WHEN SHOULD AGENTS DIFFER? LEARNING DIVERSITY SCHEDULES FROM TRAINING DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Parameter sharing improves sample efficiency in cooperative multi-agent reinforcement learning, but can limit behavioral diversity and hinder agent specialization. Existing methods encourage diversity through auxiliary objectives or intrinsic rewards, typically weighted by fixed coefficients. However, the balance between diversity and coordination evolves during training: encouraging distinct behaviors can support exploration and specialization, whereas excessive diversity pressure may interfere with coordinated policies. Learning plateaus may further require renewed exploration, motivating a state-dependent approach to diversity control. We propose TIDE, a training-dynamics-informed framework for adaptive diversity scheduling. TIDE learns a lightweight controller from comparisons of alternative diversity coefficients at shared training checkpoints, then uses training statistics to adjust diversity strength throughout learning. The framework integrates with different diversity mechanisms without changing the underlying agent architectures. Experiments with three representative algorithms on six SMAC maps show improvements in mean final win rate across all 18 settings, averaging 9.9 percentage points over default coefficients. Additional experiments demonstrate cross-map transfer and reveal nonmonotonic schedules that attenuate and reactivate diversity during training. These results support scheduling diversity-related training signals as a useful approach to cooperative multi-agent reinforcement learning.

1 INTRODUCTION

Parameter sharing is widely adopted in cooperative multi-agent reinforcement learning (MARL), allowing agents to share a common policy network while acting on their local observations and histories (Gupta & Egorov, 2017; Rashid et al., 2020; Yu et al., 2022; Wen et al., 2022). Although parameter sharing supports sample efficiency and scalability, it can also limit behavioral diversity and hinder the specialization required for effective coordination. To address this challenge, prior work has explored intrinsic rewards (Li et al., 2021; Jiang & Lu, 2021), role-conditioned policies and role decomposition (Wang et al., 2020; 2021), contrastive credit assignment (Liu et al., 2023), and explicit policy diversity control (Bettini et al., 2024). These approaches provide different mechanisms for encouraging behavioral differentiation and organizing agent specialization. An important question is how strongly differentiation should be encouraged as training progresses.

The value of behavioral differentiation depends on its contribution to team coordination, and this relationship can change throughout training. Diversity incentives can broaden exploration and help agents discover complementary behaviors. As coordinated policies emerge, however, continued pressure toward differentiation may disrupt established cooperation. Conversely, a plateau in performance may call for renewed exploration, making a uniformly decreasing schedule insufficient to capture the range of useful adjustments. The appropriate strength of diversity therefore depends on the evolving training process. Figure 1 illustrates the potential of scheduling: on 3s5z_vs_3s6z, CIA with TIDE achieves a final win rate of 0.362, compared with 0.283 for the best entry of the fixed-strength sweep and 0.295 for an independently tuned fixed strength. This motivates our central question: when and to what extent should diversity be encouraged during cooperative multi-agent training?

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.

We address this question by adapting to actual training progress. Predefined schedules encode assumptions about how the need for diversity changes over time, yet different runs can develop useful behaviors or encounter performance plateaus at different stages. Elapsed training time alone may therefore provide an incomplete basis for scheduling. Training dynamics offer additional information about optimization and progress, creating an opportunity to make diversity incentives responsive to the ongoing training process. The challenge is to turn this information into effective adjustments of diversity strength as cooperative behaviors develop. We propose TIDE, a training-dynamics-informed framework for adaptive diversity scheduling. TIDE uses training feedback to regulate the strength of existing diversity mechanisms throughout training. Its lightweight controller is trained using comparisons of subsequent training outcomes under alternative diversity strengths, providing supervision for adaptive scheduling. During a new run, the controller responds to training statistics and adjusts the influence of the diversity mechanism accordingly. TIDE integrates with mechanisms based on auxiliary objectives or intrinsic rewards without modifying the underlying agent architectures, enabling a common scheduling approach across different base algorithms.

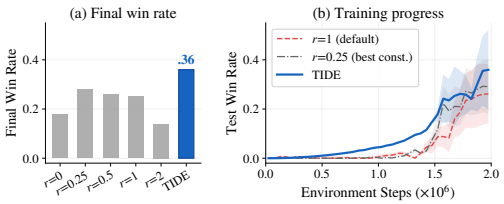


Figure 1: Fixed diversity strengths (constant sweep) versus TIDE on 3s5z_vs_3s6z with CIA. The ratio r denotes strength relative to the default setting. Results use five seeds.

We evaluate TIDE with CIA, CDS, and LIIR on six SMAC maps. Across all 18 algorithm and map combinations, TIDE improves mean final win rate over the default configurations, with an average gain of 9.9 percentage points. Comparisons with tuned fixed strengths, predefined and optimized time schedules, and alternative feedback controllers examine the sources of these improvements. Further analyses reveal nonmonotonic schedules that attenuate and reintroduce diversity incentives, and demonstrate cross-map transfer of the controller. Together, these experiments examine the effectiveness of diversity scheduling and the contribution of training feedback to its decisions.

Our contributions are threefold:

- We systematically investigate adaptive diversity scheduling in cooperative MARL, examining how the strength of diversity incentives can vary throughout training and how training dynamics can inform this adaptation.
- We propose TIDE, a lightweight feedback-driven framework that uses supervision from training outcomes to adapt diversity strength. TIDE integrates with different diversity mechanisms without modifying the underlying agent architectures.
- We conduct a systematic evaluation across three algorithms and six SMAC maps, obtaining improvements in mean final win rate across all 18 settings. Further experiments analyze the role of training feedback, nonmonotonic scheduling behavior, and cross-map transfer.

2 Related Work

Behavioral diversity and agent specialization. A range of methods promote differentiated behaviors in cooperative MARL through rewards, representations, and policy constraints. CDS encourages identity-dependent trajectories through intrinsic rewards and combines shared and agent-specific representations (Li et al., 2021). EOI promotes distinguishable agent behaviors by rewarding observations that reveal agent identity (Jiang & Lu, 2021), while LIIR learns individual intrinsic rewards to improve the team objective (Du et al., 2019). Credit-assignment methods provide another route to differentiated learning. COMA estimates individual contributions through counterfactual advantages (Foerster et al., 2018), LICA learns implicit credit assignment through a joint critic (Zhou et al., 2020), and CIA uses a contrastive objective to distinguish temporal credit representations (Liu et al., 2023). Other approaches explicitly control policy diversity (Bettini et al., 2024) or encourage trajectory differences through contrastive representations (Li & Zhu, 2025). These methods establish mechanisms for shaping agent behavior. TIDE complements this line of work by adapting the strength of existing diversity mechanisms throughout training.

Structured coordination and parameter sharing. Role-based methods organize specialization through structured policy representations. ROMA learns role-conditioned policies (Wang et al., 2020), while RODE constructs roles through action-space decomposition (Wang et al., 2021). R3DM incorporates learned dynamics to connect roles with future behavior (Goel et al., 2025). Related approaches structure cooperation through pairwise value factors (Böhmer et al., 2020) or learned subtask assignments (Yang et al., 2022). Parameter-sharing methods address specialization by determining which agents share a learner: selective parameter sharing groups agents according to their learning needs (Christianos et al., 2021), and MADPS adapts sharing using policy-distance measurements (Hu et al., 2024). These approaches modify the organization of policies or coordination. TIDE addresses the temporal use of diversity incentives within a given agent architecture.

Automated training and hyperparameter adaptation. Adapting training decisions is a central topic in automated reinforcement learning (Parker-Holder et al., 2022). Population-based training jointly updates model parameters and hyperparameters (Jaderberg et al., 2017), while population-based bandits guide hyperparameter selection with a probabilistic model (Parker-Holder et al., 2020b). Population-based reinforcement learning also uses behavioral diversity to maintain complementary exploration strategies among learners (Parker-Holder et al., 2020a). This population-level perspective concerns diversity across learners; TIDE focuses on the strength of mechanisms that encourage differentiation among cooperating agents within a team.

Adaptive learning signals and training control. Several approaches adapt learning signals during reinforcement learning. Soft actor-critic supports automatic adjustment of the entropy weight (Haarnoja et al., 2018), Agent57 uses a bandit controller to adapt exploration choices (Badia et al., 2020), and meta-gradient methods optimize hyperparameters by differentiating through learning updates (Xu et al., 2018). AutoMBPO uses optimization-state features to guide hyperparameter scheduling (Lai et al., 2021). These studies motivate the use of training feedback to guide adaptation. TIDE develops this direction for diversity scheduling in cooperative MARL, using comparisons of training outcomes under alternative diversity strengths to supervise a feedback controller.

Learning to rank. Ranking methods provide objectives for comparing alternatives through their relative outcomes. ListNet introduces a listwise objective based on distributions over candidate scores (Cao et al., 2007), with subsequent work developing theoretical analyses and algorithms for listwise ranking (Xia et al., 2008). TIDE uses a listwise objective to translate comparisons of diversity strengths into supervision for its scheduling controller.

3 Diversity Scheduling as Adaptive Training Control

We consider cooperative Dec-POMDPs (Oliehoek & Amato, 2016) under centralized training with decentralized execution. Diversity scheduling operates at the training level: a scheduler observes training dynamics and regulates diversity strength to improve the final team policy.

Different mechanisms influence training through different pathways:

$$\begin{aligned} \mathcal{L}_{\mathrm{CIA}} &= \mathcal{L}_{\mathrm{TD}} + \alpha \mathcal{L}_{\mathrm{CL}}, \\ y_{\mathrm{CDS}} &= R + \beta r^{\mathrm{int}} + \gamma Q', \\ R_{i,\mathrm{LIIR}} &= R + \lambda r_i^{\mathrm{int}}. \end{aligned} \tag{1}$$

TIDE accesses these mechanisms through a common interface: the strength of the diversity-related training signal. This coefficient regulates how strongly the mechanism shapes policy learning through its auxiliary objective or intrinsic reward. For a base strength θ_0, the scheduling action is

$$\theta_k = r_k \theta_0, \qquad r_k \in \mathcal{R} = \{0, 0.25, 0.5, 1, 2\}. \tag{2}$$

This relative representation provides a unified action space for suppressing, attenuating, retaining, or amplifying the controlled signal across different mechanisms.

