sharing (NoPS), where agents train independently from the start. On each task, BOSS searches for the switch point with eight single-seed training runs (three random, five EI-guided; Section 4.2); we then re-evaluate the best switch point found, together with DiCo, over 3 seeds. We report the per-agent, per-step reward at the end of training; the search minimizes its negative, y. Sections 5.1 and 5.2 present dispersion and navigation in detail, and Section 5.3 summarizes all three tasks, with predator-prey detailed in Appendix C.

5.1 Dispersion: A Balanced Middle Ground

In Dispersion, $n = 4$ agents start at the center of a 2D workspace and n food particles are placed at random. Each agent receives a reward of 1 for each particle it eats (rewards are individual, not shared) and observes its relative position to every particle and whether each has been eaten. An ideal team disperses so that each agent pursues a different particle. We train with MADDPG over a 12,240,000-frame budget, with DiCo at `desired_snd=6`.

Sharing for the whole budget performs worst (0.0109): a single policy struggles to break the symmetry needed for agents to spread across distinct targets. Sharing nothing reaches 0.0272, still short of DiCo. The search places the best switch point in the middle of the range, at 2.586M frames (21.1% of the budget); switching before 10% or after two-thirds of the budget scores far lower. The search is also efficient: after only four evaluations, the GP's Expected Improvement already peaked close to the point BOSS queried next, which turned out to be the best overall (Figure 2). Over 3 seeds (Figure 3), BOSS reaches 0.0419 against 0.0335 for DiCo, beating DiCo by 25%, NoPS by 54%, and FuPS by nearly $4\times$. DiCo's 3-seed mean is also lower than its single-seed value (0.0376), a reminder that single-seed results can read optimistic. Dispersion thus needs a substantial shared phase of about one-fifth of the budget: both too little and too much sharing leave much performance on the table.

5.2 Navigation: Splitting Almost Immediately

In Navigation, two agents are each assigned their own goal and receive individual rewards for progress toward and reaching it; each observes its own position and velocity and the relative positions of

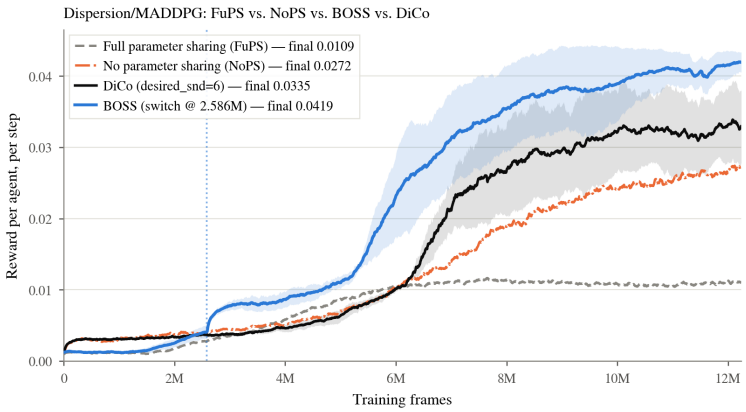


Figure 3: Dispersion (MADDPG): per-agent, per-step reward vs. training frames for FuPS (dashed gray), NoPS (dash-dot orange), DiCo (`desired_snd=6`, solid black), and BOSS (solid blue, switch at 2,586,000 frames, dotted line). DiCo and BOSS are averaged over 3 training seeds, with shaded bands showing ± 1 std; FuPS and NoPS are single-seed. All curves are EMA-smoothed ($\alpha = 0.9$) for readability.

