TEST-DRIVEN REWARD DESIGN FOR COOPERATIVE MULTI-AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In reinforcement learning (RL), especially in multi-agent reinforcement learning (MARL), reward design remains a fundamental challenge, as dense rewards often require task-specific engineering, while sparse rewards provide insufficient learning guidance for effective coordination. We observe that task completion can be characterized by whether agents satisfy task-level behavioral requirements, which can serve as implicit tests for evaluating cooperative behaviors. Based on this insight, we adapt the paradigm of Test-Driven Reinforcement Learning (TdRL) to the multi-agent setting and develop **T**est-**D**riven **M**ulti-**A**gent **R**einforcement **L**earning (TdMARL), a framework that constructs return functions from task-level evaluations and derives informative reward signals for the multi-agent system, enabling a more intuitive and systematic design process for complex cooperative environments. We provide theoretical analysis establishing the validity of extending the test-driven paradigm from single-agent RL to cooperative MARL. Furthermore, we introduce MARL-specific adaptations to TdRL, including tailored data sampling and preference-evaluation mechanisms, together with a soft uniform prior for stable return-to-reward decomposition, to better accommodate the sensitivity and non-stationarity of multi-agent systems. Experiments across dense reward, sparse reward, and sparse reward with auxiliary signals demonstrate that TdMARL provides effective reward guidance and achieves robust coordination performance across diverse reward settings. TdMARL offers a new perspective on reward design for cooperative MARL by shifting the focus from manually engineering scalar rewards to specifying task-level behavioral requirements.

1 INTRODUCTION

Cooperative multi-agent reinforcement learning (MARL) has made substantial progress through policy-gradient and value-decomposition methods that learn decentralized policies under centralized training (Yu et al., 2022; Rashid et al., 2020). Nevertheless, reward design remains central to the behaviors a team learns. In cooperative MARL, a reward must both specify desirable joint behavior and provide feedback that enables agents to discover it. Combining these two roles in a function makes reward design a critical yet difficult aspect of cooperative MARL (Booth et al., 2023), particularly when success depends on coordinated interactions among agents.

Dense rewards provide frequent guidance, but their design often relies heavily on domain expertise to identify useful intermediate behaviors, construct appropriate shaping terms, and carefully tune their relative weights. Moreover, arbitrary shaping can change the optimal policy (Ng et al., 1999), and agent-specific rewards may encourage locally useful skills at the expense of the intended team objective (Majumdar et al., 2020; Wang et al., 2022). Conversely, sparse team rewards offer the opposite trade-off by encoding success or failure through simple, task-aligned criteria, but provide little guidance for exploration and credit assignment (Booth et al., 2023; Jeon et al., 2022).

Existing work addresses different aspects of this challenge. Intrinsic-reward and exploration methods enrich sparse feedback through individual contributions, curiosity, causal influence, shared exploration objectives, or generated subgoals (Du et al., 2019; Zheng et al., 2021; Jeon et al., 2022; Liu et al., 2023; Li et al., 2023a; Na & Moon, 2024). Although they improve policy learning, these methods generally supplement a predefined task reward without separating behavioral requirements from learning guidance. Other MARL approaches express objectives through reward ma-

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.

chines or trajectory preferences, but rely on formal representations or offline preference datasets, respectively (Neary et al., 2020; Varricchione et al., 2026; Bui et al., 2025).

The goal of task completion rather than maximizing proxy rewards motivates our central question: *How can cooperative MARL transform simple task requirements into informative per-step guidance, overcoming both the design burden of dense rewards and the limited feedback of sparse rewards?*

As conceptually illustrated in Appendix A, our key observation is that specifying successful cooperation at the trajectory level is often easier than assigning scalar rewards to individual interactions. A designer only needs to encode task-aligned requirements in a tester, which then automatically evaluates whether joint trajectories achieve goals, complete relevant subtasks, or satisfy safety and performance criteria. Following Test-Driven Reinforcement Learning (TdRL) (Yu et al., 2026), pass-fail tests capture essential requirements, while indicative tests quantify task-relevant progress. This formulation separates task specification from the construction of informative learning guidance.

However, extending TdRL to cooperative MARL introduces both theoretical and practical challenges. TdRL's theoretical analysis assumes maximum-entropy optimization and does not directly address cooperative policy learning under decentralized execution (Yu et al., 2026). Meanwhile, rare cooperative successes can leave uniformly sampled pairs uninformative, and small indicator fluctuations can produce unreliable preferences as joint policies evolve. Matching segment returns alone does not uniquely determine how rewards should be distributed across timesteps.

To address these challenges, we propose **T**est-**D**riven **M**ulti-**A**gent **R**einforcement **L**earning (TdMARL). Reusable task-level tests automatically generate preferences over joint trajectories without human labeling. TdMARL learns returns consistent with these preferences and decomposes them into transition-level rewards for optimization. Tailored sampling and preference evaluation provide informative, reliable return-learning targets, while a soft uniform prior stabilizes reward decomposition. Theory connects test-derived returns to joint-policy optimization under stated assumptions.

Our contributions are threefold. (1) Theoretical analysis bounds test-failure probability using return-decomposition and policy-optimization errors and establishes compatibility with representative policy-gradient and value-decomposition methods. (2) TdMARL introduces tailored sampling, refined preference evaluation, and a soft uniform prior, stabilizing return learning and reward decomposition. (3) Evaluation across environments and reward settings demonstrates effective reward guidance and task-aligned coordination. Our code is accessible as an open-source repository[1].

2 Background

2.1 Related Works

We focus on work most relevant to TdMARL, with a broader discussion in Appendix B.

Reward Design in Cooperative MARL. Reward design in MARL must provide informative feedback while preserving alignment between individual behavior and the team objective. LIIR learns agent-specific intrinsic rewards by optimizing the environmental team return (Du et al., 2019). LAIES rewards individual and collaborative influence on external states through counterfactual reasoning based on a learned external-state transition model to discourage lazy behavior in sparse-reward tasks (Liu et al., 2023). FoX encourages diverse formation exploration and local formation awareness through formation-visitation bonuses and intrinsic rewards (Jo et al., 2024). These methods supplement environment-reward optimization with auxiliary incentives for exploration and coordination. TdMARL instead learns the optimization objective from task-level tests.

Task Specification and Test-Driven RL. Formal specification methods convert task descriptions into learning signals. Multi-agent reward machines encode and decompose temporally extended team objectives (Neary et al., 2020; Zhu et al., 2024). Temporal-specification methods construct dense reward monitors from quantitative temporal logic (Adalat & Belardinelli, 2026). They provide task semantics, but require logical formulas, event labels, automata, or environment abstractions. TdRL instead uses pass-fail and indicative tests to learn trajectory returns, then fits per-step rewards whose sum matches these returns (Yu et al., 2026). Our work extends this principle to cooperative joint trajectories and introduces mechanisms tailored to multi-agent non-stationarity.

[1] https://anonymous.4open.science/r/TdMARL-4FBB

Preference- and Feedback-Based Reward Learning. Preference-based MARL learns objectives from comparisons. O-MAPL learns offline policies without an explicit reward model (Bui et al., 2025), while M^3HF uses iterative human feedback (Wang et al., 2025). MAGPIE aggregates agent-specific rewards learned from online preferences into team rewards (Mu et al., 2026). TdMARL instead constructs hierarchical joint-trajectory preferences from task-level tests, prioritizing pass-fail requirements over indicative progress. It learns joint returns before temporal reward decomposition, with tailored sampling and robust preference evaluation accommodating evolving joint policies.

2.2 Decentralized Partially Observable Markov Decision Process

A cooperative multi-agent task is formulated as a decentralized partially observable Markov decision process (Dec-POMDP) (Bernstein et al., 2002), represented by $\mathcal{G} = \langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}_i\}_{i=1}^n, P, r, \{\mathcal{O}_i\}_{i=1}^n, Z, \gamma \rangle$. Here, $\mathcal{N} = \{1, \ldots, n\}$ is the agent set, $\mathcal{S}$ is the global state space, and $\mathcal{A}_i$ and $\mathcal{O}_i$ are agent i's action and observation spaces. P, r, Z, and γ denote the transition function, shared reward function, observation function, and discount factor, respectively. At timestep t, agent i observes o_t^i and selects a_t^i from $\tau_t^i = (o_0^i, a_0^i, \ldots, a_{t-1}^i, o_t^i)$. The joint action $\mathbf{a}_t = (a_t^1, \ldots, a_t^n)$ yields a reward r_t and induces the next state according to $P(s_{t+1} \mid s_t, \mathbf{a}_t)$. Under decentralized execution, the joint policy factorizes as $\boldsymbol{\pi}(\mathbf{a}_t \mid \boldsymbol{\tau}_t) = \prod_{i=1}^n \pi_i(a_t^i \mid \tau_t^i)$, where $\boldsymbol{\tau}_t = (\tau_t^1, \ldots, \tau_t^n)$. The objective is to maximize the discounted return $J(\boldsymbol{\pi}) = \mathbb{E}_{\boldsymbol{\pi}, P}[\sum_{t \geq 0} \gamma^t r_t]$.

2.3 Value Decomposition and Policy Optimization in MARL

Under centralized training with decentralized execution (CTDE), global information can be used to facilitate training, while each agent makes decisions from its local action-observation history during execution. We consider two representative CTDE algorithms as the policy learners for TdMARL.

QMIX. QMIX is an off-policy value-decomposition method that represents the joint action-value function through a monotonic mixing network (Rashid et al., 2020):

$$Q_{\text{tot}}(\boldsymbol{\tau}_t, \mathbf{a}_t, s_t) = f_{\text{mix}}\left(Q_1(\tau_t^1, a_t^1), \ldots, Q_n(\tau_t^n, a_t^n) \mid s_t\right), \qquad \partial Q_{\text{tot}} / \partial Q_i \geq 0. \tag{1}$$

The mixer uses the global state, while each Q_i uses only local information. Monotonicity enables decentralized greedy execution, with Q_{tot} learned through off-policy temporal-difference updates.

MAPPO. MAPPO combines decentralized actors $\pi_{\theta_i}(a_t^i \mid \tau_t^i)$ with a centralized critic $V_\nu(s_t)$ (Yu et al., 2022). The actors are updated using:

$$\mathcal{L}_{\text{clip}}(\boldsymbol{\theta}) = \mathbb{E}_{t,i}\left[\min\left(\rho_t^i \hat{A}_t, \text{clip}(\rho_t^i, 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right], \tag{2}$$

where ρ_t^i is the policy ratio and $\hat{A}_t$ is the shared advantage estimated using centralized information.

QMIX and MAPPO provide complementary off-policy value-based and on-policy actor-critic backbones. TdMARL retains their original optimization structures and supplies test-derived rewards, resulting in the Td-QMIX and Td-MAPPO variants used in our experiments.

2.4 Preference-Based Reinforcement Learning

Preference-based reinforcement learning (PbRL) learns objectives from trajectory comparisons rather than predefined scalar rewards (Christiano et al., 2017). Let $\boldsymbol{\tau} = (s_0, \mathbf{a}_0, \ldots, s_T)$ denote a joint trajectory, with $\widehat{R}_\psi(\boldsymbol{\tau})$ as its return. The relation $\boldsymbol{\tau}^1 \succ \boldsymbol{\tau}^2$ indicates that $\boldsymbol{\tau}^1$ is more desirable. Under the Bradley–Terry model (Bradley & Terry, 1952), the preference probability is:

$$P_\psi\left[\boldsymbol{\tau}^1 \succ \boldsymbol{\tau}^2\right] = \left[1 + \exp\left(\widehat{R}_\psi(\boldsymbol{\tau}^2) - \widehat{R}_\psi(\boldsymbol{\tau}^1)\right)\right]^{-1}. \tag{3}$$

Given labeled trajectory pairs, the return model is trained using cross-entropy. TdMARL derives these labels automatically from task-level tests rather than human feedback.