Let x_k denote the underlying training state and $s_k = \phi(x_k)$ the statistics available to the scheduler. The action $r_k = \pi(s_k)$ governs the next Δ training steps, inducing $x_{k+1} \sim P_\Delta(\cdot \mid x_k, r_k)$. For a budget $T_{\max}$, the objective is

$$J(\pi) = \mathbb{E}[V(x_K)], \qquad \pi^\star \in \arg\max_\pi J(\pi), \tag{3}$$

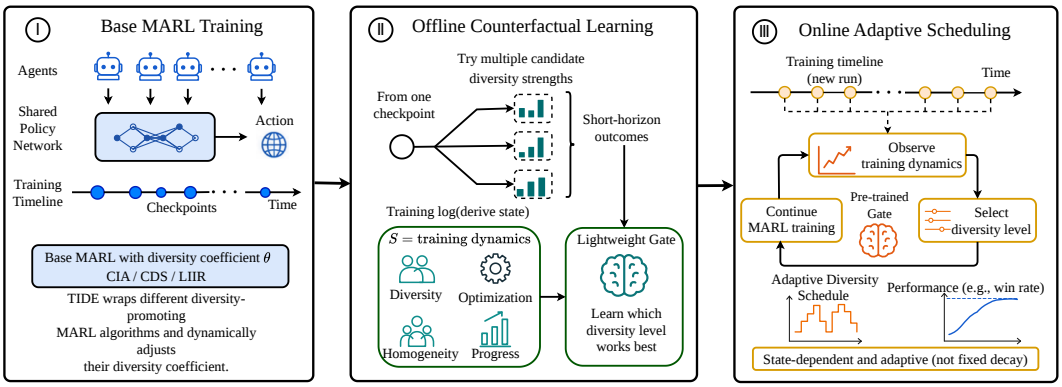


Figure 2: TIDE constructs supervision through offline counterfactual evaluation and adapts diversity strength using training feedback during new runs.

where $V(x_K)$ measures final task performance. TIDE conditions its decisions on training feedback, allowing diversity scheduling to respond to actual training progress.

4 TIDE

TIDE uses offline counterfactual evaluation to supervise a lightweight controller that adapts diversity strength from training dynamics. Figure 2 summarizes its offline and online stages.

To construct supervision, TIDE evaluates each candidate strength over a finite training horizon W. This horizon allows changes in exploration, experience collection, and policy updates to contribute to the comparison while keeping branch evaluation tractable:

$$g_W(x, r) = \mathbb{E}[G(\mathrm{Train}(x, r, W))]\,, \tag{4}$$

where $\mathrm{Train}(x, r, W)$ continues training from x under strength r for W steps, and G measures the resulting outcome. The controller learns candidate preferences from these outcomes. The horizon W sets the depth of offline evaluation, while Δ sets the frequency of online adaptation. This design combines extended evaluation of each training signal with frequent feedback during deployment. Section 5.3 examines ranking consistency across horizons and the relationship between branch preferences and final task performance.

4.1 Representing Training Dynamics

The controller observes four complementary groups of statistics: mechanism activation, describing the auxiliary loss or intrinsic reward; optimization dynamics, capturing gradient and parameter scales; value statistics, summarizing agent value estimates and logged contributions; and training progress, tracking prediction errors and performance trends. Together, they describe the signals acting on the agents and the progress of training.

The representation contains 14, 15, and 16 dimensions for CIA, CDS, and LIIR, respectively. It excludes explicit task identifiers, agent counts, and a normalized training clock. Feature processing is fitted offline and frozen for deployment. Complete definitions are provided in Appendix B.

4.2 Offline Counterfactual Supervision

An ordinary training trajectory reveals the outcome of the diversity strength actually applied. TIDE evaluates alternative choices offline by continuing from the same training state under each $r \in \mathcal{R}$ for W steps. Each comparison associates a common training context with the subsequent outcomes of candidate strengths, providing comparative supervision for the controller.

To distinguish outcomes when successful episodes are scarce, we compare kill ratio and normalized return using ranking stability on separate offline selection data. For episode i, the kill ratio is

$\kappa_i = (\text{enemies killed})/\max(\text{allies lost}, 1)$, and the branch outcome $\widehat{g}_{c,r}$ is the mean of κ_i over the episodes of the final 20K continuation steps. Averaging per-episode ratios is not equivalent to dividing total kills by total losses over the window; we use the former throughout. Kill ratio is selected for the reported SMAC maps. Collection and evaluation protocols are detailed in Appendix C.

The controller assigns each candidate a score $f_\psi(s, r)$ conditioned on training dynamics. Comparative outcomes are converted into a target distribution p_c over candidate strengths, while the controller scores induce a predicted distribution $q_{\psi,c}$:

$$p_{c,r} = \frac{\exp(\widehat{g}_{c,r}/T)}{\sum_{r' \in \mathcal{R}} \exp(\widehat{g}_{c,r'}/T)}, \qquad q_{\psi,c,r} = \frac{\exp(f_\psi(s_c, r))}{\sum_{r' \in \mathcal{R}} \exp(f_\psi(s_c, r'))}, \tag{5}$$

where T is the target temperature. Targets are built from the raw branch outcomes: T rescales the outcome differences without changing their order, and no per-group normalization is applied before the softmax. A listwise cross-entropy objective aligns these distributions:

$$\mathcal{L}_{\text{rank}}(\psi) = -\frac{1}{|\mathcal{D}|} \sum_{c \in \mathcal{D}} \sum_{r \in \mathcal{R}} p_{c,r} \log q_{\psi,c,r}, \tag{6}$$

where $\mathcal{D}$ contains the offline comparison groups. Model selection uses held-out decision regret, measured by the outcome gap between the selected and best candidates. Target construction, controller architecture, and optimization settings are provided in Appendix B.

Algorithm 1 TIDE: Adaptive Diversity Scheduling

Require: Base algorithm $\mathcal{A}$, base strength θ_0, candidate set $\mathcal{R}$, evaluation horizon W, control interval Δ, warm-up duration T_c, training budget $T_{\max}$
Ensure: Scheduling controller f_ψ and trained agent policies
Offline: counterfactual supervision
Run $\mathcal{A}$ with strength θ_0 and collect reference training states $\{x_c\}$
Initialize comparison dataset $\mathcal{D} \leftarrow \varnothing$
for all reference states x_c **do**
  Extract training dynamics $s_c \leftarrow \phi(x_c)$
  for all $r \in \mathcal{R}$ **do**
    Independently continue from x_c with strength $r\theta_0$ for W steps
    Record the resulting outcome $\widehat{g}_{c,r}$
  end for
  Add $(s_c, \{\widehat{g}_{c,r}\}_{r \in \mathcal{R}})$ to $\mathcal{D}$
end for
Construct supervision and train f_ψ using Equation (6); select the model by held-out regret
Online: feedback-driven scheduling
Initialize new agents and train with strength θ_0 until T_c
for $t = T_c, T_c + \Delta, \ldots$ while $t < T_{\max}$ **do**
  Extract current training dynamics s_t
  Select $r \leftarrow \arg\max_{u \in \mathcal{R}} f_\psi(s_t, u)$
  Train with strength $r\theta_0$ for $\min(\Delta, T_{\max} - t)$ steps
end for
return f_ψ and the trained agent policies

4.3 Feedback-Driven Diversity Scheduling

The base diversity strength is retained during an initial warm-up period T_c. Afterwards, the controller observes training dynamics every Δ steps and selects

$$r_k = \arg\max_{r \in \mathcal{R}} f_\psi(s_k, r). \tag{7}$$

The selected strength governs the next training interval, forming a feedback loop between training progress and diversity scheduling. Controller parameters remain fixed throughout the run.

We use $W = 200$K steps and $\Delta = 20$K steps, with T_c matching the base exploration-annealing period. The controller is required only during training and leaves decentralized execution unchanged. Algorithm 1 gives the complete procedure; Appendix E.3 reports offline collection and runtime costs.

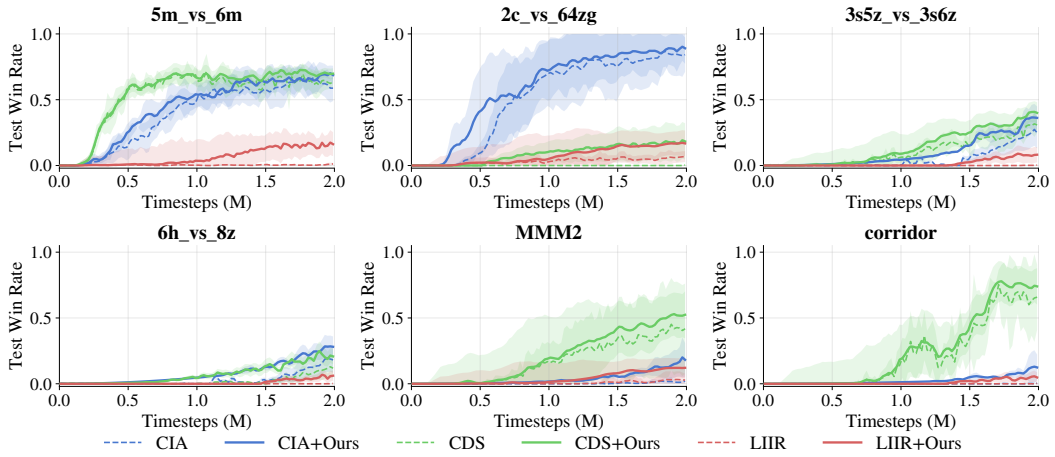


Figure 3: Learning curves of CIA, CDS, and LIIR with and without TIDE on six SMAC maps. Lines and shading indicate means and standard deviations across five seeds.