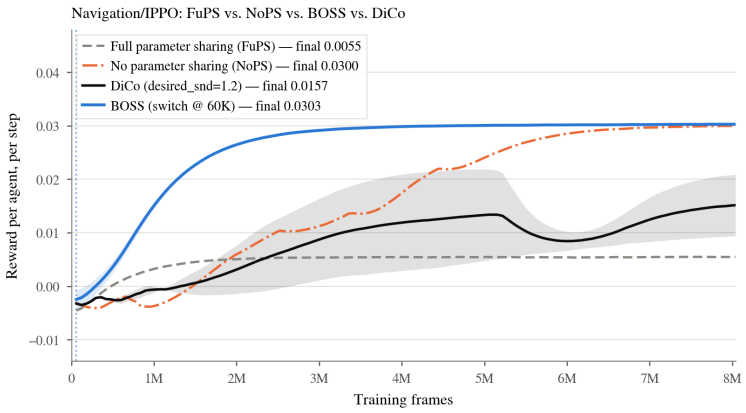


Figure 4: Navigation (IPPO): per-agent, per-step reward vs. training frames over an 8,000,000-frame budget for FuPS (dashed gray, final 0.0055), NoPS (dash-dot orange, final 0.0300), DiCo (`desired_snd=1.2`, solid black, final 0.0157), and BOSS (solid blue, switch at 60,000 frames, final 0.0303). DiCo and BOSS are averaged over 3 training seeds, with shaded bands showing ± 1 std; FuPS and NoPS are single-seed. All curves are EMA-smoothed ($\alpha = 0.9$) for readability.

all agents and goals. We train with IPPO, with DiCo at `desired_snd=1.2`. The search runs over a 12,000,000-frame budget (Table 4 in Appendix E); the four regimes are then compared over 8,000,000 frames (Figure 4).

Sharing for the whole budget performs worst (0.0055): with distinct per-agent goals, a single shared policy cannot specialize. NoPS reaches 0.0300 but slowly, and is still rising at 8M frames. DiCo reaches only 0.0157, with a curve that dips between 5M and 6M frames. The search is unambiguous: every switch point at or below 4% of the budget reaches 0.030–0.0305, while every point at 10% or more drops to 0.007–0.022. The best point shares for only 0.5% of the budget (60,000 frames) and reaches 0.0303, marginally above NoPS but 93% above DiCo and $5.5\times$ FuPS. Its main benefit is speed: BOSS plateaus by about 3M frames, while NoPS is still climbing at 8M. In contrast to dispersion, agents here have little behavior in common, so a very short shared phase suffices; it gives both agents basic goal-directed movement before they specialize and mainly buys faster convergence.

Table 1: Cross-task summary of the best switch point found by Bayesian optimization, as a fraction of that task's total frame budget, and its improvement over the DiCo baseline. BOSS and DiCo values are 3-seed means.

Task	Algorithm	Best switch (% budget)	Best return	vs. DiCo
Dispersion	MADDPG	2.59M (21%)	0.0419	+25%
Navigation	IPPO	60K (0.5%)	0.0303	+93%
Predator-prey	IPPO	7.26M (61%)	0.5921	+8%

5.3 Cross-Task Summary

Table 1 summarizes the three tasks. BOSS outperforms DiCo at its best published setting on every task, by margins from $+8\%$ (predator-prey) to $+93\%$ (navigation), without any architectural change. In predator-prey (Appendix C), two predators chase a single prey; the searched switch point (61% of the budget) beats DiCo over 3 seeds (0.5921 vs. 0.5478) but falls below full sharing (0.6423), so on this task never splitting is best, and the single-seed search overestimated the benefit of splitting.

The best amount of sharing thus varies widely across tasks and tracks how much of the optimal behavior agents can share. When each agent has its own goal from the outset (navigation), switching almost immediately is best; when agents must first learn common behavior before breaking symmetry over shared targets (dispersion), a substantial shared phase pays off; and when agents solve a tightly coupled, symmetric problem (predator-prey), sharing throughout is best. No single default switch point fits all three tasks: the best switch points found span two orders of magnitude in frames (60K to 7.26M), yet eight training runs per task were enough to find one that beats DiCo on every task.

Limitations. Each search evaluation uses a single training seed, so individual observations are noisy. We therefore re-evaluated the best switch point and DiCo over 3 seeds on every task; on predator-prey, this reversed the single-seed comparison with full sharing, which suggests evaluating more seeds per query or always including full sharing ($s = S_{\max}$) as a candidate.

6 Conclusions

We proposed temporal parameter sharing, which treats the homogeneity–heterogeneity trade-off in MARL as a question of *when* agents stop sharing parameters rather than *whether* they share them. We formalized the resulting two-phase training as a bilevel optimization problem and introduced BOSS, which searches over the switch point with Bayesian optimization. Across three VMAS tasks, BOSS outperforms DiCo on every task without any architectural change, and the best amount of sharing ranges from almost none (navigation) to a substantial shared phase (dispersion) to sharing throughout (predator-prey), reflecting how much of each task's optimal behavior is shared across agents. Our predator-prey results also show that single-seed evaluations can overstate the benefit of splitting, so each search query should be evaluated robustly. In future work, we plan to make the search cheaper by exploiting the training prefix that queries share, and to extend the approach beyond a single switch point, for example to gradual or group-wise splitting.