3 From TdRL to TdMARL: Theoretical Foundation

TdRL learns objectives from task-level evaluations instead of manually designed engineered rewards (Yu et al., 2026). Extending this principle to cooperative MARL requires connecting test-

induced joint-trajectory returns to factorized policy optimization under CTDE. We establish this connection through policy-level objective alignment and MAPPO compatibility.

Let $z_{\mathrm{pf}}^{k} : \mathcal{T} \to \{0, 1\}$ denote the k-th pass-fail test. The satisfactory joint-trajectory set is:

$$\mathcal{T}^{+} = \left\{ \boldsymbol{\tau} \in \mathcal{T} \,\middle|\, z_{\mathrm{pf}}^{k}(\boldsymbol{\tau}) = 1 \text{ for all } k \in \{1, \ldots, K_{\mathrm{pf}}\} \right\}. \tag{4}$$

Let Π_{Θ} be the factorized policy class and let $\mathcal{T}_{\Theta} = \bigcup_{\boldsymbol{\pi} \in \Pi_{\Theta}} \mathrm{supp}(P_{\boldsymbol{\pi}})$ be its trajectory support.

Definition 1. *A bounded return $\widehat{R}_{\psi}$ is Δ-test-separated if the following inequality holds for some $\Delta > 0$:*

$$\widehat{R}_{\psi}(\boldsymbol{\tau}^{+}) \geq \widehat{R}_{\psi}(\boldsymbol{\tau}^{-}) + \Delta, \tag{5}$$

for every $\boldsymbol{\tau}^{+} \in \mathcal{T}^{+} \cap \mathcal{T}_{\Theta}$ and $\boldsymbol{\tau}^{-} \in \mathcal{T}_{\Theta} \setminus \mathcal{T}^{+}$.

Pass-fail tests impose this success boundary, while indicative tests refine comparisons not resolved by pass-fail outcomes. The learned per-step reward induces the discounted segment return:

$$\widehat{G}_{\phi}(\boldsymbol{\tau}) = \sum_{t=0}^{T-1} \gamma^{t} r_{\phi}(s_t, \mathbf{a}_t), \tag{6}$$

where s_t and $\mathbf{a}_t$ denote the global state and joint action available during centralized training. Its return-decomposition error under $\boldsymbol{\pi}$ is:

$$\varepsilon_{\boldsymbol{\pi}} = \mathbb{E}_{P_{\boldsymbol{\pi}}} \left[\left| \widehat{G}_{\phi}(\boldsymbol{\tau}) - \widehat{R}_{\psi}(\boldsymbol{\tau}) \right| \right]. \tag{7}$$

Proposition 1. *Suppose that $\widehat{R}_{\psi}$ is Δ-test-separated. Let $C = \sup_{\boldsymbol{\tau} \in \mathcal{T}_{\Theta}} \widehat{R}_{\psi}(\boldsymbol{\tau})$ and assume that some $\boldsymbol{\pi}^{+} \in \Pi_{\Theta}$ generates satisfactory trajectories with return C almost surely. For any δ-optimal policy $\widehat{\boldsymbol{\pi}}$ of $\widehat{J}_{\phi}(\boldsymbol{\pi}) = \mathbb{E}_{P_{\boldsymbol{\pi}}}[\widehat{G}_{\phi}(\boldsymbol{\tau})]$, the following bound holds:*

$$P_{\widehat{\boldsymbol{\pi}}}(\boldsymbol{\tau} \notin \mathcal{T}^{+}) \leq \min\left\{1, \Delta^{-1}\left(\varepsilon_{\widehat{\boldsymbol{\pi}}} + \varepsilon_{\boldsymbol{\pi}^{+}} + \delta\right)\right\}. \tag{8}$$

Thus, under the stated realizability assumption, reducing decomposition error and policy suboptimality tightens the failure-probability bound, while a larger Δ strengthens the guarantee.

Proposition 2. *For a fixed r_{ϕ} and factorized joint policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\mathbf{a}_t \mid \boldsymbol{\tau}_t) = \prod_{i=1}^{n} \pi_{\theta_i}(a_t^i \mid \tau_t^i)$, let $A_{\phi,t}^{\boldsymbol{\pi}}$ be the exact centralized advantage. The policy gradient with respect to θ_i is:*

$$\nabla_{\theta_i} \widehat{J}_{\phi} = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{t=0}^{T-1} \gamma^{t} \nabla_{\theta_i} \log \pi_{\theta_i}(a_t^i \mid \tau_t^i) A_{\phi,t}^{\boldsymbol{\pi}} \right]. \tag{9}$$

At the behavior policy, $\rho_t^i = 1$ and clipping is locally inactive. Thus, with exact advantages and the same discounted time weighting, the MAPPO surrogate gradient equals the policy gradient above.

The propositions connect test-derived returns to MAPPO optimization, with proofs provided in Appendix C.1, while Appendix C.2 establishes Td-QMIX compatibility under monotonic factorization.

4 Methodology

4.1 TdMARL Overview

Figure 1 illustrates the framework, while Appendix D provides Algorithm 1 and reports reasonable training-time costs for TdMARL. The procedure begins by collecting joint rollouts and storing each episode batch in the MARL training buffer $\mathcal{B}_M$ and the return buffer $\mathcal{B}_R$. Subsequent training samples segments from $\mathcal{B}_R$, constructs test-induced preferences, learns segment returns, decomposes them into per-step rewards, and relabels the rewards in $\mathcal{B}_M$. Instantiating this process with QMIX or MAPPO yields Td-QMIX or Td-MAPPO, which retain standard off-policy temporal-difference or on-policy clipped policy-gradient updates, respectively, as detailed in Appendix E. The return and reward models are updated alongside the MARL learner as new joint trajectories are collected. The test-driven pipeline is confined to centralized training, leaving decentralized execution unchanged.

Furthermore, successful joint trajectories are often rare in MARL, so the uniform sampling used in TdRL yields few informative comparisons. TdMARL therefore combines 50% uniformly sampled pairs with 50% targeted pairs that differ on the least-passed test. Pass rates are periodically recomputed from $\mathcal{B}_R$ so that targeted sampling follows the test currently most difficult for the agents. Section 5.3 empirically evaluates the effectiveness of this sampling strategy.

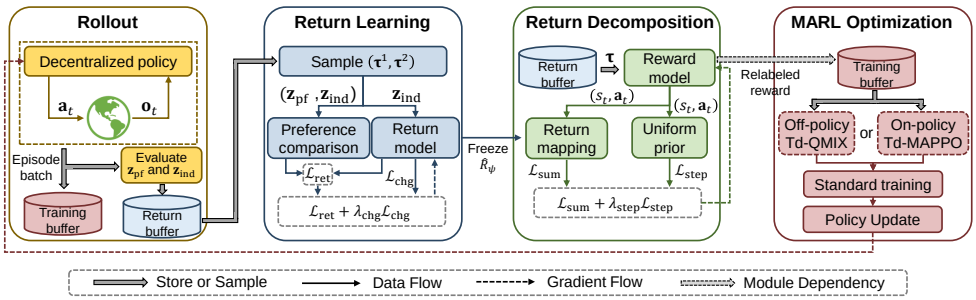

Figure 1: Overview of TdMARL, including joint rollout collection, test-based return learning, return decomposition, and MARL optimization.

4.2 Test-Based Joint Return Construction

Given a batch of sampled joint-trajectory pairs, TdMARL evaluates each trajectory using K_{pf} pass-fail tests and K_{ind} indicative tests. The resulting test outcomes are represented as:

$$\begin{aligned} \mathbf{z}_{\text{pf}}(\boldsymbol{\tau}) &= \left(z_{\text{pf}}^{k}(\boldsymbol{\tau})\right)_{k=1}^{K_{\text{pf}}} \in \{0,1\}^{K_{\text{pf}}}, \\ \mathbf{z}_{\text{ind}}(\boldsymbol{\tau}) &= \left(z_{\text{ind}}^{\ell}(\boldsymbol{\tau})\right)_{\ell=1}^{K_{\text{ind}}} \in \mathbb{R}^{K_{\text{ind}}}. \end{aligned} \tag{10}$$

Pass-fail tests identify whether required cooperative behaviors are satisfied, while indicative tests measure task-relevant progress and are oriented so that larger values represent better behavior.

Each pair $(\boldsymbol{\tau}^1, \boldsymbol{\tau}^2)$ is assigned a preference label $y_{12} \in \{0, 0.5, 1\}$ from the pass-fail and indicative outcomes $\mathbf{z}_{\text{pf}}$ and $\mathbf{z}_{\text{ind}}$. Here, $y_{12} = 0$, $y_{12} = 1$, and $y_{12} = 0.5$ indicate preference for $\boldsymbol{\tau}^1$, preference for $\boldsymbol{\tau}^2$, and a tie, respectively. The preference label is determined hierarchically: **(1)** the trajectory passing more pass-fail tests is preferred. **(2)** If the pass counts are equal, pass-fail tests are examined in ascending order of empirical pass rate, and the first differing outcome determines the preference. **(3)** If unresolved, a trajectory is preferred when all its indicative outcomes exceed those of the other by more than ϵ_{pref}. **(4)** Otherwise, the indicative outcomes are variance-normalized and examined in descending order of empirical skewness, with the first difference exceeding ϵ_{pref} determining the preference. Pairs that remain indistinguishable are labeled as ties.

Compared with the TdRL rule, TdMARL introduces three adaptations. It continues to compare indicative outcomes when both trajectories pass all pass-fail tests, applies a tolerance ϵ_{pref} to prevent fluctuations from producing unreliable preferences, and floors normalization at $\sigma_{\min}$ to avoid amplifying low-variance indicators. The effect of these adaptations is examined in Section 5.3.

The preference labels supervise the learning of segment-level returns. Specifically, the return model aggregates the indicative outcomes as $\widehat{R}_\psi(\boldsymbol{\tau}) = f_\psi(\mathbf{z}_{\text{ind}}(\boldsymbol{\tau}))$. For each pair, log-softmax is applied to $\left[\widehat{R}_\psi(\boldsymbol{\tau}^1), \widehat{R}_\psi(\boldsymbol{\tau}^2)\right]$ to obtain the log probabilities $\ell_{\psi,1}$ and $\ell_{\psi,2}$. The return model is trained by minimizing the following binary cross-entropy loss over the labeled pair dataset $\mathcal{D}$:

$$\mathcal{L}_{\text{ret}}(\psi) = -\mathbb{E}_{\mathcal{D}}\left[(1 - y_{12})\ell_{\psi,1} + y_{12}\ell_{\psi,2}\right]. \tag{11}$$

To stabilize learning, the change loss penalizes deviations from the pre-update prediction $\overline{R}_\psi(\boldsymbol{\tau})$:

$$\mathcal{L}_{\text{chg}}(\psi) = \mathbb{E}_{\mathcal{D}}\left[\sum_{q=1}^{2}\left(\widehat{R}_\psi(\boldsymbol{\tau}^q) - \overline{R}_\psi(\boldsymbol{\tau}^q)\right)^2\right]. \tag{12}$$

The return-model parameters ψ are optimized using $\mathcal{L}_{\text{RT}}(\psi) = \mathcal{L}_{\text{ret}}(\psi) + \lambda_{\text{chg}}\mathcal{L}_{\text{chg}}(\psi)$. During each round, training stops when $\|\nabla_\psi \mathcal{L}_{\text{chg}}\|_2 > c_{\text{es}}\|\nabla_\psi \mathcal{L}_{\text{ret}}\|_2$, preventing excessive deviation from the frozen predictions. Figure 4 empirically validates the effectiveness of the learned return model.