5 Experiments

5.1 Setup and Evidence Across Diversity Mechanisms

We evaluate CIA, CDS, and LIIR on six hard or super-hard SMAC maps (Samvelyan et al., 2019), using StarCraft II 4.10, difficulty 7, and a 2M-step training budget. Results are averaged over five training seeds. Evaluation uses 32 greedy episodes every 10K steps, and final win rate averages the last ten evaluations. Shading and $\pm$ values indicate sample standard deviation across seeds. Base comparisons use matched seeds and identical configurations. All main controllers are evaluated on independent training seeds excluded from offline collection, testing their generalization to unseen training trajectories. Unless otherwise stated, controllers are trained for their respective algorithm and map settings. Appendix A provides the complete protocol.

Figure 3 shows that TIDE improves mean final win rate across all 18 algorithm and map combinations. Averaged over the six maps, the gains are 10.7, 10.2, and 8.8 percentage points for CIA, CDS, and LIIR, respectively. The overall mean win rate increases from approximately 0.230 to 0.329. These improvements across distinct base mechanisms support adaptive diversity scheduling under the tested budgets and configurations. For example, CIA on MMM2 improves from 0.021 ± 0.018 to 0.182 ± 0.129, while CDS on corridor improves from 0.658 ± 0.248 to 0.740 ± 0.138. Appendix Table 5 reports all final values, with available seed-level results in Table 6.

The largest mean gains occur on different maps for the three algorithms: MMM2 for CIA, 2c_vs_64zg for CDS, and 5m_vs_6m for LIIR. This variation highlights the interaction between the base learning mechanism and the coordination demands of each task. A common scheduling interface accommodates these differences by choosing coefficient strengths from the training statistics produced by each learner.

Additional experiments on six SMACv2 configurations (Ellis et al., 2023) extend the evaluation to procedurally generated scenarios. With a separately trained controller for each configuration, CIA+TIDE improves mean win rate by approximately 7.5 percentage points on average. Appendix H also reports a comparison with R3DM (Goel et al., 2025) on two SMAC maps.

5.2 Does Training Feedback Improve the Schedule?

We compare scheduling strategies on three CIA maps with distinct temporal patterns. The baselines include fixed-strength tuning, linear and step decay, random scheduling, online UCB, a time-only controller, frozen schedule replay, and budget-matched schedule search. The time-only controller uses the same supervision and training objective as TIDE, with training time as its input. Frozen replay applies schedules extracted from separate TIDE development runs. Schedule search optimizes

Table 1: Comparison of diversity scheduling strategies with CIA. Entries report mean final win rate ± standard deviation across five seeds.

Controller	3s5z_vs_3s6z	2c_vs_64zg	6h_vs_8z
Default diversity strength	.248 ± .118	.842 ± .131	.181 ± .098
Linear decay	.269 ± .121	.853 ± .138	.198 ± .102
Step decay	.288 ± .109	.839 ± .119	.218 ± .091
Random scheduling	.237 ± .158	.789 ± .208	.172 ± .128
Online UCB	.315 ± .098	.838 ± .149	.236 ± .089
TIDE	.362 ± .087	.891 ± .118	.281 ± .079
Independently tuned fixed strength	.295 ± .106	.876 ± .104	.239 ± .086
Time-only controller	.337 ± .094	.884 ± .111	.263 ± .082
Frozen schedule replay	.329 ± .101	.882 ± .109	.251 ± .087
Budget-matched schedule search	.349 ± .091	.896 ± .103	.274 ± .081

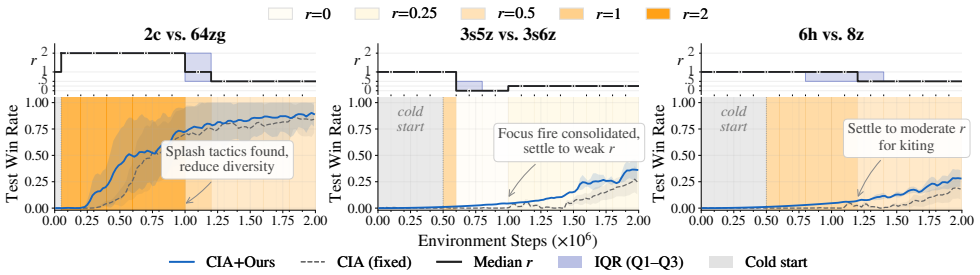


Figure 4: CIA scheduling and performance. Top: relative diversity strength (median, interquartile range). Bottom: win rate (mean, standard deviation) across five seeds.

a three-segment temporal schedule that permits both increases and decreases in diversity strength. Development protocols, selection rules, and budgets are provided in Appendix E.

TIDE achieves higher mean final win rates than UCB, the independently tuned fixed strength, the time-only controller, and frozen replay on all three maps. Relative to the independently tuned fixed strength, the gains are 6.7, 1.5, and 4.2 percentage points on 3s5z_vs_3s6z, 2c_vs_64zg, and 6h_vs_8z, respectively. The corresponding gains over the time-only controller are 2.5, 0.7, and 1.8 points, showing a consistent mean improvement when scheduling uses training feedback. TIDE leads schedule search by 1.3 and 0.7 points on 3s5z_vs_3s6z and 6h_vs_8z, while search leads by 0.5 points on 2c_vs_64zg. Together, these comparisons support adjusting diversity during training and using observed progress to refine the resulting schedule.

The larger gains over fixed coefficients on 3s5z_vs_3s6z and 6h_vs_8z highlight the value of changing the training signal as cooperation develops. On 2c_vs_64zg, the tuned fixed coefficient already reaches a mean win rate of 0.876, and both TIDE and schedule search improve on this level. The time-only and replay controls also outperform the default setting on all three maps, showing that the temporal allocation of diversity pressure contributes to the gains. TIDE's higher means relative to these two controls further support adapting that allocation to the progress of each training run.

5.3 How Faithful Is the Branch Supervision?

Increasing the evaluation horizon from 50K to 200K improves CIA's final win rate on 3s5z_vs_3s6z from 0.291 ± 0.104 to 0.362 ± 0.087. A nominal 400K horizon, with branches truncated at the 2M-step training limit, yields 0.366 ± 0.085. The reported mean changes little between the primary 200K configuration and this longer-horizon variant. Appendix Table 14 reports the complete horizon comparison.

We further evaluate supervision across horizons W, $2W$, and $4W$ using a common set of reference training states and repeated continuations. Kendall's τ_b and top-choice agreement measure ranking consistency. A terminal evaluation applies each candidate for one control interval Δ, then follows

Table 2: Counterfactual supervision on three CIA maps. Brackets indicate 95% confidence intervals clustered by source training run. Lower terminal regret is better.

Map	$\tau_b(W, 2W)$	$\tau_b(W, 4W)$	Top agreement $(W, 4W)$	Terminal regret
3s5z_vs_3s6z	.68 [.55, .79]	.47 [.31, .61]	.60 [.40, .80]	.031 [.012, .056]
2c_vs_64zg	.76 [.64, .85]	.63 [.48, .76]	.73 [.53, .87]	.015 [.004, .031]
6h_vs_8z	.62 [.46, .75]	.39 [.21, .55]	.53 [.33, .73]	.046 [.021, .077]

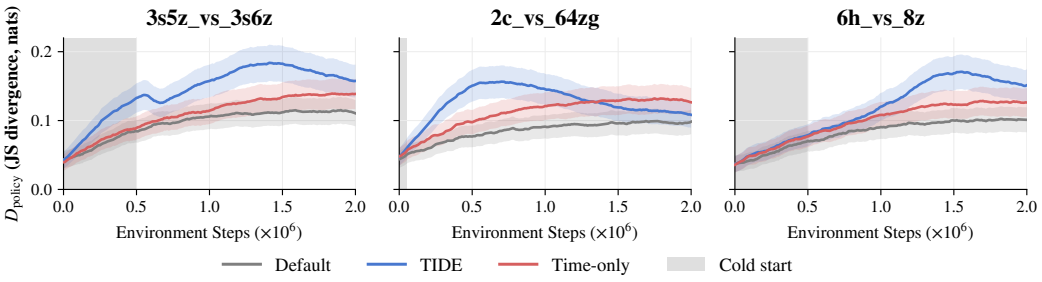


Figure 5: Policy diversity measured by Jensen-Shannon divergence on three CIA maps. Shading shows standard deviation across five seeds.

the same frozen TIDE controller until the training budget is exhausted. This measures the final-performance regret of the candidate preferred by the W-horizon outcome. The evaluation protocol is detailed in Appendix C.

The W-horizon rankings remain positively correlated with both longer horizons. Kendall's τ_b ranges from 0.62 to 0.76 for W versus $2W$, and from 0.39 to 0.63 for W versus $4W$. Top-choice agreement with $4W$ ranges from 0.53 to 0.73. Under the terminal evaluation, the selected candidates incur win-rate regret between 0.015 and 0.046. These measurements characterize the persistence of candidate rankings and their relationship to subsequent task performance.

5.4 Schedules and Behavioral Diversity

Figures 4 and 5 connect the temporal use of diversity incentives with the evolution of policy differences. Figure 4 reports diversity strength as the median and interquartile range, and win rate as the mean and standard deviation across five seeds. Figure 5 measures Jensen-Shannon divergence between compatible agents' action distributions on a fixed shared probe set. Action semantics and availability are matched across comparisons. The probe construction and metric definitions are provided in Appendix G.

On 3s5z_vs_3s6z, TIDE temporarily suppresses diversity incentives and subsequently restores them at a reduced strength. Policy divergence increases over much of training before easing toward the end. The schedule separates a period of reduced differentiation pressure from a later period of renewed incentives. This temporal pattern addresses the map's combination of complementary agent behaviors and coordinated execution.