AI Use Statement

In this work, we used generative AI tools (Claude, Anthropic) for drafting and revising prose throughout the paper, for readability and copy-editing, for finding and formatting references, and for drawing the overview schematic (Figure 1). We also used generative AI to help refine the presentation of the method: it suggested revisions to the formal statement of the bilevel problem (in particular, defining the lower-level solution as the output of the training procedure, Definition 1) and to the argument for why gradient-based bilevel methods do not apply, and it drafted parts of the text that interpret our experimental results. In addition, we used generative AI for minor assistance with experiment code, such as plotting scripts and small implementation snippets. The research idea, the two-phase training method, the use of Bayesian optimization, the experimental design, and all experiments are the authors' own work. We have reviewed all AI-assisted work: we checked every AI-suggested mathematical statement and equation, verified all AI-drafted interpretations against our experimental results, checked all AI-suggested references against their original sources, and re-ran

and tested any code that generative AI touched. We take responsibility for the final content of this paper, including all text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *Autonomous Agents and Multiagent Systems (AAMAS 2017 Workshops)*, volume 10642 of *Lecture Notes in Computer Science*, pages 66–83. Springer, 2017.

Justin K. Terry, Nathaniel Grammel, Ananth Hari, Luis Santos, and Benjamin Black. Revisiting parameter sharing in multi-agent deep reinforcement learning. *arXiv preprint arXiv:2005.13625*, 2020.

Filippos Christianos, Georgios Papoudakis, Muhammad A. Rahman, and Stefano V. Albrecht. Scaling multi-agent reinforcement learning with selective parameter sharing. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1989–1998. PMLR, 2021.

Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, DJ Strouse, Joel Z Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International conference on machine learning*, pages 3040–3049. PMLR, 2019.

Jiechuan Jiang and Zongqing Lu. The emergence of individuality. In *International conference on machine learning*, pages 4992–5001. PMLR, 2021.

Chenghao Li, Tonghan Wang, Chengjie Wu, Qianchuan Zhao, Jun Yang, and Chongjie Zhang. Celebrating diversity in shared multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:3991–4002, 2021.

Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. MAVEN: Multi-agent variational exploration. *Advances in neural information processing systems*, 32, 2019.

Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. ROMA: Multi-agent reinforcement learning with emergent roles. *arXiv preprint arXiv:2003.08039*, 2020.

Matteo Bettini, Ryan Kortvelesy, and Amanda Prorok. Controlling behavioral diversity in multi-agent reinforcement learning. In *Forty-first International Conference on Machine Learning*, 2024a. URL `https://openreview.net/forum?id=qQjUgItPq4`.

Hu Fu, Pengyi Li, Hao Chen, Xuanyu Xiang, Biao Luo, and Yihua Tan. HPS: Hyperspherical parameter sharing for efficient multi-agent reinforcement learning. In *Forty-third International Conference on Machine Learning*, 2026.

Haoyuan Qin, Zhengzhu Liu, Chenxing Lin, Chennan Ma, Songzhu Mei, Siqi Shen, and Cheng Wang. GradPS: Resolving futile neurons in parameter sharing network for multi-agent reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025.

Kale-ab Tessera, Muhammad Arrasy Rahman, Amos Storkey, and Stefano Albrecht. HyperMARL: Adaptive hypernetworks for multi-agent RL. *Advances in Neural Information Processing Systems*, 38:114352–114398, 2026.

Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153(1):235–256, 2007.

Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. A review on bilevel optimization: from classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22 (2):276–295, 2018.

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

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

Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 157–163. Morgan Kaufmann, 1994.

Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H S Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the StarCraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in Neural Information Processing Systems*, 30, 2017.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *International Conference on Learning Representations*, 2016.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25, pages 2951–2959, 2012.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, Chrisantha Fernando, and Koray Kavukcuoglu. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. *Advances in Neural Information Processing Systems*, 32, 2019.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. VMAS: A vectorized multi-agent simulator for collective robot learning. In *International Symposium on Distributed Autonomous Robotic Systems*, pages 42–56. Springer, 2022.