4.3 Return-to-Reward Decomposition

The learned return assigns a scalar score to an entire segment containing $T_{\boldsymbol{\tau}}$ valid transitions, whereas cooperative MARL requires a shared reward at each timestep. TdMARL learns a reward

model $r_\phi(s_t, \mathbf{a}_t)$ that maps the global state and joint action to a per-step reward. Following the return-matching principle of TdRL (Yu et al., 2026), the reward model minimizes:

$$\mathcal{L}_{\text{sum}}(\phi) = \mathbb{E}_{\boldsymbol{\tau}\sim\mathcal{B}_R}\left[\left(\widehat{R}_\psi(\boldsymbol{\tau}) - \sum_{t=0}^{T_{\boldsymbol{\tau}}-1}\gamma^t r_\phi(s_t, \mathbf{a}_t)\right)^2\right]. \tag{13}$$

Since return matching alone may produce large rewards that cancel across timesteps, TdMARL introduces a soft uniform prior. Let $Z_{\gamma,\boldsymbol{\tau}} = \sum_{t=0}^{T_{\boldsymbol{\tau}}-1}\gamma^t$. The prior is defined as:

$$\mathcal{L}_{\text{step}}(\phi) = \mathbb{E}_{\boldsymbol{\tau}\sim\mathcal{B}_R}\left[T_{\boldsymbol{\tau}}^{-1}\sum_{t=0}^{T_{\boldsymbol{\tau}}-1}\left(r_\phi(s_t, \mathbf{a}_t) - Z_{\gamma,\boldsymbol{\tau}}^{-1}\widehat{R}_\psi(\boldsymbol{\tau})\right)^2\right]. \tag{14}$$

The objective is $\mathcal{L}_{\text{RW}}(\phi) = \mathcal{L}_{\text{sum}}(\phi) + \lambda_{\text{step}}\mathcal{L}_{\text{step}}(\phi)$. The prior stabilizes decomposition by biasing per-step rewards toward a uniform level consistent with the discounted segment return, preserving state-action dependence. Predicted rewards relabel $\mathcal{B}_M$ before optimization. Appendix F empirically validates the decomposition, while Section 5.3 establishes the prior's necessity in MARL.

5 Experiments

Our experiments evaluate TdMARL's effectiveness across reward settings, its alignment with specified cooperative behaviors, contributions of its components, and sensitivity to key hyperparameters.

5.1 Experimental Setup

We evaluate TdMARL on three cooperative benchmarks spanning complementary reward settings. **MPE.** We consider two Multi-Agent Particle Environment tasks (Lowe et al., 2017). Cooperative Navigation requires three agents to cover three landmarks, while Predator–Prey requires three agents to capture a faster and more agile prey. Both tasks evaluate test-driven reward construction using tests aligned with the objectives of engineered dense rewards. **LBF.** In Level-Based Foraging (LBF) (Papoudakis et al., 2021), two level-1 agents must cooperate to collect a level-2 fruit. This task evaluates whether auxiliary tests induce richer cooperative behaviors. **SMACv2.** We adopt fully sparse variants of the Protoss, Terran, and Zerg 5-vs-5 scenarios from SMACv2 (Ellis et al., 2023) to assess whether simple survival and elimination tests provide effective coordination guidance. Additionally, results for the larger 10-vs-10 and 20-vs-20 scenarios are provided in Appendix H.4. Environment configurations, reward settings, and test specifications are provided in Appendix G.1.

We select QMIX (Rashid et al., 2020) and MAPPO (Yu et al., 2022) as widely adopted off-policy value-decomposition and on-policy policy-gradient backbones, respectively. Integrating TdMARL yields Td-QMIX and Td-MAPPO, which are compared with their backbones in LBF and MPE to assess backbone-dependent performance. On SMACv2, we additionally include MAPPO-based LAIES (Liu et al., 2023) and QMIX-based FoX (Jo et al., 2024) as sparse-reward exploration baselines. Their intrinsic rewards encourage counterfactual influence on external states and formation diversity and awareness, respectively. QMIX-Dense and MAPPO-Dense use environment-provided dense rewards for informative per-step guidance as oracle references. Method configurations, hyperparameters, evaluation protocols, and computational resources are reported in Appendix G.2.

5.2 Main Results

The main results are organized by how task information is specified for TdMARL's test-based reward construction. In MPE, the test specifications follow the behavioral objectives encoded by engineered dense rewards. In SMACv2, the tests are defined from sparse terminal requirements. LBF extends the sparse success condition with auxiliary behavioral signals. Since TdMARL learns from task-level evaluations, the main text emphasizes metrics that directly reflect the specified behaviors. To complement these metrics with a conventional performance measure, Appendix H.1 reports training curves measured by the environment rewards. All curves average four random seeds and 32 test episodes per seed at each evaluation. Shaded regions denote 95% confidence intervals across seeds.

Dense-Reward Tasks: MPE. MPE examines whether test-derived rewards can induce the cooperative behaviors targeted by carefully engineered dense rewards. On Cooperative Navigation, Figure 11 and Table 5 in Appendix H.2 identify Td-MAPPO as the strongest landmark coordinator.

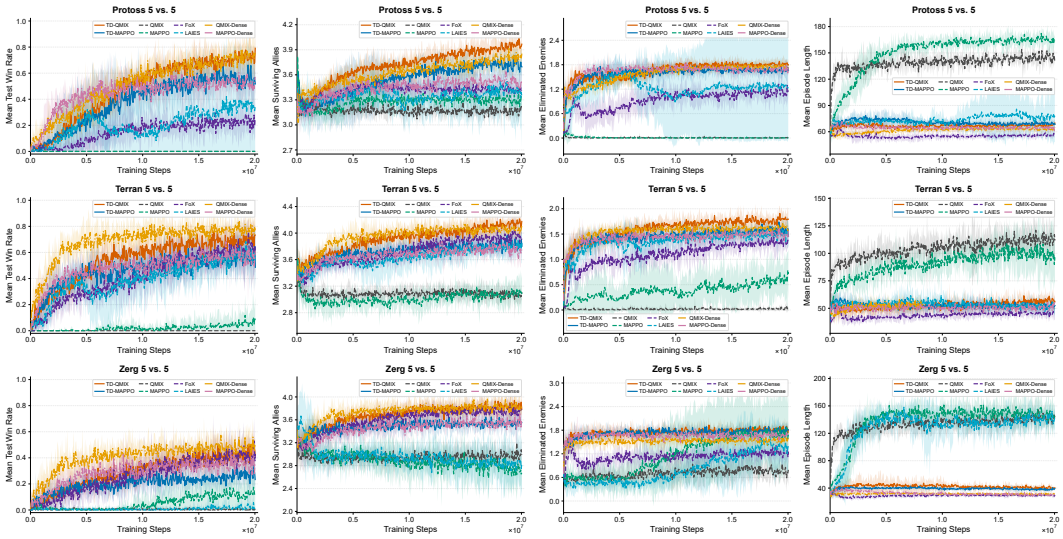

Figure 2: Performance on the SMACv2 5-vs-5 scenarios, including win rate, allied units surviving, enemy units eliminated, and episode length.

Relative to MAPPO, it increases sustained coverage from 0.23 to 0.55, a 139% improvement, and proximity from 0.69 to 0.76. Consistent with these gains, Figure 12 shows that the policy learned by Td-MAPPO directs the agents to reach the landmarks by timestep 20 and maintain coverage through timestep 79. By comparison, both QMIX-based methods perform poorly on landmark coverage. Although Td-QMIX achieves the highest safe separation rate of 0.93, its sustained coverage and proximity remain low at 0.03 and 0.46, while QMIX reaches only 0.11 coverage. Figure 13 shows that its learned policy fails to maintain stable landmark assignments. This backbone dependence is a limitation of TdMARL, motivating future work on MARL algorithms tailored to test-driven rewards.

A similar backbone-dependent pattern emerges on Predator–Prey. Figure 14 and Table 6 in Appendix H.2 show improved pursuit behavior for both Td variants, with Td-MAPPO achieving the largest gains. It increases proximity from 0.58 to 0.64 and mean prey-contact frames from 0.77 to 1.93 relative to MAPPO. Td-QMIX likewise improves proximity from 0.55 to 0.61 and contact frames from 0.02 to 0.22, although its capture performance remains limited. This gap is reflected in the rollouts. Figure 15 shows that the policy learned by Td-MAPPO drives predators to approach from multiple directions and form an encircling pursuit pattern, whereas Figure 16 indicates the policy learned by Td-QMIX produces less structured movement without stable containment.

Importantly, the primary test outcomes map directly to the intended task behaviors: sustained coverage measures persistent landmark occupancy, while prey-contact frames measure successful interception. Their improvements are realized in task execution rather than merely reflected in surrogate scores. Taken together, these results demonstrate that test-derived rewards can induce meaningful coordination policies, while their quality remains influenced by the underlying MARL backbone.

Fully Sparse Rewards: SMACv2. SMACv2 tests whether simple task-level evaluations can guide complex cooperative combat. The tester checks allied survival and complete enemy elimination, while indicative tests average surviving allies and eliminated enemies over each trajectory. Figure 2 and Table 7 (Appendix H.3) compare Td-QMIX and Td-MAPPO with their sparse backbones, exploration baselines FoX and LAIES, and dense-reward references across the Protoss, Terran, and Zerg 5-vs-5 scenarios. The sparse backbones learn little useful behavior: QMIX remains near zero win rate, while MAPPO reaches at most 0.12. FoX obtains win rates of 0.23, 0.62, and 0.41, compared with 0.72, 0.69, and 0.43 for Td-QMIX. LAIES reaches 0.35, 0.57, and 0.02, compared with 0.55, 0.57, and 0.27 for Td-MAPPO, showing a particularly large gap on Zerg. Td-QMIX closely approaches the QMIX-Dense results of 0.72, 0.77, and 0.48. Td-MAPPO likewise approaches MAPPO-Dense, which achieves 0.53, 0.57, and 0.35. Across the scenarios, the test-driven methods outperform their dense counterparts in most allied-survival and enemy-elimination compar-

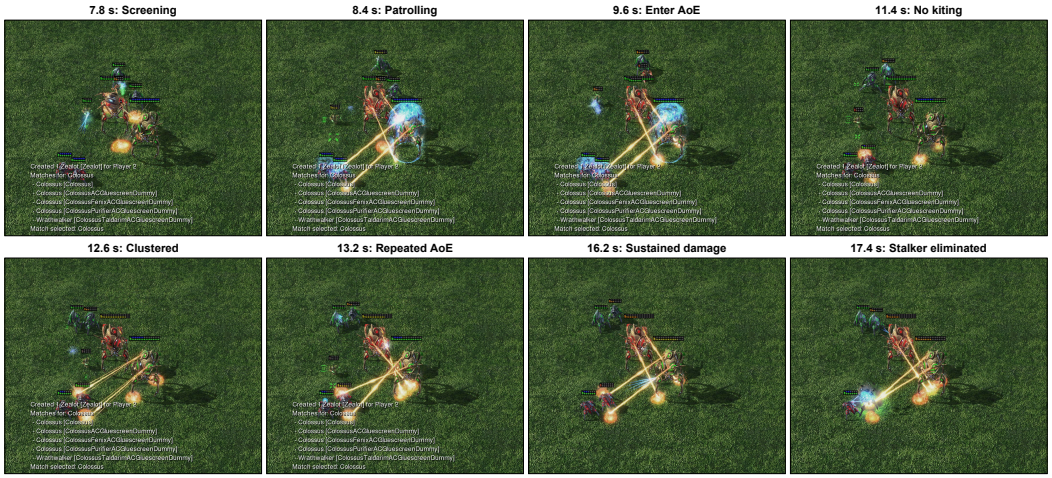

Figure 3: Keyframes from a 24-second Protoss 5-vs-5 victory by Td-MAPPO.

isons, showing that the learned policies realize the tester-specified behaviors. Overall, simple tests tied directly to task requirements yield win rates comparable to complex dense-reward engineering.