On 2c_vs_64zg, where effective positioning is central to coordination, TIDE amplifies diversity after warm-up and subsequently reduces it. Policy divergence rises initially and then declines, eventually falling below the time-only controller. Stronger incentives are concentrated in the earlier controlled phase, while lower incentives accompany the later refinement of coordinated movement and positioning.

On 6h_vs_8z, which emphasizes coordinated focus fire, TIDE retains the default diversity strength for longer before attenuation. Policy divergence increases through the middle and later stages of training and then tapers toward the end. Compared with the other two maps, this schedule maintains differentiation pressure over a longer portion of training before reducing it during the final refinement of team behavior.

Table 3: Cross-map transfer with CIA. Source-trained controllers exclude target-map data. Results report mean ± standard deviation across five seeds.

Target map	Default	Target-trained TIDE	Source-trained TIDE
5m_vs_6m	$.588 \pm .098$	$.687 \pm .071$	$.648 \pm .092$
3s5z_vs_3s6z	$.248 \pm .118$	$.362 \pm .087$	$.316 \pm .098$
6h_vs_8z	$.181 \pm .098$	$.281 \pm .079$	$.238 \pm .091$

Across the three maps, useful schedules differ in both the timing and direction of coefficient changes. Suppression followed by reintroduction on 3s5z_vs_3s6z is particularly relevant to the motivation for adaptive control: diversity incentives can remain useful after an earlier period of attenuation. The accompanying policy trajectories connect coefficient scheduling with measured behavioral differences and support treating diversity strength as a training decision that is revisited throughout learning.

5.5 Transfer, Ablations, and Resource Requirements

Beyond generalization across independent training runs, we evaluate cross-map transfer within CIA. Each transferred controller is trained using source-map data and deployed on a target map excluded from controller training. The transferred controller improves mean final win rate by 6.0, 6.8, and 5.7 percentage points on 5m_vs_6m, 3s5z_vs_3s6z, and 6h_vs_8z, respectively. These results demonstrate controller reuse across maps without target-map supervision.

Removing training-progress features produces the largest increase in validation regret, from 0.0147 to 0.0392, followed by removing value statistics, which increases regret to 0.0275. The full representation also outperforms the progress-only variant in both regret and best-candidate accuracy, supporting the use of complementary training signals. Additional loss, encoding, and surrogate ablations are reported in Appendix F.

Performance varies little across control intervals from 5K to 50K steps and decreases at 100K on the three CIA maps. The primary configuration uses a 20K interval and a 200K evaluation horizon. Core offline collection requires 35M environment steps per algorithm and map setting, approximately 175 GPU-hours on the reported hardware. The schedule-search baseline uses 34M development steps under a 35M cap, with core collection interactions as the budget reference. Detailed resource accounting is provided in Appendix E.3.

6 Discussion and Conclusion

We presented TIDE, a training-dynamics-informed framework that adaptively regulates the strength of diversity incentives in cooperative MARL. Through offline counterfactual supervision, TIDE trains a lightweight controller that responds to training feedback and integrates with existing diversity mechanisms. Across three algorithms and six SMAC maps, TIDE improves mean final win rate over the default configurations in all 18 settings. The experiments further demonstrate generalization to unseen training trajectories and cross-map reuse of the controller. The resulting schedules exhibit distinct temporal patterns, including gradual attenuation and suppression followed by reintroduction, highlighting the temporal organization of diversity as a useful dimension of cooperative learning.

These results support adapting diversity-related training signals over the course of learning. TIDE provides a unified feedback interface for existing mechanisms and learns schedules that vary with the progress of individual training runs. Future work will explore more efficient supervision collection, scheduling objectives that capture longer-term effects, and controller reuse across a broader range of cooperative tasks.

Reproducibility Statement

The appendix specifies the control interface, checkpoint placement, gate architecture, feature families, evaluation protocol, scheduling controls, reliability diagnostics, and behavioral measurements. It includes all reported main-table values and available CIA seed-level results. Implementation-specific logging and configuration details are tied to the corresponding experiment configurations and run metadata.

Statement on the Use of Generative AI

Generative AI was used to assist with manuscript drafting, language revision, literature checking, and the design of additional evaluations. It was not used to run the reported experiments or to generate the reported numerical results. The authors remain responsible for the manuscript, experimental record, and final claims.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In Advances in Neural Information Processing Systems, 2021.

Adrià Puigdomènech Badia et al. Agent57: Outperforming the atari human benchmark. In Proceedings of the 37th International Conference on Machine Learning (ICML), pp. 507–517, 2020.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In Advances in Neural Information Processing Systems, volume 24. Curran Associates, Inc., 2011.

Matteo Bettini, Ryan Kortvelesy, and Amanda Prorok. Controlling behavioral diversity in multi-agent reinforcement learning. In Proceedings of the International Conference on Machine Learning, 2024.

Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In Proceedings of the International Conference on Machine Learning, pp. 980–991, 2020.

Zhe Cao et al. Learning to rank: From pairwise approach to listwise approach. In Proceedings of the 24th International Conference on Machine Learning (ICML), pp. 129–136, 2007.

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, pp. 1989–1998, 2021.

Yali Du, Lei Han, Meng Fang, Ji Liu, Tianhong Dai, and Dacheng Tao. LIIR: Learning individual intrinsic reward in multi-agent reinforcement learning. In Advances in Neural Information Processing Systems, volume 32, 2019.

Benjamin Ellis, Jonathan Cook, Skander Moalla, and Mikayel Samvelyan. SMACv2: An improved benchmark for cooperative multi-agent reinforcement learning. In Advances in Neural Information Processing Systems, 2023.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In Proceedings of the AAAI Conference on Artificial Intelligence, 2018.

Harsh Goel, Mohammad Omama, Behdad Chalaki, Vaishnav Tadiparthi, Ehsan Moradi Pari, and Sandeep Chinchali. R3dm: Enabling role discovery and diversity through dynamics models in multi-agent reinforcement learning. In Proceedings of the International Conference on Machine Learning, 2025.

Jayesh K. Gupta and Maxim Egorov. Cooperative multi-agent control using deep reinforcement learning. In Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, pp. 66–83, 2017.

Tuomas Haarnoja et al. Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905, 2018.

Tianyi Hu, Zhiqiang Pu, Xiaolin Ai, Tenghai Qiu, and Jianqiang Yi. Measuring policy distance for multi-agent reinforcement learning. In Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, 2024.

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, 2017.

Jiechuan Jiang and Zongqing Lu. The emergence of individuality in multi-agent reinforcement learning. In Proceedings of the International Conference on Machine Learning, 2021.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In Proceedings of the 19th International Conference on Machine Learning (ICML), pp. 267–274, 2002.

Hang Lai, Jian Shen, Weinan Zhang, Yimin Huang, Xing Zhang, Ruiming Tang, Yong Yu, and Zhenguo Li. On effective scheduling of model-based reinforcement learning. In Advances in Neural Information Processing Systems, volume 34, pp. 3694–3705. Curran Associates, Inc., 2021.

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

Tianxu Li and Kun Zhu. Encouraging metric-aware diversity in contrastive representation space. In Advances in Neural Information Processing Systems, volume 38, pp. 60857–60883. Curran Associates, Inc., 2025. doi: 10.52202/085713-2035.

S. Liu, Y. Zhou, J. Song, T. Zheng, K. Chen, T. Zhu, others, and M. Song. Contrastive identity-aware learning for multi-agent value decomposition. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 37, pp. 11595–11603, June 2023.

R. Duncan Luce. Individual Choice Behavior: A Theoretical Analysis. Courier Corporation, 2012.

Frans A. Oliehoek and Christopher Amato. A Concise Introduction to Decentralized POMDPs. Springer, 2016.

Jack Parker-Holder, Aldo Nguyen, and Stephen J. Roberts. Effective diversity in population based reinforcement learning. In Advances in Neural Information Processing Systems, volume 33, 2020a.

Jack Parker-Holder, Vu Nguyen, and Stephen J. Roberts. Provably efficient online hyperparameter optimization with population-based bandits. In Advances in Neural Information Processing Systems (NeurIPS), 2020b.

Jack Parker-Holder, Raghu Rajan, Xingyou Song, André Biedenkapp, and Yingjie Miao. Automated reinforcement learning (AutoRL): A survey and open problems. Journal of Artificial Intelligence Research, 74:517–568, 2022.

R. L. Plackett. The analysis of permutations. Journal of the Royal Statistical Society: Series C (Applied Statistics), 24(2):193–202, 1975.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. volume 21, pp. 1–51, 2020.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft multi-agent challenge. In Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems, pp. 2186–2188, 2019.

Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. ROMA: Multi-agent reinforcement learning with emergent roles. In Proceedings of the International Conference on Machine Learning, pp. 9876–9886, 2020.

Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. RODE: Learning roles to decompose multi-agent tasks. In Proceedings of the International Conference on Learning Representations, 2021.

Muning Wen, Jakub Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. Advances in Neural Information Processing Systems, 35, 2022.

Fen Xia et al. Listwise approach to learning to rank: Theory and algorithm. In Proceedings of the 25th International Conference on Machine Learning (ICML), pp. 1192–1199, 2008.

Zhongwen Xu, Hado van Hasselt, and David Silver. Meta-gradient reinforcement learning. In Advances in Neural Information Processing Systems, volume 31, 2018.

Mingyu Yang, Jian Zhao, Xunhan Hu, Wengang Zhou, Jiangcheng Zhu, and Houqiang Li. Ldsa: Learning dynamic subtask assignment in cooperative multi-agent reinforcement learning. In Advances in Neural Information Processing Systems, volume 35, 2022.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, and Alexandre Bayen. The surprising effectiveness of PPO in cooperative multi-agent games. In Advances in Neural Information Processing Systems, volume 35, 2022.

Meng Zhou, Ziyu Liu, Pengwei Sui, Yixuan Li, and Yuk Ying Chung. Learning implicit credit assignment for cooperative multi-agent reinforcement learning. In Advances in Neural Information Processing Systems, volume 33, 2020.