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. *arXiv preprint arXiv:2306.00577*, 2023.

Matteo Bettini, Amanda Prorok, and Vincent Moens. BenchMARL: Benchmarking multi-agent reinforcement learning. *Journal of Machine Learning Research*, 25(217):1–10, 2024b.

A BOSS ALGORITHM

Algorithm 1 gives the complete BOSS procedure described in Section 4.2.

Algorithm 1 BOSS: Bayesian optimization of the switch point with two-phase training (Definition 1).

function TWOPHASETRAIN(s)
  $\boldsymbol{\theta}_{\mathrm{HOM}} \leftarrow \boldsymbol{\theta}_0 \in \Theta_{\mathrm{HOM}}$ ▷ random shared initialization
  while frames collected $< s$ **do** ▷ shared phase: $\mathcal{A}_{\mathrm{sh}}(\boldsymbol{\theta}_0; s)$
    $\boldsymbol{\theta}_{\mathrm{HOM}} \leftarrow$ RL update with parameters shared across agents ▷ IPPO / MADDPG / IDDPG
  end while
  $\theta^i_{\mathrm{HET}} \leftarrow \boldsymbol{\theta}_{\mathrm{HOM}}$ for all $i \in \mathcal{N}$ ▷ switch: untie parameters
  while frames collected $< S_{\max} - s$ **do** ▷ lower level: $\mathcal{A}_{\mathrm{ind}}$, Eq. 6
    for $i \in \mathcal{N}$ **do**
      $\theta^i_{\mathrm{HET}} \leftarrow \mathrm{RL\text{-}update}\left(\theta^i_{\mathrm{HET}};\ \ell_i(\theta^i_{\mathrm{HET}}; \boldsymbol{\theta}^{-i}_{\mathrm{HET}})\right)$
    end for
  end while
  return $y = L_U(\boldsymbol{\theta}_{\mathrm{HET}})$ ▷ Equation 3
end function

Require: Budget of N_0 random initial points and T EI-guided queries
Ensure: Best switch point s^*
$\mathcal{D} \leftarrow \{(s_j, \mathrm{TWOPHASETRAIN}(s_j))\}_{j=1}^{N_0}$ with s_j drawn at random from $[0, S_{\max}]$
for $t = 1, \ldots, T$ **do**
  Fit the GP posterior $f \mid \mathcal{D}$ (Equation 8)
  $s' \leftarrow \arg\max_{s \in [0,\, S_{\max}]} \mathrm{EI}(s)$ ▷ Equation 10
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s', \mathrm{TWOPHASETRAIN}(s'))\}$
end for
return $s^* \leftarrow \arg\min_{(s_j,\, y_j) \in \mathcal{D}} y_j$

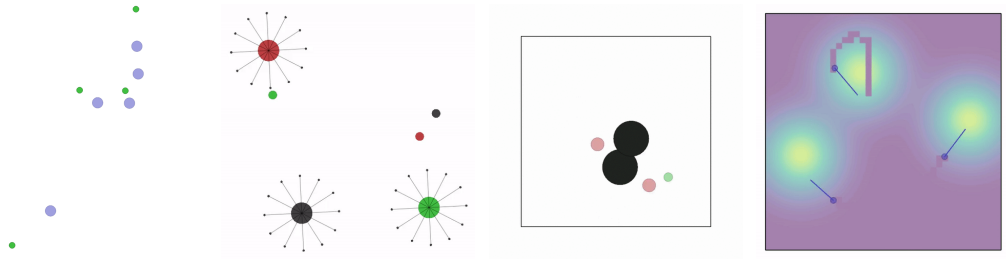
B TASKS AND EXPERIMENTAL SETUP

Figure 5 shows the four VMAS (Bettini et al., 2022) tasks used in this paper: dispersion, navigation, and predator-prey in Section 5, and sampling in Appendix D. The renders are taken from the simulator's scenario previews. In dispersion, green discs are agents and purple discs are the food particles they must spread out to collect. In navigation, each colored disc is an agent and the small disc of matching color is its goal; the radiating lines are each agent's range sensor. In predator-prey, the red discs are the two predators, the green disc is the prey, and the black discs are static obstacles. In sampling, agents (small circles with a heading line) move over the underlying reward distribution, with brighter regions indicating higher reward density. All experiments use TorchRL (Bou et al., 2023) and BenchMARL (Bettini et al., 2024b) and were run on NVIDIA A40 and L40S GPUs.