Representative keyframes in Figure 3 show how the policy learned by Td-MAPPO realizes the test objectives during a successful 24-second Protoss 5-vs-5 battle. A Zealot screens and moves laterally near the opposing Stalkers, while the Colossus maintains a firing position and the allied Stalker attacks from the flank. After entering the Colossus's area of effect at 9.6 seconds, the opposing Stalkers remain clustered and fail to kite, exposing both units to repeated damage until one is eliminated at 17.4 seconds. This progression reflects role-differentiated coordination, where frontline screening restricts enemy movement and ranged units exploit the resulting formation. It shows that Td-MAPPO translates the tester's elimination and survival criteria into cooperative combat behavior.

Figure 4 visualizes learned returns over enemy elimination and allied survival. Across settings, high returns track greater allied survival and enemy elimination, and winning trajectories cluster in high-return regions. On average, winning trajectories score 1.50–1.69 standard deviations above losses. Only 3.7%–11.3% of win–loss comparisons assign a win an equal or lower return. This supports alignment between learned returns and tester-defined preferences. Appendix H.4 shows that TdMARL remains effective in 10-vs-10 scenarios, whereas its 20-vs-20 learning curves are still rising at the end of the same training budget, resulting in lower final win rates. This slower convergence limits learning efficiency for larger teams and motivates future improvements.

Sparse Rewards with Auxiliary Signals: LBF. LBF examines whether TdMARL can extend a sparse task specification to induce richer cooperative behaviors. Successful pickup corresponds to the original sparse reward, while agent pairing, simultaneous fruit coverage, and coordinated loading attempts describe the preceding cooperative process. Figure 18 and Table 9 in Appendix H.5 show that the Td variants maintain comparable pickup performance, with Td-QMIX achieving 8.93 pickups versus 9.16 for QMIX and Td-MAPPO achieving 9.84 versus 10.02 for MAPPO. The clearest behavioral improvement appears in agent pairing, where Td-QMIX increases the score from 0.35 to 0.41 and Td-MAPPO from 0.39 to 0.40. The double-cover score also increases from 0.31 to 0.32 for both backbones, while the load-attempt frequency remains comparable. Moreover, for both Td variants, the three auxiliary metrics rise early in training and then decline, while successful pickups continue to improve. Since these metrics are averaged over fixed 50-step episodes and each pickup respawns the fruit at a new location, early policies may inflate their values by lingering near a fruit or repeatedly executing LOAD. More efficient policies complete pickups quickly and spend more time relocating, so the decline reflects reduced redundant behavior rather than degraded cooperation.

Figures 19 and 20 in Appendix H.5 show that QMIX agents separate more widely after fruit respawn, whereas Td-QMIX tends to remain paired while relocating. These results demonstrate that auxiliary tests can induce targeted cooperative behaviors without materially compromising task performance.

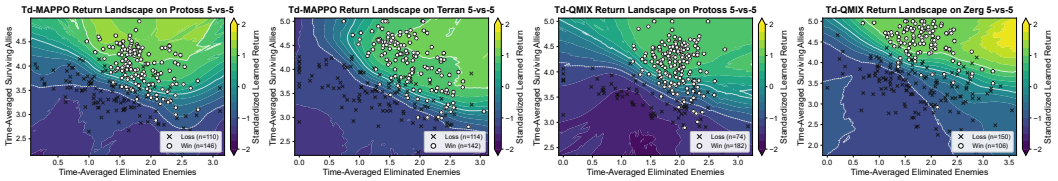

Figure 4: Test-induced return landscapes on SMACv2 using 256 trajectories per panel. Circles and crosses denote wins and losses, while colors show standardized learned returns.

5.3 Ablation Study

Figure 21 and Table 10 in Appendix I evaluate the multi-agent preference evaluator, targeted sampling, two-stage return-to-reward learning, and soft uniform prior. The direct-training variant bypasses the return model and decomposition stage, training the per-step reward network from preferences over aggregated rewards. Overall, the complete method achieves the best or near-best performance across all six comparisons. Removing the soft uniform prior reduces Protoss win rates from 0.72 to 0.20 for Td-QMIX and from 0.55 to 0.09 for Td-MAPPO, and Td-MAPPO's LBF pickups from 9.84 to 7.96. Replacing targeted sampling with TdRL sampling reduces Td-MAPPO's sustained coverage rate from 0.55 to 0.31 on CN, mean successful pickups from 9.84 to 8.25 on LBF, and win rate from 0.55 to 0.39 on Protoss. The original TdRL preference rule reduces Td-MAPPO's CN sustained coverage rate from 0.55 to 0.01, while direct reward training lowers this rate to 0.02 and mean successful pickups on LBF from 9.84 to 7.88. These results show the combined adaptations provide more reliable performance across tasks and MARL backbones.

5.4 Hyperparameter Sensitivity

The default hyperparameters are shared across all environments, including the return-model early-stop coefficient $c_{\text{es}} = 10$ and soft-uniform-prior coefficient $\lambda_{\text{step}} = 1$. Only the return-buffer size B_R differs due to trajectory segmentation, using 256×4 for LBF and Cooperative Navigation and 512×1 elsewhere. Figure 22 and Table 11 in Appendix J examine these parameters. Very small buffers cause severe degradation. On LBF, reducing B_R from 256×4 to 32×4 lowers performance from 8.93 and 9.84 to 0.03 and 0.00 for Td-QMIX and Td-MAPPO, respectively. On Protoss, 128×1 similarly yields win rates of only 0.00 and 0.04, compared with 0.72 and 0.55 at the defaults. An excessively large $c_{\text{es}} = 50$ also makes Td-MAPPO unstable on Protoss, producing 0.41 ± 0.44. Weak priors reduce LBF performance, with $\lambda_{\text{step}} = 0.01$ yielding 7.60 for Td-QMIX and $\lambda_{\text{step}} = 0.1$ yielding 7.71 for Td-MAPPO. Outside these extremes, performance remains broadly stable. The results support $B_R \in [128 \times 4, 512 \times 4]$ for four-segment tasks and $B_R \in [256 \times 1, 2048 \times 1]$ otherwise, together with $c_{\text{es}} \in [1, 20]$ and $\lambda_{\text{step}} \in [1, 10]$.

6 Conclusion and Future Work

This work introduced TdMARL, a test-driven framework that replaces manual reward engineering with task-level specifications for cooperative MARL. Pass-fail and indicative tests generate trajectory preferences, from which TdMARL learns segment returns and decomposes them into per-step rewards. Tailored sampling, preference evaluation, and a soft uniform prior adapt this process to multi-agent non-stationarity. Theoretical analysis establishes the validity of extending the test-driven paradigm to cooperative MARL under CTDE. Experiments on MPE, SMACv2, and LBF demonstrate effective guidance across dense, fully sparse, and auxiliary-signal settings. Learned rewards induce specified cooperative behaviors and often approach engineered dense rewards.

Two limitations remain. First, the performance gap between Td-QMIX and Td-MAPPO shows that TdMARL depends on the coordination capability of its underlying backbone, thereby constraining how effectively learned rewards translate into coordinated policies. Second, learning efficiency decreases as team size grows, as indicated by unfinished convergence in several 20-vs-20 SMACv2 scenarios under the fixed training budget. Future work will develop a MARL algorithm specifically tailored to the test-driven framework and improve its learning efficiency for larger teams.

AI USE STATEMENT

In this work, we used generative AI tools for grammar correction, occasional text rewriting, and data visualization. We did not use generative AI tools to generate scientific ideas, design experiments, analyze data, implement algorithms, or produce experimental results. The remaining required disclosure tasks were not applicable to this work. We also occasionally used generative AI tools to support bibliographic searches. All AI-assisted material was carefully reviewed and revised by the authors. We verified suggested references against their original sources and checked AI-assisted visualizations against the underlying experimental data. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The complete TdMARL training procedure is provided in Algorithm 1 and Appendix D, with source code and experiment configurations available at https://anonymous.4open.science/r/TdMARL-4FBB. The assumptions and complete proofs underlying the theoretical results are given in Appendix C. Appendix G.1 specifies the environments, reward settings, and task-level tests, while Appendix G.2 reports the model configurations, hyperparameters, and evaluation protocol. The Supplementary Material also includes videos corresponding to the screenshots, along with videos of the main experiments. Additional training curves, numerical results, ablations, and sensitivity analyses are provided in Appendices H, I, and J. Together, these materials document the implementation and evaluation procedures required to reproduce the reported results.

REFERENCES

Omar Adalat and Francesco Belardinelli. Expressive temporal specifications for reward monitoring. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 19533–19541, 2026.

Gaon An, Junhyeok Lee, Xingdong Zuo, Norio Kosaka, Kyung-Min Kim, and Hyun Oh Song. Direct preference-based policy optimization without reward modeling. *Advances in Neural Information Processing Systems*, 36:70247–70266, 2023.

Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4): 819–840, 2002.

Serena Booth, W Bradley Knox, Julie Shah, Scott Niekum, Peter Stone, and Alessandro Allievi. The perils of trial-and-error reward design: misdesign through overfitting and invalid task specifications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 5920–5929, 2023.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

The Viet Bui, Tien Mai, and Hong Thanh Nguyen. O-mapl: Offline multi-agent preference learning. *arXiv preprint arXiv:2501.18944*, 2025.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Sam Devlin and Daniel Kudenko. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *Tenth international conference on autonomous agents and multi-agent systems*, pp. 225–232. ACM, 2011.

Yali Du, Lei Han, Meng Fang, Ji Liu, Tianhong Dai, and Dacheng Tao. Liir: Learning individual intrinsic reward in multi-agent reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *Advances in neural information processing systems*, 36:37567–37593, 2023.

Xuening Feng, Zhaohui Jiang, Timo Kaufmann, Puchen Xu, Eyke Hüllermeier, Paul Weng, and Yifei Zhu. Duo: diverse, uncertain, on-policy query generation and selection for reinforcement learning from human feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 16604–16612, 2025.