A Experimental Protocol and Complete Results

A.1 Tasks, Training Budgets, and Base Configurations

The six SMAC maps are 5m_vs_6m, 2c_vs_64zg, 3s5z_vs_3s6z, 6h_vs_8z, MMM2, and corridor. They cover homogeneous and heterogeneous teams with between two and ten allied agents. All reported main comparisons use StarCraft II 4.10 at difficulty 7 and a training budget of 2M environment steps. The common training budget, game version, and evaluation aggregation provide a consistent basis for the comparisons reported here.

Table 4 summarizes the base configurations. TIDE changes the specified coefficient rather than the architecture, replay size, or base learning rate. CIA and CDS use value decomposition; LIIR uses an actor-critic learner and updates its intrinsic reward model under its original training procedure.

Table 4: Base configurations used in the reported experiments. Algorithm- and map-specific optimizer choices follow the corresponding run configurations.

Setting	CIA	CDS	LIIR
Backbone	QMIX	QPLEX	Actor-critic
Agent recurrent width	64	64	64
Mixing embedding	32	64	Not applicable
Nominal optimizer	RMSProp	RMSProp	Adam
Learning rate	5×10^{-4}	5×10^{-4}	5×10^{-4}
Replay capacity	5,000 episodes	5,000 episodes	On-policy
Parallel environments	1	1	32
Batch size	32 episodes	32 episodes	32 episodes
Discount	0.99	0.99	0.99
Target update period	200 episodes	200 episodes	Not applicable
Gradient clipping threshold	10	10	10
Exploration endpoints	$1.0 \rightarrow 0.05$	$1.0 \rightarrow 0.05$	$0.5 \rightarrow 0.01$
Default controlled scalar	$\alpha_0 = 0.02$	$\beta_0 = 1$	$\lambda_0 = 1$

The cold start is determined by the base learner's exploration-annealing period. CIA uses 50K steps on 5m_vs_6m and 2c_vs_64zg, and 500K on the other four maps. CDS uses 50K on 5m_vs_6m and corridor, and 500K elsewhere. LIIR uses 50K on all six maps. The default coefficient is used throughout this prefix. For the scheduling controls, the first decision occurs at T_c, followed by decisions at $T_c + j\Delta$; this convention also covers 50K cold starts even though 50K is not an integer multiple of the 20K interval.

A.2 Evaluation and Uncertainty

For each training seed, evaluation runs 32 episodes with exploratory action selection disabled every 10K environment steps. The reported final score averages the last ten evaluations, covering the final 100K steps and 320 episodes per seed, so every entry in Table 6 is a multiple of $1/320$. Standard deviations are computed across training seeds after aggregating the ten final evaluations within each seed. This aggregation captures variation across complete learning trajectories.

The gate uses recent win rate and its slope as input features. Consequently, evaluation episodes that provide this feedback are part of the training procedure. In the scheduling controls, a control-validation stream supplies those features, while an independent reporting stream measures performance. The reporting stream is reserved for performance measurement; controller inputs and development choices use the control-validation stream and development data. Both streams are applied consistently across the compared controls.

We report means and sample standard deviations across training seeds for each setting. Across-map averages summarize the evaluated collection of tasks. For analyses that aggregate uncertainty across maps, stratified bootstrap procedures preserve the task structure and the training-run unit of analysis (Agarwal et al., 2021). In the continuation reliability study, resampling is grouped by source training run to retain the relationship among checkpoints and branches from the same trajectory.

A.3 All Main Results

Table 5 contains the values underlying Figure 3. TIDE increases mean final win rate in all 18 settings, covering auxiliary-loss control in CIA and intrinsic-reward control in CDS and LIIR. The average gains are 10.7, 10.2, and 8.8 percentage points, respectively. The shared direction of improvement across these learning mechanisms supports using a common coefficient interface for adaptive scheduling.

The results also illustrate the range of settings in which scheduling contributes. CIA improves by 16.1 percentage points on MMM2 and 12.0 points on corridor. CDS gains 18.0 points on 2c_vs_64zg, while LIIR gains 15.0 points on 5m_vs_6m. On settings with higher default win rates, TIDE further improves CIA on 2c_vs_64zg and CDS on corridor. These comparisons show that scheduling benefits settings with both low and high default win rates. Table 6 provides the matched CIA runs underlying the aggregate results.

Table 5: Complete SMAC results. Each entry is a final win rate, mean ± sample standard deviation over five training seeds.

	CIA		CDS		LIIR	
Map	Default	TIDE	Default	TIDE	Default	TIDE
5m_vs_6m	$.588 \pm .098$	$.687 \pm .071$	$.636 \pm .036$	$.698 \pm .041$	$.012 \pm .008$	$.162 \pm .091$
2c_vs_64zg	$.842 \pm .131$	$.891 \pm .118$	$.003 \pm .006$	$.183 \pm .138$	$.068 \pm .041$	$.168 \pm .098$
3s5z_vs_3s6z	$.248 \pm .118$	$.362 \pm .087$	$.308 \pm .062$	$.400 \pm .081$	$.003 \pm .006$	$.082 \pm .058$
6h_vs_8z	$.181 \pm .098$	$.281 \pm .079$	$.118 \pm .078$	$.208 \pm .061$	$.002 \pm .004$	$.061 \pm .038$
MMM2	$.021 \pm .018$	$.182 \pm .129$	$.418 \pm .343$	$.524 \pm .178$	$.028 \pm .048$	$.121 \pm .078$
corridor	$.004 \pm .004$	$.124 \pm .098$	$.658 \pm .248$	$.740 \pm .138$	$.002 \pm .004$	$.048 \pm .038$

Table 6: Per-seed CIA final win rates. Column indices identify matched runs; they are not literal random-seed IDs. Values are rounded to three decimals.

Map	Method	Run 1	Run 2	Run 3	Run 4	Run 5
5m_vs_6m	CIA	.647	.709	.584	.453	.547
	CIA+TIDE	.622	.703	.803	.659	.650
2c_vs_64zg	CIA	.959	.925	.866	.625	.834
	CIA+TIDE	.906	.972	.903	.691	.984
3s5z_vs_3s6z	CIA	.091	.291	.159	.356	.344
	CIA+TIDE	.397	.209	.372	.409	.422
6h_vs_8z	CIA	.081	.191	.141	.153	.341
	CIA+TIDE	.200	.294	.403	.225	.284
MMM2	CIA	.016	.047	.009	.003	.031
	CIA+TIDE	.359	.006	.209	.131	.203
corridor	CIA	.000	.006	.009	.000	.003
	CIA+TIDE	.253	.184	.059	.116	.006

B Features, Gate, and Complete Procedure

B.1 Feature Definitions and Preprocessing

Table 7 records the four feature families. Window-level statistics use the preceding training window, so decisions do not access future outcomes. Mechanism features follow the corresponding base learner; value and error statistics in LIIR refer to its critic. For value-decomposition learners, c_i denotes the per-agent contribution statistic recorded by the corresponding implementation. This definition keeps the feature aligned with the base learner's mixing architecture.

For feature coordinate j, preprocessing uses $u_j = (s_j - \mu_j)/\max(\sigma_j, \epsilon)$, where μ_j and σ_j are fitted on the offline training partition and frozen and ϵ is the fixed stability constant used by the implementation. Clipping bounds are the training minimum minus half an interquartile range and the training maximum plus half an interquartile range, represented in the same coordinates as u_j.

Table 7: Gate observations. There are 13 common coordinates and 1, 2, or 3 mechanism-specific coordinates.

Family	Count	Statistics
CIA mechanism	1	Mean contrastive identity-aware loss.
CDS mechanism	2	Mean intrinsic-reward magnitude; entropy of mixing-attention weights.
LIIR mechanism	3	Intrinsic-reward mean, temporal spread, and cross-agent dispersion.
Optimization	4	Global gradient norm after clipping; Frobenius norms of encoder, recurrent, and output-layer weights.
Value statistics	3	Across-agent standard deviation of chosen-action values; mean and standard deviation of the logged per-agent contribution c_i. For LIIR, use per-agent critic values.
Progress	6	Total loss; mean absolute TD or critic error; mean chosen and target values; latest available control win rate; ordinary least-squares slope of control win rate over the preceding 100K steps.

Validation and target-map data do not contribute to these quantities. The saved gate package includes the transform and clipping bounds together with the weights.

Clipping occurs on 3.1% of online queries in the reported runs. The fixed transform places the four feature families on comparable numerical scales, and coordinate-wise clipping bounds the values presented to the gate during deployment. Packaging this transform with the trained weights ensures that the same preprocessing is used throughout controller evaluation and reuse.

B.2 Gate Architecture and Fitting

The candidate encoding is a five-dimensional one-hot vector. Concatenating it with the state gives an input dimension of 19, 20, or 21. The MLP has hidden widths 40 and 12, ReLU activations, and a scalar output, yielding 1,305, 1,345, or 1,385 trainable parameters. Gate training uses Adam with learning rate 10^{-3}, weight decay 10^{-4}, batches of eight checkpoint groups, at most 500 epochs, and early stopping after 15 epochs without improvement in validation regret. The target-temperature schedule decreases linearly from 2 to 0.5 during the first 60% of the maximum training schedule and then stays at 0.5.

The reported gate ablations use leave-one-checkpoint-out evaluation across the six CIA maps, with 25 groups per map and 150 groups in total. This protocol measures candidate selection at held-out checkpoints and provides a common basis for comparing features, encodings, and losses. Deployment experiments separately evaluate the resulting controllers on new training seeds, while the transfer study evaluates reuse on target maps.

B.3 End-to-End Procedure

Algorithm 2 distinguishes offline collection from deployment. A restored checkpoint includes the recurrent learner and its training context. Candidate continuations begin from this shared state and use their recorded environment and minibatch seeds, so each outcome can be traced to its source checkpoint and continuation.