Hyperparameters. For IPPO, MADDPG, and IDDPG we use BenchMARL's default algorithm and model configurations, including network architectures, optimizer settings, and training hyperparameters, without task-specific tuning. The same configuration is used in both the shared and the independent phase of BOSS and for the FuPS and NoPS controls; the only quantities that differ across runs are the total frame budget $S_{\max}$ of each task (given in Section 5 and Appendices C and D) and, for BOSS, the switch point s. DiCo uses the same defaults with the per-task diversity target `desired_snd` from Bettini et al. (2024a).

C PREDATOR-PREY: SHARING THROUGHOUT

In predator-prey, two cooperating predators (the adversary group) chase a single prey in a near-zero-sum pursuit task, and both groups are trained with IPPO. Unlike the other tasks, the environment has two asymmetric agent groups; only the predator group can be split, since the prey group has a single member. Our runs train the predators against a prey that keeps learning throughout, whereas the DiCo



(a) Dispersion (b) Navigation (c) Predator-prey (d) Sampling

Figure 5: The four VMAS tasks used in this paper. Renders are taken from the simulator's scenario previews (Bettini et al., 2022).

baseline freezes the prey after 3,000,000 frames, a real difference in training dynamics between the two.

We compare full sharing of the predator policy for the entire 12,000,000-frame budget (FuPS), DiCo (`desired_snd=0.6`), and the BOSS search over the switch point, which optimizes the predators' reward averaged over the last five training iterations. We do not have a NoPS run for this task. Table 2 lists the eight switch points the search evaluated, in the order they were queried: the first three are the random initial design and the remaining five are GP + Expected Improvement proposals (Section 4.2). Figure 6 compares FuPS, DiCo, and the best switch point found.

Table 2: Predator-prey (IPPO) switch-point Bayesian optimization results (seed 0, 12M-frame total budget, adversary reward).

Switch point (frames)	% of budget shared	Final adversary return	BO order
60,000	0.5%	0.3351	1
5,460,000	45.5%	0.6260	2
10,800,000	90.0%	0.6510	3
10,320,000	86.0%	0.6373	4
7,260,000	60.5%	0.6785	5
7,980,000	66.5%	0.6659	6
6,660,000	55.5%	0.5985	7
8,700,000	72.5%	0.6199	8

Full sharing is a strong policy here, reaching 0.6423: unlike in navigation and dispersion, the two predators face a symmetric sub-problem (chasing the same prey), so there is little need for them to specialize. The search landscape reverses the pattern seen in navigation. The point that shares least (0.5% of the budget) scores by far the worst (0.3351), because the predators need substantial coordination before specializing. From about 45% of the budget onward, every point lands in a noisy 0.60–0.68 plateau with no sharp optimum. The best point in the search, at 60.5% of the budget, reached 0.6785 with a single seed; over 3 seeds it reaches 0.5921, still 8% above DiCo (0.5478, also over 3 seeds) but 8% below full sharing. Its single-seed value was therefore the top of a noisy plateau rather than a reliable optimum. Unlike in navigation and dispersion, where the best balance lies well short of full sharing, here it lies at the fully shared end: splitting the predators does not pay off.

D SAMPLING: A SHORT SHARED PHASE

In sampling, three agents trained with IDDPG search for reward peaks in the environment; collisions between agents are disabled, and episodes last 100 time steps. We compare DiCo (`desired_snd=5`) against the BOSS search over a 4,800,000-frame budget; we do not have FuPS or NoPS runs for this task, and all sampling results are single-seed. Table 3 lists the eight switch points the search evaluated, in the same format as Table 2, and Figure 7 compares DiCo with the best switch point found.