Lukas Fluri, Leon Lang, Alessandro Abate, Patrick Forré, David Krueger, and Joar Skalse. The perils of optimizing learned reward functions: Low training error does not guarantee low regret. *arXiv preprint arXiv:2406.15753*, 2024.

Jifeng Hu, Yanchao Sun, Hechang Chen, Sili Huang, Yi Chang, Lichao Sun, et al. Distributional reward estimation for effective multi-agent deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:12619–12632, 2022.

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. *Advances in neural information processing systems*, 31, 2018.

Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pp. 2107–2116. PMLR, 2018.

Rodrigo Toro Icarte, Toryn Q Klassen, Richard Valenzano, and Sheila A McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73:173–208, 2022.

Jeewon Jeon, Woojun Kim, Whiyoung Jung, and Youngchul Sung. Maser: Multi-agent reinforcement learning with subgoals generated from experience replay buffer. In *International conference on machine learning*, pp. 10041–10052. PMLR, 2022.

Yonghyeon Jo, Sunwoo Lee, Junghyuk Yeom, and Seungyul Han. Fox: Formation-aware exploration in multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 12985–12994, 2024.

Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *arXiv preprint arXiv:2106.05091*, 2021.

Jiahui Li, Kun Kuang, Baoxiang Wang, Xingchen Li, Fei Wu, Jun Xiao, and Long Chen. Two heads are better than one: A simple exploration framework for efficient multi-agent reinforcement learning. *Advances in neural information processing systems*, 36:20038–20053, 2023a.

Wei Li, Weiyan Liu, Shitong Shao, and Shiyi Huang. Aiir-mix: Multi-agent reinforcement learning meets attention individual intrinsic reward mixing network. In *Asian Conference on Machine Learning*, pp. 579–594. PMLR, 2023b.

Xinran Li, Zifan Liu, Shibo Chen, and Jun Zhang. Individual contributions as intrinsic exploration scaffolds for multi-agent reinforcement learning. *arXiv preprint arXiv:2405.18110*, 2024.

Boyin Liu, Zhiqiang Pu, Yi Pan, Jianqiang Yi, Yanyan Liang, and Du Zhang. Lazy agents: A new perspective on solving sparse reward problem in multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 21937–21950. PMLR, 2023.

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.

Hao Ma, Shijie Wang, Zhiqiang Pu, Siyao Zhao, and Xiaolin Ai. Vision-based generic potential function for policy alignment in multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19287–19295, 2025.

Zixian Ma, Rose Wang, Fei-Fei Li, Michael Bernstein, and Ranjay Krishna. Elign: Expectation alignment as a multi-agent intrinsic reward. *Advances in Neural Information Processing Systems*, 35:8304–8317, 2022.

Somdeb Majumdar, Shauharda Khadka, Santiago Miret, Stephen McAleer, and Kagan Tumer. Evolutionary reinforcement learning for sample-efficient multiagent coordination. In *International Conference on Machine Learning*, pp. 6651–6660. PMLR, 2020.

Ni Mu, Yao Luan, Yiqin Yang, and Qing-Shan Jia. Multi-Agent Reinforcement Learning via Agent-Specific Preference. *IEEE Transactions on Automation Science and Engineering*, 2026.

Hyungho Na and Il-Chul Moon. Lagma: Latent goal-guided multi-agent reinforcement learning. *arXiv preprint arXiv:2405.19998*, 2024.

Cyrus Neary, Zhe Xu, Bo Wu, and Ufuk Topcu. Reward machines for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2007.01962*, 2020.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, pp. 278–287, 1999.

Pierriccardo Olivieri, Fausto Lasca, Alessandro Gianola, and Matteo Papini. Do it for her: First-order temporal logic reward specification in reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 24613–24621, 2026.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021. URL http://arxiv.org/abs/2006.07869.

Yun Qu, Yuhang Jiang, Boyuan Wang, Yixiu Mao, Cheems Wang, Chang Liu, and Xiangyang Ji. Latent reward: Llm-empowered credit assignment in episodic reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pp. 20095–20103, 2025.

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. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

Aadirupa Saha, Aldo Pacchiano, and Jonathan Lee. Dueling rl: Reinforcement learning with trajectory preferences. In *International conference on artificial intelligence and statistics*, pp. 6263–6289. PMLR, 2023.

Utsav Singh, Wesley A Suttle, Brian M Sadler, Vinay P Namboodiri, and Amrit Singh Bedi. Piper: Primitive-informed preference-based hierarchical reinforcement learning via hindsight relabeling. *arXiv preprint arXiv:2404.13423*, 2024.

Giovanni Varricchione, Natasha Alechina, Mehdi Dastani, and Brian Logan. Synthesising reward machines for cooperative multi-agent reinforcement learning. *Journal of Artificial Intelligence Research*, 85, 2026.

Li Wang, Yupeng Zhang, Yujing Hu, Weixun Wang, Chongjie Zhang, Yang Gao, Jianye Hao, Tangjie Lv, and Changjie Fan. Individual reward assisted multi-agent reinforcement learning. In *International conference on machine learning*, pp. 23417–23432. PMLR, 2022.

Ziyan Wang, Zhicheng Zhang, Fei Fang, and Yali Du. M3hf: Multi-agent reinforcement learning from multi-phase human feedback of mixed quality. *arXiv preprint arXiv:2503.02077*, 2025.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624, 2022.

Zhao Yu, Xiuping Wu, and Liangjun Ke. Test-driven reinforcement learning in continuous control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 27996–28004, 2026.

Ningyuan Zhang, Wenliang Liu, and Calin Belta. Distributed control using reinforcement learning with temporal-logic-based reward shaping. In *Learning for Dynamics and Control Conference*, pp. 751–762. PMLR, 2022.

Lulu Zheng, Jiarui Chen, Jianhao Wang, Jiamin He, Yujing Hu, Yingfeng Chen, Changjie Fan, Yang Gao, and Chongjie Zhang. Episodic multi-agent reinforcement learning with curiosity-driven exploration. *Advances in Neural Information Processing Systems*, 34:3757–3769, 2021.

Chenyang Zhu, Wen Si, Jinyu Zhu, and Zhihao Jiang. Decomposing temporal equilibrium strategy for coordinated distributed multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17618–17627, 2024.

Yiwen Zhu, Jinyi Liu, Pengjie Gu, Yifu Yuan, Zhenxing Ge, Wenya Wei, Zhou Fang, Yujing Hu, and Bo An. Improving reward models with proximal policy exploration for preference-based reinforcement learning. *Advances in Neural Information Processing Systems*, 38:108938–108971, 2026.

Test-Driven Reward Design for Cooperative Multi-Agent Reinforcement Learning (Appendix)

CONTENTS

A Conceptual Illustration of TdMARL **15**

B Detailed Related Works on Reward Design **15**

B.1 Reward Design in Cooperative MARL 15

B.2 Task Specification and Test-Driven Reinforcement Learning 16

B.3 Preference- and Feedback-Based Reward Learning 16

C Additional Theoretical Analysis **17**

C.1 Proofs for Test-to-Policy Alignment and MAPPO Compatibility 17

C.2 Theoretical Analysis of Td-QMIX 19

D TdMARL Training Algorithm **20**

E MARL Optimization with Test-Derived Rewards **20**

E.1 Optimization in Td-QMIX 21

E.2 Optimization in Td-MAPPO 21

F Validation of Return-to-Reward Decomposition **22**

G Experimental Details **22**

G.1 Environment Configurations 22

G.1.1 Level-Based Foraging 22

G.1.2 Multi-Agent Particle Environment 24

G.1.3 StarCraft Multi-Agent Challenge v2 25

G.2 Implementation Details 27

H Additional Experimental Results **28**

H.1 Training Curves Measured by Environment Rewards 28

H.2 MPE Performance and Behavioral Analysis 30

H.3 Detailed SMACv2 5-vs-5 Results 32

H.4 SMACv2 Results with Larger Teams 32

H.5 LBF Performance and Behavioral Analysis 33

I Additional Ablation Results **34**

J Hyperparameter Sensitivity Analysis **37**

A Conceptual Illustration of TdMARL

Figure 5 contrasts the reward-design paradigms underlying TdMARL. Dense rewards can provide frequent guidance, but require designers to select useful behaviors, assign credit across timesteps, construct shaping terms, and tune their weights. Sparse rewards directly express task success, yet offer little feedback before completion. TdMARL combines their complementary advantages. The designer defines a task-aligned tester once, after which joint trajectories are evaluated automatically. The resulting test outcomes induce trajectory preferences for learning segment-level returns, which are subsequently decomposed into informative per-step rewards. TdMARL thereby preserves the simplicity and fidelity of task-level specification while avoiding the manual construction of dense reward terms.

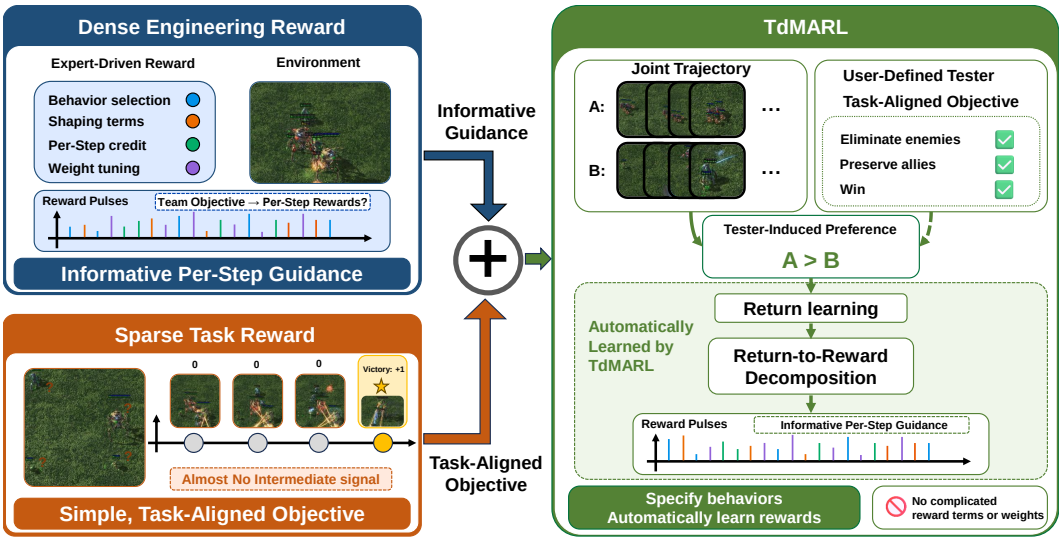

Figure 5: Conceptual illustration of TdMARL. Dense reward engineering provides informative per-step guidance but requires expert design, whereas sparse task rewards are simple and task-aligned but offer little intermediate feedback. A user-defined tester automatically evaluates joint trajectories, from which TdMARL learns task-aligned returns and informative per-step rewards.

B Detailed Related Works on Reward Design

This section discusses research related to TdMARL from three perspectives: reward design in cooperative MARL, structured task specification, and preference- and feedback-based reward learning.

B.1 Reward Design in Cooperative MARL

Dense rewards provide frequent learning guidance, but their construction often requires domain experts to identify intermediate behaviors and tune multiple shaping terms. Potential-based reward shaping preserves optimal policies under suitable conditions (Ng et al., 1999). Its extension to multi-agent stochastic games also preserves Nash equilibria, although altered exploration may still affect the joint policy reached in practice (Devlin & Kudenko, 2011). When environments provide sparse team rewards together with dense individual signals, directly combining them can misalign individual learning with team objectives. MERL separates evolutionary team optimization from gradient-based individual skill learning (Majumdar et al., 2020), while IRAT learns separate policies from team and individual rewards and constrains their discrepancy (Wang et al., 2022).