C Offline Supervision and Reliability Diagnostics

C.1 Checkpoint Placement and Surrogate Selection

Table 8 gives the saved times used by the reported 200K-branch procedure. Checkpoints are specified in advance by environment steps and span the controlled portion of training. The five checkpoints from each mother run capture changes in training dynamics over time, while five source seeds provide multiple learning trajectories for each setting.

The surrogate study compares rank changes across consecutive checkpoints and Kendall correlation across checkpoints and matched times in different runs. Table 9 reports the summary. Kill ratio yields fewer changes in the preferred candidate and higher rank correlations on all six maps.

Algorithm 2 TIDE: offline supervision, gate fitting, and online control

Inputs: base learner $\mathcal{A}$, default θ_0, candidate set $\mathcal{R}$, checkpoint times, offline seeds, W, Δ, T_c, and $T_{\max}$.
For each offline seed, train $\mathcal{A}$ at $r = 1$ and save complete states x_c and preceding-window features s_c at the prescribed checkpoints.
For each checkpoint c and candidate r, restore x_c, set $\theta = r\theta_0$, and continue training for W steps. Record the surrogate over the final 20K steps as $\widehat{g}_{c,r}$.
Assemble groups $(s_c, \{\widehat{g}_{c,r}\}_{r\in\mathcal{R}})$, retaining mother-run, map, checkpoint, and continuation-seed identifiers.
Fit the feature transform on training groups, construct targets with Equation (5), and fit the gate with Equation (6). Select its checkpoint by validation regret.
Freeze the gate, feature statistics, clipping bounds, candidate order, and base configuration.
Initialize a new learner with a deployment seed absent from offline collection. Train at $r = 1$ until T_c.
At $t = T_c + j\Delta$, summarize the preceding window, normalize and clip its features, and score every candidate. Select $r = \arg\max_u f_\psi(s_t, u)$.
Train with $\theta = r\theta_0$ until the next decision or $T_{\max}$. Continue recording independent reporting evaluations.
Return the trained decentralized policies and their evaluation record. The gate is unnecessary for executing these policies.

Table 8: Checkpoint times for the reported offline collection, in millions of environment steps.

Exploration anneal	Checkpoint 1	2	3	4	5
50K	0.10	0.50	0.90	1.30	1.70
500K	0.55	0.85	1.15	1.45	1.75

These results motivate its use for comparative supervision when successful episodes are sparse. The terminal continuation diagnostic below complements the stability analysis by connecting branch preferences with final task performance.

Table 9: Surrogate-ranking stability. Flip rate is the fraction of consecutive checkpoint comparisons in which the best candidate changes; τ is the Kendall rank correlation. Kill ratio is selected on every map.

	Flip rate		Kendall τ	
Map	Return	Kill ratio	Return	Kill ratio
5m_vs_6m	.69	.31	.21	.58
2c_vs_64zg	.75	.25	.14	.63
3s5z_vs_3s6z	.88	.36	.06	.53
6h_vs_8z	.81	.44	.11	.45
MMM2	.72	.38	.19	.52
corridor	.84	.27	.09	.64

C.2 Horizon and Continuation Reliability Diagnostic

The reliability diagnostic in Table 2 uses CIA on 3s5z_vs_3s6z, 2c_vs_64zg, and 6h_vs_8z. We restrict the shared checkpoint population to times at or before 1.2M steps so that all $4W = 800$K continuations fit within the 2M budget. For the 50K anneal schedule this includes 0.1M, 0.5M, and 0.9M; for the 500K schedule it includes 0.55M, 0.85M, and 1.15M. Later checkpoints are excluded from the common W, $2W$, and $4W$ aggregate.

We run five continuation seeds per checkpoint and candidate and restore the same pre-branch state for every candidate. Matched continuation seed identifiers provide a common randomization protocol across candidates as each branch develops its own training trajectory. All individual outcomes are retained. When a single 800K fixed-coefficient branch supplies intermediate 200K and 400K observations, its cost is counted once and the resulting dependence is preserved in the analysis.

For candidate means $\bar{g}_{c,r}^{(H)}$ at horizon H, we compute Kendall τ_b between candidate rankings at 200K and 400K, and between 200K and 800K. We use τ_b because ties can be frequent. For top-choice

agreement, the maximizing set is defined at each horizon; exact deterministic ties are resolved by the saved candidate order for the deployed choice, while set overlap provides a tie-aware supplementary statistic. Confidence intervals resample mother runs, keeping all their checkpoints and candidate outcomes together.

The terminal diagnostic measures the usefulness of a deployed action rather than continued use of a constant coefficient. We apply r for $\Delta = 20$K steps and then hand control to the same frozen TIDE gate for the remaining budget. Each branch may subsequently visit different states, but the continuation policy is identical as a function. Terminal win rate is measured using a reporting stream that the gate cannot observe. We compare its ranking across r with the original W-branch ranking and report the terminal regret of the candidate selected by the W-step label. This protocol directly matches the deployed action duration and measures candidate quality under the subsequent TIDE scheduling policy.

Table 2 in the main text summarizes the horizon correlations, top-choice agreement, and terminal regret with 95% confidence intervals. The estimates use a common checkpoint population across horizons and preserve dependence among checkpoints, candidate branches, and continuation seeds through mother-run resampling. Together, horizon consistency and terminal regret describe the persistence of candidate preferences and their usefulness when followed by adaptive scheduling. The positive correlations across horizons show that the 200K labels retain information about longer continuations, while terminal regret quantifies the performance gap associated with the selected action.

D Interpreting Local Preferences and Ranking Supervision

The distinction between local branch quality and terminal performance can be expressed through the performance-difference identity (Kakade & Langford, 2002). In the finite-horizon training process, let $Q_k^\pi(x, r)$ denote terminal task performance after choosing r for one control interval at decision k and following a reference scheduler π afterwards. Let $V_k^\pi(x)$ be the corresponding value under the reference scheduler, and let $d_k^{\widehat{\pi}}$ be the state distribution induced by a candidate scheduler $\widehat{\pi}$. For a common initial-state distribution, the finite-horizon identity gives

$$J(\widehat{\pi}) - J(\pi) = \sum_{k=0}^{K-1} \mathbb{E}_{x \sim d_k^{\widehat{\pi}}} \left[Q_k^\pi\big(x, \widehat{\pi}(\phi(x))\big) - V_k^\pi(x) \right]. \tag{8}$$

For a stochastic scheduler, the expectation also includes its chosen action. Time can be included in the full training state, and the final interval can be shortened to respect the training budget. Equation (8) expresses overall improvement as the sum of local action advantages along the training trajectories visited by the scheduler. It clarifies why the quality of decisions at successive training stages matters for final performance. TIDE obtains supervision through $g_W(x, r)$, which evaluates a fixed candidate over a finite continuation, and learns a decision rule from the observable statistics $\phi(x)$. The reliability study then examines how these preferences persist across horizons, and the terminal continuation experiment measures their quality under the deployed control interval and subsequent TIDE scheduling.

The candidate distributions in Equation (5) are related to probabilistic choice and ranking models. Luce's choice model assigns probability in proportion to a positive item weight (Luce, 2012). The Plackett ranking model extends successive choices to a probability distribution over orderings (Plackett, 1975). With exponential item weights, the probability of the first choice has the softmax form used in a top-one listwise objective. ListNet applies such distributions to supervised ranking (Cao et al., 2007), while subsequent work studies theoretical properties and algorithms for listwise losses (Xia et al., 2008).

These connections motivate a smooth objective for learning candidate preferences. TIDE constructs the target distribution from the observed continuation outcomes, allowing every candidate in a checkpoint group to contribute to supervision. The temperature controls how strongly the target concentrates on the better outcomes. This representation retains graded preferences when several candidates perform similarly and gives greater weight to candidates with larger measured advantages. Held-out agreement and regret evaluate the learned preferences, repeated continuations assess target reliability, and complete training runs measure the resulting scheduling performance.

E Scheduling Controls and Budget Accounting

E.1 Scheduling Baselines

All scheduling baselines retain the default coefficient during their cold start. The default controller uses $r = 1$ throughout. After cold start, linear decay uses $r_t = 2[1 - (t - T_c)/(T_{\max} - T_c)]$, clipped continuously to $[0, 2]$ rather than quantized to $\mathcal{R}$. Step decay uses $r = 1$ before the midpoint of training and $r = 0.25$ afterwards. The random gate samples a candidate uniformly every 20K steps.

The online UCB controller treats the five candidates as arms. Each arm is initialized once. Its feedback is the normalized improvement in the chosen surrogate over successive decision windows. The upper-confidence index is $\bar{x}_a + c\sqrt{2 \log n / n_a}$, with $c = 1$ selected on a held-out map. This controller provides an online feedback baseline that balances observed surrogate improvement with exploration of less frequently selected strengths. Its window-level updates complement the offline continuation supervision used by TIDE.

Table 10 gives the constant sweep on 3s5z_vs_3s6z. These constants are applied from the first training step. The independently tuned constant in Table 1 is selected on separate development runs, while the adaptive scheduling controls use the cold-start prefix specified above.

Table 10: Constant sweep on 3s5z_vs_3s6z with CIA. Standard deviations are unavailable for several constant choices, so only their reported means are shown.

Method	$r = 0$	$r = 0.25$	$r = 0.5$	$r = 1$	$r = 2$	TIDE
Mean final win rate	.176	.283	.259	.248	.141	.362

E.2 Controls for State Feedback

The additional controls in Table 1 use development data for all choices of candidates, switch times, model capacity, and selection rules; the five final evaluation seeds are held out from those choices.

For the tuned-constant control, we run each of the five candidates on three development seeds for 2M steps, select by mean final win rate, and evaluate the selected full-run constant on five held-out seeds. This uses 30M development steps. This development procedure fixes the coefficient before the final evaluation seeds are run.