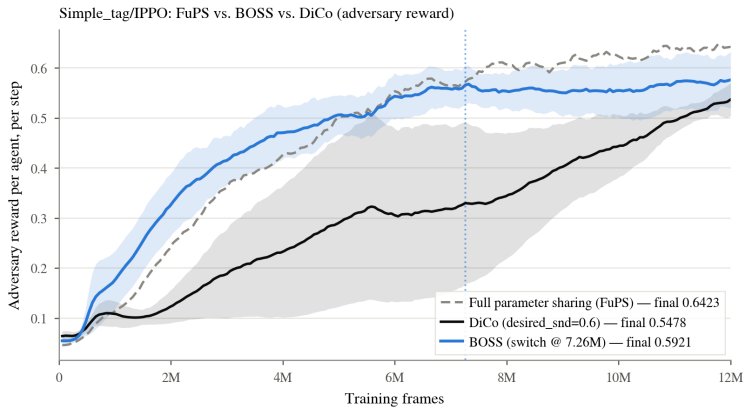


Figure 6: Predator-prey (IPPO): predator reward per agent, per step, vs. training frames for FuPS (dashed gray, final 0.6423), DiCo (`desired_snd=0.6`, solid black, final 0.5478), and the best switch point from Table 2 (solid blue, switch at 7,260,000 frames, final 0.5921). DiCo and BOSS are averaged over 3 training seeds, with shaded bands showing ± 1 std; FuPS is single-seed. All curves are EMA-smoothed ($\alpha = 0.9$) for readability.

Table 3: Sampling (IDDPG) switch-point Bayesian optimization results (seed 0, 4.8M-frame total budget).

Switch point (frames)	% of budget shared	Final return	BO order
6,000	0.1%	2.2266	1
2,106,000	43.9%	2.2160	2
4,200,000	87.5%	2.0620	3
210,000	4.4%	2.2671	4
882,000	18.4%	2.3260	5
1,260,000	26.3%	2.1817	6
690,000	14.4%	2.2704	7
978,000	20.4%	2.1730	8

The best point spends 18.4% of the budget sharing and reaches 2.3260, a modest 1.5% improvement over DiCo (2.2906). The search landscape is noisy and not unimodal: points form a plateau from about 4% to 18% of the budget that peaks at 18.4%, followed by a dip around 20–26%, a partial recovery near 44%, and a drop beyond 87%. The point that shares least (0.1% of the budget) scores respectably but below the 4–18% plateau. As in predator-prey, we read this as a noisy, single-seed landscape rather than a confirmed optimum, and seed reruns of the top candidates would be needed to settle it. Even so, every point up to 18.4% of the budget beats or matches DiCo. Sampling thus needs only a short shared phase, like navigation and in contrast to predator-prey, where sharing throughout is best.

E Extended Discussion

This section expands on the summary in Section 5.3.

Navigation search and learning dynamics. Table 4 lists the eight switch points the navigation search evaluated, in the same format as Table 2.

Full sharing performs worst by a wide margin (0.0055): with distinct per-agent goals, a single shared policy cannot specialize, and its return stays essentially flat from early in training. NoPS eventually does far better (0.0300) but slowly: starting both agents from independent random initializations, its return climbs unevenly and is still rising at 8M frames. DiCo reaches 0.0157 and is the least stable of the three, climbing to about 0.0135 near 5M frames, dipping to about 0.0085 by 6M, and only

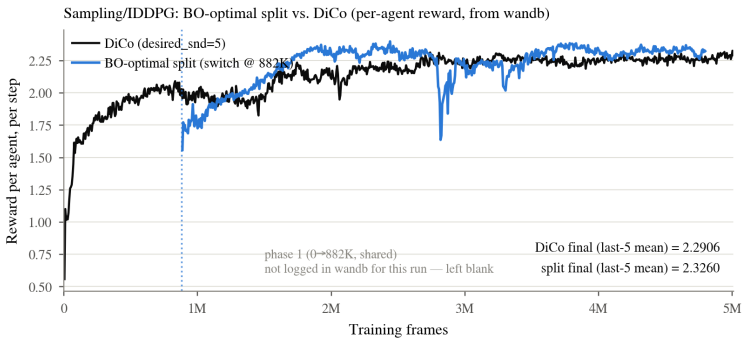


Figure 7: Sampling (IDDPG): per-agent reward vs. training frames for DiCo (`desired_snd=5`, solid black) and the best switch point from Table 3 (solid blue, switch at 882,000 frames), drawn from Weights & Biases logs. The shared phase before the switch was not logged for this run and is left blank. Values are averages over the last five training iterations; both runs are single-seed.