A broad line of work introduces intrinsic rewards to improve exploration and credit assignment. LIIR learns agent-specific intrinsic rewards whose parameters are optimized using the environmental team return, aligning individual learning incentives with the original cooperative objective (Du

et al., 2019). Other methods use curiosity and episodic memory (Zheng et al., 2021) or reward expectation alignment (Ma et al., 2022). For sparse-reward exploration, MASER selects replay-buffer subgoals using individual and joint Q-values and provides goal-directed intrinsic rewards (Jeon et al., 2022). LAIES uses a learned external-state transition model and counterfactual reasoning to reward individual and collaborative influence on external states, discouraging lazy behavior (Liu et al., 2023). FoX combines formation-visitation bonuses with intrinsic rewards that encourage formation awareness under partial observability (Jo et al., 2024). Further approaches combine curiosity with inter-agent influence (Li et al., 2023a), dynamically mix intrinsic and extrinsic rewards (Li et al., 2023b), construct exploration scaffolds from individual contributions (Li et al., 2024), or generate latent goal-reaching paths (Na & Moon, 2024).

Related studies improve the quality or source of reward guidance. DRE-MARL models reward distributions to increase robustness under uncertain feedback (Hu et al., 2022). Recent methods use vision-language models as potential functions (Ma et al., 2025), or employ language models to generate multidimensional latent rewards for temporal and multi-agent credit assignment (Qu et al., 2025). These studies improve exploration, credit allocation, or semantic guidance through auxiliary incentives and richer supervision. TdMARL instead begins with task-level behavioral requirements and learns optimization rewards from their evaluations.

B.2 Task Specification and Test-Driven Reinforcement Learning

Reward machines represent reward functions as finite-state machines, exposing temporal task structure for decomposition, counterfactual learning, and automated shaping (Icarte et al., 2018; 2022). Their multi-agent extensions decompose team-level reward machines into local machines for decentralized learning (Neary et al., 2020). Reward machines can also be synthesized from temporal-logic specifications and environment abstractions (Varricchione et al., 2026), while temporal equilibrium strategies can be encoded as individual reward machines for coordinated learning (Zhu et al., 2024). These methods provide explicit task semantics and, under suitable conditions, formal guarantees. However, these approaches typically rely on formal task specifications, event-labeling interfaces, or symbolic environment abstractions.

Temporal logic provides a related mechanism for translating structured objectives into rewards. Temporal-logic formulas and automata have been used to synthesize shaped rewards for distributed control (Zhang et al., 2022). Recent methods construct dense reward monitors from quantitative temporal logic (Adalat & Belardinelli, 2026), or use first-order temporal specifications to represent non-Markovian rewards over heterogeneous domains (Olivieri et al., 2026). Although these approaches reduce low-level reward engineering, they still require formal task specifications.

TdRL offers a different interface (Yu et al., 2026). It represents task objectives using pass-fail tests that identify satisfactory trajectories and indicative tests that quantify task-relevant progress. Test outcomes induce trajectory comparisons from which a return function and transition-level rewards are learned. This interface uses executable trajectory tests rather than explicit reward machines or temporal-logic formulas. TdMARL extends the principle to cooperative joint trajectories, introducing tailored sampling and refined preference evaluation for evolving joint-policy distributions, together with a soft uniform prior for stable temporal reward decomposition.

B.3 Preference- and Feedback-Based Reward Learning

Preference-based RL learns from comparisons rather than absolute numerical rewards. Early methods learn reward functions from human comparisons over trajectory segments (Christiano et al., 2017), sometimes together with demonstrations (Ibarz et al., 2018). Later work improves feedback efficiency through unsupervised pretraining, active querying, off-policy learning, and replay relabeling (Lee et al., 2021). Other studies establish theoretical formulations for trajectory preferences (Saha et al., 2023), directly optimize policies without reward modeling (An et al., 2023), or integrate preference learning with hierarchical RL (Singh et al., 2024). Although many methods rely on supplied preference labels, PIPER generates feedback automatically from sparse environment rewards, illustrating that preference supervision need not be exclusively human-provided.

A further challenge is that low reward-prediction error under the data distribution need not imply low policy regret after optimization shifts the behavior distribution (Fluri et al., 2024). Recent methods

improve preference-buffer coverage through policy-proximal exploration (Zhu et al., 2026), or select queries according to diversity, uncertainty, and the current policy distribution (Feng et al., 2025). This issue is particularly relevant to MARL because multiple evolving policies continually change the joint trajectory distribution.

Within cooperative MARL, O-MAPL uses value decomposition to learn offline policies from trajectory preferences without an explicit reward model; its preference datasets can be generated by rule-based or language-model evaluators (Bui et al., 2025). M^3HF incorporates multi-phase human feedback into policy training, using language models to interpret feedback and update rewards through predefined templates and adaptive weights (Wang et al., 2025). These methods respectively emphasize offline policy learning and iterative feedback-based reward refinement. MAGPIE learns agent-specific rewards from online preferences and aggregates them into a team reward, with feedback supplied by automated teachers or other evaluators (Mu et al., 2026). TdMARL instead constructs hierarchical joint-trajectory preferences from team-level pass-fail and indicative tests. Building on TdRL's return-to-reward learning pipeline, it introduces targeted sampling, refined preference evaluation, and a soft uniform prior to improve preference supervision and temporal reward allocation as joint policies evolve.

C Additional Theoretical Analysis

Unlike the maximum-entropy analysis of TdRL (Yu et al., 2026), our results are formulated in terms of return-decomposition and policy-optimization errors and do not require a particular trajectory-distribution update.

C.1 Proofs for Test-to-Policy Alignment and MAPPO Compatibility

Proof of Proposition 1.

Proof. Let $C = \sup_{\boldsymbol{\tau}\in\mathcal{T}_\Theta} \widehat{R}_\psi(\boldsymbol{\tau})$. Because $\boldsymbol{\pi}^+$ generates satisfactory trajectories with return C almost surely, Δ-test separation implies that $\widehat{R}_\psi(\boldsymbol{\tau}) \le C - \Delta$ for every $\boldsymbol{\tau} \in \mathcal{T}_\Theta \setminus \mathcal{T}^+$. Therefore, for any $\boldsymbol{\pi} \in \Pi_\Theta$, we obtain:

$$\begin{aligned} C - \mathbb{E}_{P_{\boldsymbol{\pi}}}[\widehat{R}_\psi(\boldsymbol{\tau})] &= \mathbb{E}_{P_{\boldsymbol{\pi}}}[C - \widehat{R}_\psi(\boldsymbol{\tau})] \\ &\ge \Delta P_{\boldsymbol{\pi}}(\boldsymbol{\tau} \notin \mathcal{T}^+). \end{aligned} \tag{15}$$

By the definition of $\varepsilon_{\boldsymbol{\pi}}$ and Jensen's inequality, the difference between the expected returns is bounded as follows:

$$\left|\mathbb{E}_{P_{\boldsymbol{\pi}}}[\widehat{G}_\phi(\boldsymbol{\tau})] - \mathbb{E}_{P_{\boldsymbol{\pi}}}[\widehat{R}_\psi(\boldsymbol{\tau})]\right| \le \varepsilon_{\boldsymbol{\pi}}. \tag{16}$$

Moreover, the δ-optimality of $\widehat{\boldsymbol{\pi}}$ gives:

$$\widehat{J}_\phi(\widehat{\boldsymbol{\pi}}) \ge \widehat{J}_\phi(\boldsymbol{\pi}^+) - \delta. \tag{17}$$

Applying Eq. (16) to both policies, using δ-optimality, and noting that $\mathbb{E}_{P_{\boldsymbol{\pi}^+}}[\widehat{R}_\psi(\boldsymbol{\tau})] = C$, we obtain:

$$\begin{aligned} \mathbb{E}_{P_{\widehat{\boldsymbol{\pi}}}}[\widehat{R}_\psi(\boldsymbol{\tau})] &\ge \mathbb{E}_{P_{\widehat{\boldsymbol{\pi}}}}[\widehat{G}_\phi(\boldsymbol{\tau})] - \varepsilon_{\widehat{\boldsymbol{\pi}}} \\ &\ge \mathbb{E}_{P_{\boldsymbol{\pi}^+}}[\widehat{G}_\phi(\boldsymbol{\tau})] - \delta - \varepsilon_{\widehat{\boldsymbol{\pi}}} \\ &\ge C - \varepsilon_{\boldsymbol{\pi}^+} - \varepsilon_{\widehat{\boldsymbol{\pi}}} - \delta. \end{aligned} \tag{18}$$

Finally, combining this inequality with Eq. (15) gives:

$$P_{\widehat{\boldsymbol{\pi}}}(\boldsymbol{\tau} \notin \mathcal{T}^+) \le \min\left\{1, \frac{\varepsilon_{\widehat{\boldsymbol{\pi}}} + \varepsilon_{\boldsymbol{\pi}^+} + \delta}{\Delta}\right\}, \tag{19}$$

which proves the proposition. □

The discounted sum-matching term $\mathcal{L}_{\text{sum}}$ controls the decomposition error in Proposition 1. By the Cauchy–Schwarz inequality:

$$\varepsilon_{\boldsymbol{\pi}} \leq \sqrt{\mathbb{E}_{P_{\boldsymbol{\pi}}}\left[\left(\widehat{G}_\phi(\boldsymbol{\tau}) - \widehat{R}_\psi(\boldsymbol{\tau})\right)^2\right]}. \tag{20}$$

To account for the entropy regularization used by MAPPO, consider the following objective:

$$\widehat{J}_{\phi,\beta}(\boldsymbol{\pi}) = \widehat{J}_\phi(\boldsymbol{\pi}) + \beta\mathcal{H}(\boldsymbol{\pi}), \tag{21}$$

where $\beta \geq 0$. We assume that the entropy functional has finite range:

$$B_{\mathcal{H}} = \sup_{\boldsymbol{\pi},\boldsymbol{\pi}' \in \boldsymbol{\Pi}_\Theta} \left|\mathcal{H}(\boldsymbol{\pi}) - \mathcal{H}(\boldsymbol{\pi}')\right|. \tag{22}$$

If $\widehat{\boldsymbol{\pi}}_\beta$ is δ-optimal for this objective, its expected decomposed return satisfies:

$$\mathbb{E}_{P_{\widehat{\boldsymbol{\pi}}_\beta}}[\widehat{G}_\phi(\boldsymbol{\tau})] \geq \mathbb{E}_{P_{\boldsymbol{\pi}^+}}[\widehat{G}_\phi(\boldsymbol{\tau})] - \delta - \beta B_{\mathcal{H}}. \tag{23}$$

Substituting this inequality into the preceding argument yields the entropy-regularized bound:

$$P_{\widehat{\boldsymbol{\pi}}_\beta}(\boldsymbol{\tau} \notin \mathcal{T}^+) \leq \min\left\{1, \frac{\varepsilon_{\widehat{\boldsymbol{\pi}}_\beta} + \varepsilon_{\boldsymbol{\pi}^+} + \delta + \beta B_{\mathcal{H}}}{\Delta}\right\}. \tag{24}$$

Thus, entropy regularization contributes an explicit error term without changing the underlying test-to-policy argument.