The time-only gate consumes normalized time $t/T_{\max}$ and a candidate encoding. It uses the same branch groups, target outcomes, split, ranking loss, and stopping rule as TIDE. To keep nominal network size identical, time occupies one coordinate of an otherwise zero input vector with the original state dimension. It receives no loss, value, gradient, or reward-progress feature. This control therefore isolates the value of conditioning on observed training state beyond a learned function of time.

Frozen schedule replay uses five TIDE development trajectories with seeds disjoint from the final evaluation seeds. At each decision time, we take the median of their selected ratios in the ordered candidate set and freeze the resulting sequence. That sequence is then applied by time to each evaluation run, including the same cold-start prefix. Replay preserves the temporal pattern learned on the development runs, providing a direct comparison between a shared schedule and decisions conditioned on each run's current training state. Its extraction cost is reported separately from gate fitting.

The budget-matched search optimizes $\xi = (r_1, r_2, r_3, u_1, u_2)$, where $r_j \in \mathcal{R}$ and $0 < u_1 < u_2 < 1$ are switch fractions of the post-cold-start interval. The schedule uses r_1, then r_2, then r_3, with no monotonicity constraint. Switch fractions lie on the decision grid and each segment lasts at least five windows. This family contains increases, decreases, and suppression followed by reintroduction. A tree-structured Parzen estimator (Bergstra et al., 2011) proposes schedules using final win rate as its development objective. The first four proposals consist of the default schedule and three uniformly sampled valid schedules; the remaining seven use the estimator fitted to completed trials. Candidate strengths are categorical, switch times are discrete, and TIDE outcomes are not used to choose search proposals.

The search uses a 35M-step cap. We evaluate eleven proposed schedules on a first development seed for 2M steps each, then evaluate the best three on two further seeds for another 12M steps. The selected schedule is the finalist with the highest common three-seed 2M-step mean and is evaluated on five independent final seeds. This allocation uses 34M of the 35M available steps; all eleven initial proposals, including any default or duplicate schedule, count against the budget. The search optimizer uses only completed development outcomes.

E.3 Computational Cost and Reuse

Table 11 separates measured or reported cost estimates from control budgets. On the RTX 4090 setup used for timing, a 2M-step run takes roughly 10 to 12 GPU-hours. The collection estimate uses the lower end of that range and is approximate. Surrogate-selection branches, hyperparameter sweeps, checkpoint I/O, and control-validation episodes are additional whenever they are not included in the quoted estimate.

Table 11: Resource accounting per algorithm and map. Reported hardware times are approximate; control rows with unmeasured runtime show interaction budgets. Final evaluation runs are additional for every method.

Stage or control	Environment steps	Cost	Status
Mother runs	5×2M	~ 50 GPU-h	Reported estimate
Candidate continuations	125×0.2M	~ 125 GPU-h	Reported estimate
Core offline collection	35M	~ 175 GPU-h	Sum of preceding rows
Gate fitting	0	~ 14 CPU-min	Reported estimate
Five final training runs	10M	~ 50 to 60 GPU-h	Common additional work
Constant tuning, 5 choices and 3 seeds	30M	Not timed	Control budget
Schedule search, 11 initial and 6 repeat runs	34M	Not timed	Control budget, 35M cap
Five-value, five-seed sweep	50M	~ 250 GPU-h	Arithmetic reference only
Replay extraction, after gate fitting	10M	Not timed	Control budget

If the same gate is used for N subsequent runs, the core collection cost per run is approximately $175/N$ GPU-hours, in addition to each base training run and any control overhead. For example, reuse over $N = 20$ runs allocates approximately 8.75 GPU-hours of core collection to each run. Cross-map transfer provides another route to reuse: a source-trained controller can guide learning on a target map using the same feature interface and frozen preprocessing. The source collection budget remains part of the total cost accounting.

Gate forward-pass overhead is reported below 0.01%. Runtime accounting distinguishes these network evaluations from feature aggregation, data transfers, control evaluations, synchronization, and checkpoint storage. Candidate branches can run in parallel to shorten collection wall time; the total interaction budget and GPU-hours are recorded across all branches.

F Ablations and Hyperparameter Sensitivity

Table 12 reports the gate ablations, including the full-feature reference and the progress-only control. Regret is computed in the units of each block's outcome definition, and accuracy records agreement with the best candidate. These metrics describe the magnitude and frequency of candidate-selection errors, respectively.

Regret and accuracy emphasize different errors. If two candidates have nearly identical outcomes, choosing the second counts as a full accuracy error but incurs little regret. A predictor can therefore have lower regret and lower accuracy than another. The encoding ablation exhibits this pattern: scalar encoding has lower reported regret, while one-hot has higher accuracy. The deployed one-hot architecture provides the common configuration for the reported training runs, with 67.3% best-candidate agreement; scalar encoding achieves the lower validation regret of 0.0118.

Table 12: Gate ablations under the reported checkpoint-level validation protocol. Regret is the gap to the best candidate under that block's outcome definition; accuracy is best-candidate agreement. The reference row of each block is the deployed configuration.

Comparison	Variant	Regret	Accuracy
Loss	Listwise cross-entropy	.0147	67.3%
	Pointwise MSE	.0196	53.3%
	Pairwise ranking	.0213	56.0%
Candidate encoding	One-hot	.0147	67.3%
	Scalar	.0118	65.3%
	Polynomial	.0176	67.3%
Surrogate	Stability-selected	.0147	67.3%
	Raw return	.0891	61.3%
Feature families	Full features	.0147	67.3%
	Remove mechanism activation	.0218	62.7%
	Remove optimization statistics	.0181	65.3%
	Remove value statistics	.0275	60.0%
	Remove learning progress	.0392	53.3%
	Learning progress only	.0204	61.3%

Tables 13 and 14 address sensitivity to the two time scales. Intervals from 5K to 50K yield similar final performance, with the highest reported means at 20K on all three maps. Increasing the branch horizon from 50K to 200K improves both candidate-selection metrics and deployment performance, while the nominal 400K setting provides a smaller additional gain at a higher collection cost. In this sensitivity study, a branch starting at step t_c runs for $\min(W, T_{\max} - t_c)$ steps, so the nominal 400K branches are truncated at 2M steps. For the 500K-anneal checkpoint set, their lengths are 400K, 400K, 400K, 400K, and 250K; for the 50K-anneal set, they are 400K, 400K, 400K, 400K, and 300K. The 400K row therefore summarizes this budget-limited continuation protocol. The separate reliability experiment in Appendix C uses an earlier shared checkpoint population for full 200K, 400K, and 800K continuations.

Table 13: Reported decision-interval sensitivity for CIA. Entries are final win rates, mean $\pm$ standard deviation over five seeds.

Δ	3s5z_vs_3s6z	2c_vs_64zg	6h_vs_8z
5K	$.351 \pm .094$	$.884 \pm .121$	$.274 \pm .083$
10K	$.358 \pm .089$	$.889 \pm .117$	$.279 \pm .081$
20K	$.362 \pm .087$	$.891 \pm .118$	$.281 \pm .079$
50K	$.344 \pm .092$	$.878 \pm .126$	$.268 \pm .086$
100K	$.312 \pm .101$	$.861 \pm .133$	$.247 \pm .094$

Table 14: Branch-horizon sensitivity. Gate metrics use the checkpoint-level validation protocol. Deployment win rate and cost estimates are for CIA on 3s5z_vs_3s6z. W is nominal; branches stop at 2M steps.

W	Regret	Accuracy	Offline GPU-h	Final win rate
50K	.0341	51.3%	~ 81	$.291 \pm .104$
100K	.0208	61.3%	~ 113	$.334 \pm .092$
200K	.0147	67.3%	~ 175	$.362 \pm .087$
400K	.0139	68.7%	~ 281	$.366 \pm .085$

The cost column uses the approximate rate of 5 GPU-hours per million environment steps from Appendix E.3, rather than separate runtime measurements. For the nominal 400K setting on 3s5z_vs_3s6z, five mother runs and five candidates per checkpoint require $10 + 5 \times 5 \times (4 \times 0.4 + 0.25) = 56.25$M steps, giving approximately 281 GPU-hours after truncation.

Table 15 reports the temperature, capacity, and collection-size studies. The observed values favor moderate gate capacity and show diminishing returns to additional mother runs. These validation metrics isolate how each design choice affects candidate selection. With five mother runs and five checkpoints, the per-map corpus contains 25 groups; 125 is the number of candidate branches.

Table 15: Additional gate sensitivity results. Regret uses checkpoint-level evaluation.

Study	Configuration	Regret	Accuracy
Target temperature	Constant 0.5	.0189	63.3%
	Constant 1	.0171	64.7%
	Constant 2	.0224	58.7%
	Anneal $2 \to 0.5$	.0147	67.3%
	Anneal $0.5 \to 2$	.0246	56.0%
Hidden widths	$(16, 8)$	.0198	Not reported
	$(32, 12)$	.0154	Not reported
	$(40, 12)$	.0147	Not reported
	$(64, 32)$	.0163	Not reported
	$(128, 64)$	.0241	Not reported
Offline corpus per map	1 run, 5 groups	.0412	Not reported
	2 runs, 10 groups	.0271	Not reported
	3 runs, 15 groups	.0193	Not reported
	5 runs, 25 groups	.0147	Not reported
	8 runs, 40 groups	.0138	Not reported

G Behavioral Measurements

This section defines the behavioral evaluation used to interpret the measured trajectories in Figure 5. The objective is to observe policy differences separately from auxiliary-loss values and selected coefficient ratios. We use the same three CIA maps as the scheduling controls and compare the default learner, TIDE, time-only control, and frozen replay.