Table 4: Navigation (IPPO) switch-point Bayesian optimization results (seed 0, 12M-frame total budget).

Switch point (frames)	% of budget shared	Final return	BO order
60,000	0.5%	0.0304	1
4,260,000	35.5%	0.0084	2
8,400,000	70.0%	0.0069	3
480,000	4.0%	0.0301	4
2,100,000	17.5%	0.0124	5
6,300,000	52.5%	0.0075	6
1,200,000	10.0%	0.0217	7
180,000	1.5%	0.0303	8

partially recovering by 8M. The search is decisive: every point that shared for 4% of the budget or less reached 0.030–0.0305, while every point that shared for 10% or more dropped to 0.007–0.022.

The best point shares for just 0.5% of the budget (60,000 frames) and reaches 0.0303, only marginally above NoPS but 93% above DiCo and about $5.5\times$ FuPS. More striking than the final value is how quickly it is reached: after the brief shared phase, the independent policies rise almost immediately and plateau by about 3M frames, well ahead of NoPS, which is still climbing at 8M, and of DiCo, which never reaches a comparable level (Figure 4). A shared phase of well under 1% of the budget is enough to give both agents a common starting competence, basic movement toward a goal, before they specialize. This head start is worth several million frames of faster convergence even though it barely changes the final score, and the search finds it without being told to look for fast convergence.

Task structure and switch points. The fraction of the budget that each task's best switch point spends sharing spans nearly the whole range, from 0.5% (navigation) to 61% (predator-prey), and the ordering follows how symmetric each task's coordination problem is. Navigation assigns each agent its own goal from the outset, so a shared policy has little to learn before the agents must specialize, and the best switch point is essentially "split immediately." Sampling and dispersion lie in between. In sampling, agents share a common search space with no individually assigned targets, so a short shared phase (about 18% of the budget) lets them learn generic search behavior before they differentiate which regions each covers. In dispersion, agents must additionally break symmetry over *which* agent claims which particle, which benefits from a somewhat longer shared phase (about 21%). Predator-prey lies at the opposite extreme from navigation: the two predators solve a tightly coupled, symmetric pursuit problem, and splitting them discards the coordination each gains from training alongside an identical partner. The search's switch point is accordingly the latest of the four (61% of the budget), and over 3 seeds never splitting is best (8% above that switch point, whereas in dispersion the best switch point beats never sharing by 54%). In short, *the less individually differentiated a*

task's optimal behavior is, the longer the homogeneous phase should run, a pattern that holds across all four tasks despite their different algorithms (IPPO, MADDPG, IDDPG) and reward structures.

Search efficiency. A fixed budget of eight training runs per task (three random, five EI-guided) was enough to find a switch point that beats DiCo on every task, even though the best switch points differ by two orders of magnitude in frames (60K to 7.26M) and the search needed no task-specific tuning beyond the frame budget $S_{\max}$. A fixed, hand-picked sweep would struggle here: because navigation's and predator-prey's best points lie at opposite ends of $[0, S_{\max}]$, any sweep coarse enough to be affordable across tasks (e.g., three or four evenly spaced candidates) risks missing the best region on some task, whereas the GP surrogate directs its five adaptive queries toward whichever region the random initial design suggests is promising.

Caveats. First, every point in the switch-point searches uses a single training seed, so individual (s, y) observations are noisy. We re-evaluated the best switch point and DiCo over 3 seeds for dispersion, navigation, and predator-prey. For navigation and dispersion, the reruns confirmed the search's conclusions: the best region of navigation's landscape is separated from the rest by a margin much larger than the run-to-run noise, and in dispersion BOSS's margin over DiCo grew. For predator-prey, whose landscape is a noisy plateau, the rerun lowered the searched switch point from 0.6785 to 0.5921, below full sharing. Sampling's landscape is also noisy and non-unimodal (Table 3), and we have not yet rerun its top candidates. Second, the naive controls are incomplete: sampling has neither a FuPS nor a NoPS run, and predator-prey has no NoPS run. Third, in predator-prey the DiCo baseline freezes the prey after 3M frames, whereas our runs train against a learning prey, so the two are not trained under identical conditions.