Proof of Proposition 2.

Proof. Under the factorized joint policy, the probability of a joint trajectory can be expressed as follows:

$$P_{\boldsymbol{\theta}}(\boldsymbol{\tau}) = p_{\text{env}}(\boldsymbol{\tau}) \prod_{t=0}^{T-1} \prod_{j=1}^{n} \pi_{\theta_j}(a_t^j \mid \tau_t^j), \tag{25}$$

where $p_{\text{env}}(\boldsymbol{\tau})$ contains all policy-independent initial-state, transition, and observation terms. Consequently, differentiating its log-probability with respect to θ_i gives:

$$\nabla_{\theta_i} \log P_{\boldsymbol{\theta}}(\boldsymbol{\tau}) = \sum_{t=0}^{T-1} \nabla_{\theta_i} \log \pi_{\theta_i}(a_t^i \mid \tau_t^i). \tag{26}$$

Applying the likelihood-ratio identity to $\widehat{J}_\phi = \mathbb{E}_{\boldsymbol{\pi}}[\widehat{G}_\phi(\boldsymbol{\tau})]$ and removing rewards preceding each action by causality gives:

$$\nabla_{\theta_i} \widehat{J}_\phi = \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t=0}^{T-1} \gamma^t \nabla_{\theta_i} \log \pi_{\theta_i}(a_t^i \mid \tau_t^i) Q^{\boldsymbol{\pi}}_{\phi,t}\right]. \tag{27}$$

Here, $Q^{\boldsymbol{\pi}}_{\phi,t}$ is the centralized action-value function induced by r_ϕ. Subtracting a centralized baseline independent of a_t^i leaves the expectation unchanged. Choosing the corresponding centralized value function as the baseline therefore gives:

$$\nabla_{\theta_i} \widehat{J}_\phi = \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t=0}^{T-1} \gamma^t \nabla_{\theta_i} \log \pi_{\theta_i}(a_t^i \mid \tau_t^i) A^{\boldsymbol{\pi}}_{\phi,t}\right]. \tag{28}$$

For the MAPPO surrogate, define the agent-wise likelihood ratio as follows:

$$\rho_t^i(\theta_i) = \frac{\pi_{\theta_i}(a_t^i \mid \tau_t^i)}{\pi_{\theta_i^{\text{old}}}(a_t^i \mid \tau_t^i)}. \tag{29}$$

At the behavior policy, $\theta_i = \theta_i^{\text{old}}$ and $\rho_t^i = 1$. Because this value lies inside the clipping interval, the clipped and unclipped surrogate terms have the same local derivative. With exact advantages and consistent discounted time weighting, we obtain:

$$\nabla_{\theta_i} \mathcal{L}_{\text{clip}}|_{\theta_i=\theta_i^{\text{old}}} = \nabla_{\theta_i} \widehat{J}_\phi. \tag{30}$$

This establishes first-order compatibility. When actor parameters are shared, the corresponding gradient is obtained by summing Eq. (28) over agents. □

This result applies to the gradient at the behavior policy and does not assume that clipped updates monotonically improve the learned objective. Errors caused by finite samples, estimated advantages, and multiple optimization epochs are absorbed into the optimization error δ in Proposition 1.

C.2 Theoretical Analysis of Td-QMIX

Unlike MAPPO, QMIX restricts the representable joint action-value functions through monotonic mixing (Rashid et al., 2020). A test-derived reward alone does not ensure that its optimal value function satisfies this structural requirement. We therefore establish the Td-QMIX guarantee under approximate monotonic representability.

Proposition 3. *Fix a bounded transition reward r_ϕ and let $Q_\phi^*(\xi, \mathbf{a})$ denote its optimal centralized joint action-value function, where ξ represents centralized training information. Suppose that QMIX represents the joint value function as follows:*

$$Q_\omega(\xi, \mathbf{a}) = f_\omega \left(Q_1(\tau^1, a^1), \ldots, Q_n(\tau^n, a^n), \xi\right). \tag{31}$$

Assume that f_ω is nondecreasing in every individual utility and that Q_ω approximates Q_ϕ^* with some error $\eta \geq 0$:

$$\left\|Q_\omega - Q_\phi^*\right\|_\infty \leq \eta. \tag{32}$$

Define the decentralized greedy action of each agent as follows:

$$\widehat{a}^i \in \arg\max_{a^i} Q_i(\tau^i, a^i). \tag{33}$$

Let the corresponding centralized optimal action be:

$$\mathbf{a}^* \in \arg\max_{\mathbf{a}} Q_\phi^*(\xi, \mathbf{a}). \tag{34}$$

Then the suboptimality of $\widehat{\mathbf{a}} = (\widehat{a}^1, \ldots, \widehat{a}^n)$ is bounded as follows:

$$Q_\phi^*(\xi, \mathbf{a}^*) - Q_\phi^*(\xi, \widehat{\mathbf{a}}) \leq 2\eta. \tag{35}$$

Proof. Because each $\widehat{a}^i$ maximizes its individual utility and f_ω is nondecreasing in every utility, the joint action $\widehat{\mathbf{a}}$ maximizes Q_ω. Using this property and the uniform approximation bound, the centralized suboptimality can be decomposed as follows:

$$\begin{aligned} Q_\phi^*(\xi, \mathbf{a}^*) - Q_\phi^*(\xi, \widehat{\mathbf{a}}) &= \left[Q_\phi^*(\xi, \mathbf{a}^*) - Q_\omega(\xi, \mathbf{a}^*)\right] \\ &\quad + \left[Q_\omega(\xi, \mathbf{a}^*) - Q_\omega(\xi, \widehat{\mathbf{a}})\right] \\ &\quad + \left[Q_\omega(\xi, \widehat{\mathbf{a}}) - Q_\phi^*(\xi, \widehat{\mathbf{a}})\right] \\ &\leq \eta + 0 + \eta = 2\eta. \end{aligned} \tag{36}$$

This proves the proposition. □

When $\eta = 0$, decentralized greedy execution exactly maximizes Q_ϕ^*. More generally, if Q_ϕ^* has a unique maximizer with an action gap greater than 2η, then $\widehat{\mathbf{a}} = \mathbf{a}^*$. This guarantee depends on approximate monotonic representability, which is the standard structural requirement of QMIX and is not ensured by the source of the learned reward.

Algorithm 1 TdMARL

Require: Pass-fail tests $\mathcal{Z}_{\mathrm{pf}}$ and indicative tests $\mathcal{Z}_{\mathrm{ind}}$
Require: MARL backbone Alg $\in \{\mathrm{QMIX}, \mathrm{MAPPO}\}$
Initialize Alg learner, return model f_ψ, and reward model r_ϕ
Initialize MARL buffer $\mathcal{B}_M \leftarrow \emptyset$ and return buffer $\mathcal{B}_R \leftarrow \emptyset$
while $t_{\mathrm{env}} < t_{\max}$ **do**
  Collect a joint episode batch $\mathcal{E}$ using the current policies
  Relabel the rewards in $\mathcal{E}$ using r_ϕ
  Store $\mathcal{E}$ in $\mathcal{B}_M$
  Split $\mathcal{E}$ into trajectory segments
  for all segment $\boldsymbol{\tau}$ **do**
    Compute $\mathbf{z}_{\mathrm{pf}}(\boldsymbol{\tau})$ and $\mathbf{z}_{\mathrm{ind}}(\boldsymbol{\tau})$
    Store the segment and its test outcomes in $\mathcal{B}_R$
  end for
  if $\mathcal{B}_R$ can provide a minibatch and a test-driven update is due **then**
    Update test statistics from $\mathcal{B}_R$
    for each return-model gradient step **do**
      Sample trajectory pairs using the mixed sampling strategy
      Construct labels using the TdMARL preference evaluator
      Optimize $\mathcal{L}_{\mathrm{RT}}$ with respect to ψ
    end for
    Compute segment-return targets using f_ψ
    for each reward-model gradient step **do**
      Sample trajectory segments from $\mathcal{B}_R$
      Optimize $\mathcal{L}_{\mathrm{RW}}$ with respect to ϕ
    end for
    Relabel the rewards in $\mathcal{B}_M$ using r_ϕ
  end if
  if $\mathcal{B}_M$ can provide a training batch **then**
    Retrieve a batch from $\mathcal{B}_M$
    Update Alg using the relabeled rewards
    if Alg $=$ MAPPO **then**
      Clear $\mathcal{B}_M$
    end if
  end if
end while

D TdMARL Training Algorithm

Algorithm 1 summarizes the complete training procedure. The MARL buffer $\mathcal{B}_M$ denotes replay storage for Td-QMIX and the current on-policy rollout storage for Td-MAPPO. Discounting in reward decomposition is indexed relative to the beginning of each segment. The reward outputs raw per-step rewards, while discounting is applied only when computing $\mathcal{L}_{\mathrm{sum}}$ and by the underlying MARL learner.

Table 1 reports estimated training wall-clock time. Relative to the original backbones, TdMARL increases runtime by 35.4–52.7% on MPE and 32.7–63.3% on SMACv2, while approximately doubling it on LBF. Because the backbones train relatively quickly, these increases leave TdMARL within a practical wall-clock range for QMIX- and MAPPO-based MARL. The overhead comes from trajectory testing, return and reward learning, and reward relabeling. These operations are confined to centralized training, leaving decentralized execution unchanged.

E MARL Optimization with Test-Derived Rewards

Our Td-QMIX implementation builds on the QMIX implementation in EPyMARL (Papoudakis et al., 2021)[2]. Our Td-MAPPO implementation follows the MAPPO algorithm of Yu et al. (2022) and its official implementation [3]. We port MAPPO into EPyMARL to provide both backbones under a unified training interface.

[2] https://github.com/uoe-agents/epymarl
[3] https://github.com/marlbenchmark/on-policy

Table 1: Estimated training wall-clock time and relative overhead on a server with two AMD EPYC 9654 CPUs and eight RTX 4090 GPUs.

Backbone	Original (h)	TdMARL (h)	Increase
Cooperative Navigation			
QMIX	8.57 ± 0.07	12.69 ± 0.08	$+48.1\%$
MAPPO	9.10 ± 0.01	13.33 ± 0.06	$+46.6\%$
Predator–Prey			
QMIX	14.06 ± 0.03	19.03 ± 0.13	$+35.4\%$
MAPPO	12.82 ± 0.03	19.57 ± 0.02	$+52.7\%$
LBF			
QMIX	5.14 ± 0.01	10.88 ± 0.18	$+111.7\%$
MAPPO	5.80 ± 0.02	11.83 ± 0.04	$+103.9\%$
Protoss 5-*vs*-5			
QMIX	17.26 ± 3.00	23.49 ± 1.26	$+36.1\%$
MAPPO	17.19 ± 2.02	22.81 ± 2.68	$+32.7\%$
Terran 5-*vs*-5			
QMIX	18.53 ± 1.67	28.51 ± 2.76	$+53.8\%$
MAPPO	19.41 ± 2.57	26.90 ± 2.48	$+38.6\%$
Zerg 5-*vs*-5			
QMIX	18.57 ± 0.52	28.92 ± 4.11	$+55.7\%$
MAPPO	16.43 ± 2.60	26.82 ± 2.74	$+63.3\%$

After reward decomposition, the predicted per-step reward is $\bar{r}_{\phi,t} = r_\phi(s_t, \mathbf{a}_t)$. Only $\bar{r}_{\phi,t}$ is stored in the MARL training data. The underlying learner uses the same discount factor γ as reward decomposition.