For a compatible pair of agents (i, j) and a common probe history h, let $p_i(\cdot \mid h)$ and $p_j(\cdot \mid h)$ be action distributions after aligning action semantics. Define

$$D_{\text{policy}} = \mathbb{E}_{h \sim \mathcal{P}} \left[\frac{1}{|\mathcal{C}(h)|} \sum_{(i,j) \in \mathcal{C}(h)} \text{JS}\left(p_i(\cdot \mid h), p_j(\cdot \mid h)\right) \right], \tag{9}$$

where $\mathcal{P}$ is a frozen probe distribution and $\mathcal{C}(h)$ includes only pairs with compatible unit types and action semantics. Probe histories without a compatible pair are excluded and their frequency reported. Jensen-Shannon divergence uses the mixture $m = (p_i + p_j)/2$ and equals $[\text{KL}(p_i \| m) + \text{KL}(p_j \| m)]/2$.

The probe set is constructed from development trajectories before comparing the final methods. Complete local histories are replayed to reconstruct recurrent hidden states under each evaluated policy. The same physical observation history, previous physical actions, target ordering, and availability mask are used when substituting agent identity; identity-conditioned inputs are changed consistently throughout the replayed history. We compare only identities of the same unit type. Attack actions refer to the same target entities, and incompatible movement or attack choices are not compared by their raw integer labels. The resulting diagnostic measures identity-conditioned policy differences on a shared, fixed distribution of observation histories.

For CIA's value-based policies, we evaluate a fixed ϵ-greedy distribution with $\epsilon = 0.05$ on the shared legal-action set. The exploration mass is distributed uniformly and ties in the greedy action are resolved deterministically. Keeping ϵ fixed avoids confounding policy differences with the base exploration schedule. This construction expresses policy differences through legal-action choices and the fixed exploration probability.

Action disagreement is the fraction of compatible agent pairs choosing different greedy actions on the same probe histories. With fixed ϵ and a fixed number of shared legal actions, the above JS

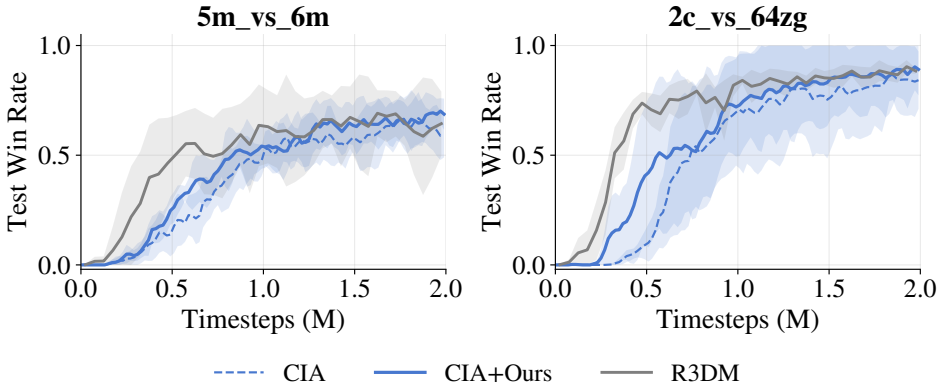


Figure 6: Comparison with R3DM on two SMAC maps. Results use five matched seeds and the same 2M-step base training budget.

divergence is zero for matching greedy actions and a fixed positive value otherwise. JS divergence and action disagreement thus provide two related summaries of identity-conditioned action choices. On-policy action disagreement on different local observations is reported separately, because it also reflects differences in situation. State-visitation divergence uses discretized relative position, distance to nearby enemies, and health bins collected during independent evaluation episodes. Its binning and smoothing are frozen using development data and results are stratified by unit type. It captures the differences in experience realized by agents during team execution.

Figure 5 reports policy-divergence and action-disagreement trajectories, with one column per map and uncertainty across training seeds. Its main curves compare the default coefficient, TIDE, and the time-only gate; frozen replay and state-visitation divergence are reported as supplementary diagnostics. Figure 4 provides the corresponding coefficient and win-rate trajectories. The comparison follows the temporal relationship among selected coefficients, policy differences, and task performance. A useful schedule can raise diversity during one phase and reduce it during another, so the trajectory records both the development of distinct behaviors and their later refinement. This view aligns the behavioral analysis with TIDE's objective of allocating diversity incentives throughout training.

H Additional Experiments and Schedule Summaries

H.1 SMACv2

Table 16 reports CIA results on six SMACv2 configurations. Each scenario uses its own offline collection and gate training, the same 2M-step budget, and five deployment seeds. The average gain across these six settings is 0.0747. These experiments extend the evaluation to procedurally generated StarCraft scenarios with varying unit compositions and team sizes. The positive gains in all six configurations show that adaptive coefficient scheduling also benefits learning when the scenario distribution varies within a task.

Table 16: Reported SMACv2 final win rates, mean ± standard deviation over five seeds. Gates are trained separately for each scenario.

Scenario	CIA	CIA+TIDE
Protoss 5 versus 5	$.482 \pm .089$	$.546 \pm .071$
Terran 5 versus 5	$.583 \pm .036$	$.640 \pm .029$
Zerg 5 versus 5	$.362 \pm .095$	$.434 \pm .076$
Protoss 10 versus 11	$.097 \pm .094$	$.186 \pm .108$
Terran 10 versus 11	$.172 \pm .101$	$.256 \pm .081$
Zerg 10 versus 11	$.211 \pm .068$	$.293 \pm .054$

H.2 Comparison with R3DM

Figure 6 shows the two-map comparison with R3DM (Goel et al., 2025). On 5m_vs_6m, CIA+TIDE has the higher reported final mean, approximately 0.69 ± 0.07 against 0.64 ± 0.15. On 2c_vs_64zg, both means round to 0.89, while R3DM has smaller reported seed dispersion and learns faster in the plotted curves. The comparison places coefficient scheduling alongside a method that structures agent roles. CIA+TIDE improves the final mean on 5m_vs_6m and reaches a comparable final mean on 2c_vs_64zg, showing that adapting the training signal is a useful route to stronger coordination. Both methods use the same 2M-step base training budget in this comparison; TIDE's offline collection is accounted for separately in Appendix E.3.

H.3 Schedules Across All Main Settings

Figure 7 summarizes the selected coefficients across all 18 main settings. Each panel reports the median and interquartile range across five training seeds, with gray regions marking the algorithm-specific cold start. The common candidate set, $\{0, 0.25, 0.5, 1, 2\}$, makes it possible to compare when each controller suppresses, attenuates, retains, or amplifies its base training signal. The panels complement the final win rates in Table 5 by showing how the coefficient is allocated over the course of learning.

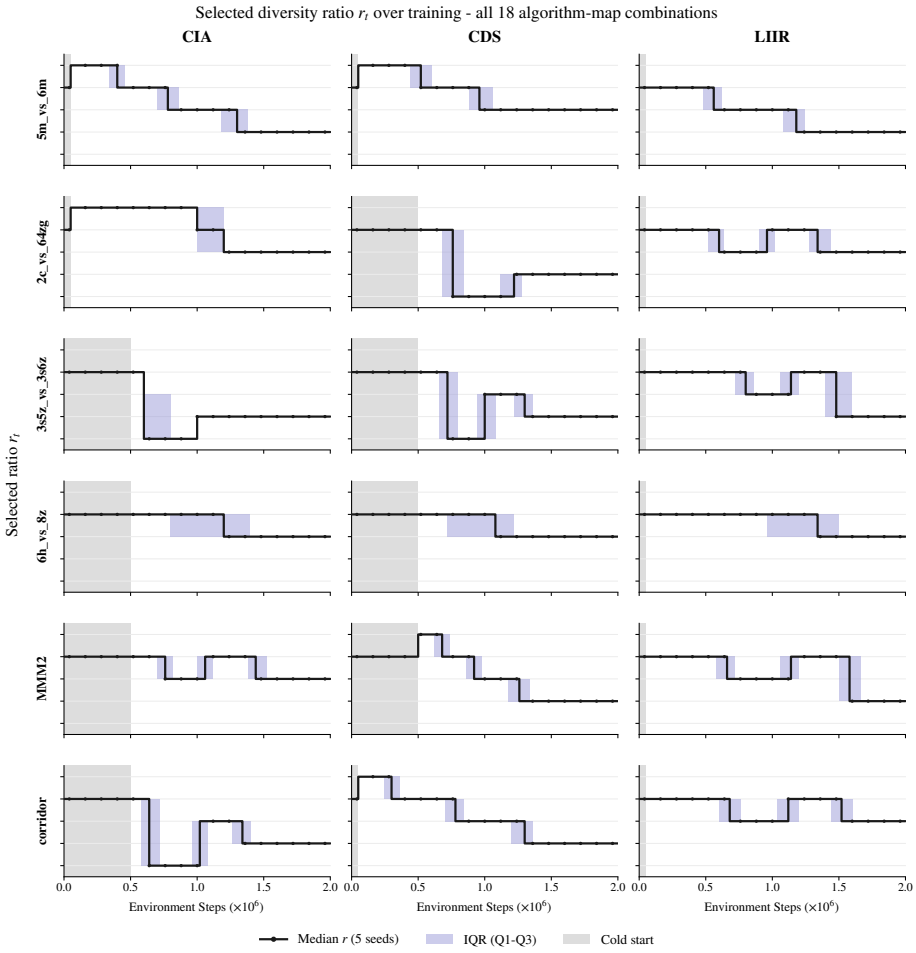


Figure 7: Diversity schedules across all 18 experimental settings.

The CIA examples discussed in the main text illustrate how to read these trajectories. On 3s5z_vs_3s6z, suppression is followed by renewed incentives; on 2c_vs_64zg, an early increase is followed by attenuation; and on 6h_vs_8z, the default coefficient is retained longer before it is reduced. These distinct patterns show why a shared action space can be useful without prescribing the same temporal schedule for every task. The gate selects among the same five ratios using the training statistics available in each setting.

The figure also connects the method's common interface to its evaluation across three base algorithms. CIA, CDS, and LIIR supply different mechanism features, while the optimization, value, and progress features retain a common role in scheduling. Combined with the positive mean gains in all 18 settings, the schedule summaries support adapting the strength of an existing learning signal as training progresses. The separately collected development schedules used for frozen replay provide the time-based counterpart to this feedback-driven procedure.