E.1 Optimization in Td-QMIX

For Td-QMIX, $\mathcal{B}_M$ is an off-policy replay buffer whose rewards are relabeled using $\bar{r}_{\phi,t}$. QMIX decomposes the centralized joint action-value function into individual utilities through a monotonic mixing network (Rashid et al., 2020):

$$Q_{\text{tot}}(\boldsymbol{\tau}_t, \mathbf{a}_t, s_t) = f_{\text{mix}}\left(Q_1(\tau_t^1, a_t^1), \ldots, Q_n(\tau_t^n, a_t^n) \mid s_t\right), \qquad \partial_{Q_i} Q_{\text{tot}} \geq 0. \tag{37}$$

Each utility Q_i depends only on the local history and action of agent i, while the mixer uses the global state during centralized training. Monotonicity ensures that individual greedy actions jointly maximize Q_{tot}:

$$a_t^{i,*} \in \arg\max_{a^i} Q_i(\tau_t^i, a^i), \qquad \mathbf{a}_t^* \in \arg\max_{\mathbf{a}} Q_{\text{tot}}(\boldsymbol{\tau}_t, \mathbf{a}, s_t). \tag{38}$$

Given a relabeled transition, the temporal-difference target is:

$$y_t^Q = \bar{r}_{\phi,t} + \gamma(1 - d_t) \max_{\mathbf{a}'} Q_{\text{tot}}^-(\boldsymbol{\tau}_{t+1}, \mathbf{a}', s_{t+1}), \tag{39}$$

where d_t indicates termination and Q_{tot}^- is the target network. The maximizing joint action is obtained through the target individual utilities. The joint value function is optimized using:

$$\mathcal{L}_Q(\omega) = \mathbb{E}_{\mathcal{B}_M}\left[\left(y_t^Q - Q_{\text{tot}}(\boldsymbol{\tau}_t, \mathbf{a}_t, s_t; \omega)\right)^2\right]. \tag{40}$$

Target-network updates follow standard QMIX, while decentralized execution selects actions independently from the learned individual utilities.

E.2 Optimization in Td-MAPPO

For Td-MAPPO, $\mathcal{B}_M$ denotes the latest on-policy rollout batch. Its rewards are relabeled before computing value targets and advantages. Using the centralized critic V_ν, the temporal residual and

generalized advantage estimate are:

$$\begin{aligned} \delta_t^V &= \bar{r}_{\phi,t} + \gamma(1-d_t)V_\nu(s_{t+1}) - V_\nu(s_t), \\ \widehat{A}_t &= \sum_{\ell=0}^{H-t-1} (\gamma\lambda_{\text{GAE}})^\ell \delta_{t+\ell}^V, \end{aligned} \tag{41}$$

where H is the rollout horizon. The critic target is $\widehat{V}_t = \widehat{A}_t + V_\nu(s_t)$, and the critic minimizes:

$$\mathcal{L}_V(\nu) = \mathbb{E}_t\left[\left(V_\nu(s_t) - \widehat{V}_t\right)^2\right]. \tag{42}$$

The decentralized actors maximize:

$$\begin{aligned} \mathcal{J}_{\text{actor}}(\boldsymbol{\theta}) = \mathbb{E}_{t,i}\Big[&\min\left(\rho_t^i \widehat{A}_t, \text{clip}(\rho_t^i, 1-\epsilon, 1+\epsilon)\widehat{A}_t\right) \\ &+ \beta\mathcal{H}\left(\pi_{\theta_i}(\cdot \mid \tau_t^i)\right)\Big]. \end{aligned} \tag{43}$$

The remaining rollout, normalization, and optimization procedures follow standard MAPPO. The return and reward models are used only during centralized training and do not alter decentralized execution.

F Validation of Return-to-Reward Decomposition

To examine whether return decomposition produces behaviorally meaningful per-step rewards, we analyze a representative LBF trajectory generated under the experimental setup described in Appendix G.1.1. The policy is learned by Td-MAPPO. In this trajectory, two level-1 agents must cooperate to collect level-2 fruits.

Figure 6 shows that the decomposed rewards vary across transitions rather than collapsing to a uniform allocation. Figure 7 further relates these rewards to cooperative behavior. At timestep 2, both agents are adjacent to the first fruit and receive a positive reward of 0.0364. Their successful pickup at timestep 3 receives 0.1484, the largest reward among the first 12 timesteps highlighted in red. After a new fruit appears, the repositioning steps from timesteps 4 to 9 receive mostly negative rewards. Once both agents approach the fruit at timestep 10, the reward rises to 0.0540, followed by 0.0886 for the pickup at timestep 11. After collection, a new fruit respawns far from the agents, and the reward falls to nearly zero.

These patterns show that the learned decomposition assigns positive feedback to successful pickups and the coordination immediately preceding them, while distinguishing less useful transitions. The alignment between rewards and cooperative behavior provides qualitative evidence that segment-level evaluations are converted into meaningful per-step guidance for policy optimization.

G Experimental Details

G.1 Environment Configurations

This section describes the environment configurations, original reward functions, task-level tests, and return-buffer settings used in each benchmark.

G.1.1 Level-Based Foraging

Experiments are conducted in the cooperative LBF configuration `Foraging-8x8-2p-1f-coop-v3` (Papoudakis et al., 2021). The environment contains an 8×8 grid, two level-1 agents, and one level-2 fruit. Each agent can remain stationary, move in four directions, or execute the LOAD action. Successful collection requires both adjacent agents to execute LOAD, because neither agent can independently collect the level-2 fruit. The sight range is set to 8, covering the complete grid.

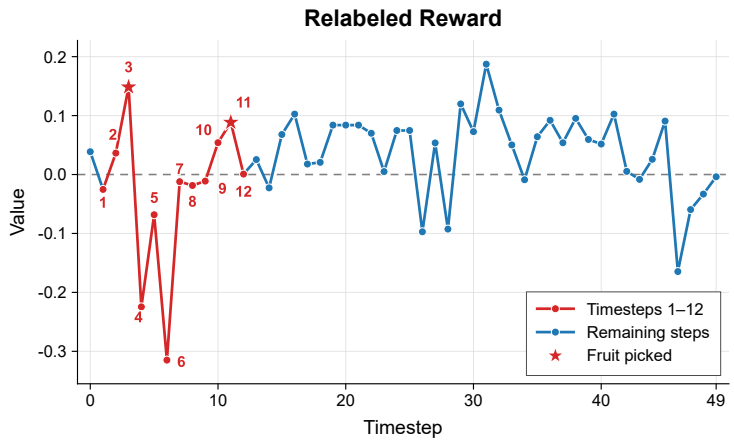

Figure 6: Per-step rewards relabeled by Td-MAPPO for a representative LBF trajectory. The timesteps 1–12 are highlighted, and stars mark successful fruit pickups.

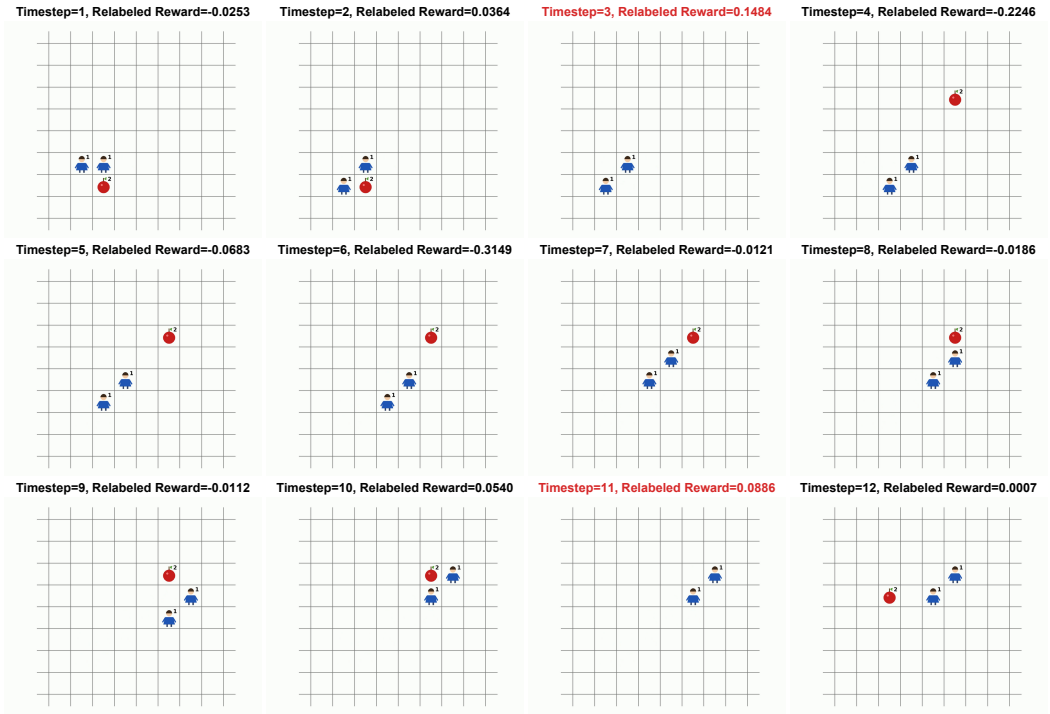

Figure 7: Visualization of 12 LBF timesteps, with fruits picked at timesteps 3 and 11.

The original LBF reward is normalized and assigned only after successful collection. Under this configuration, each participating agent receives 0.5, and the individual rewards are summed into the shared environmental reward:

$$r_t^{\text{env}} = \begin{cases} 1, & \text{if the agents jointly collect the fruit,} \\ 0, & \text{otherwise.} \end{cases} \tag{44}$$

Failed loading attempts receive no penalty. After collection, a new level-2 fruit is spawned at another location and early termination is disabled. Each episode therefore continues for 50 timesteps. For return learning, each episode is divided into four contiguous segments, and the return-buffer capacity

is set to 256, yielding the 256×4 configuration. The environmental reward is sparse because it identifies successful collection without evaluating the preceding coordination.

The `LBFTester` implements the trajectory evaluation described in Section 4.2. Each sampled segment $\boldsymbol{\tau}$ is mapped to four pass-fail outcomes and four indicative outcomes:

$$\begin{aligned} \mathbf{z}_{\mathrm{pf}}(\boldsymbol{\tau}) &= \left(z^1_{\mathrm{pf}}, z^2_{\mathrm{pf}}, z^3_{\mathrm{pf}}, z^4_{\mathrm{pf}}\right), \\ \mathbf{z}_{\mathrm{ind}}(\boldsymbol{\tau}) &= \left(z^1_{\mathrm{ind}}, z^2_{\mathrm{ind}}, z^3_{\mathrm{ind}}, z^4_{\mathrm{ind}}\right). \end{aligned} \tag{45}$$

These outcomes evaluate four stages of cooperative fruit collection:

1. **Agent pairing.** $z^1_{\mathrm{pf}} = 1$ when the Euclidean distance between the agents is at most 2.0 for at least 50% of valid timesteps. z^1_{ind} is their average proximity, computed as the temporal mean of $(1 + d^{\mathrm{aa}}_t)^{-1}$.
2. **Joint fruit coverage.** $z^2_{\mathrm{pf}} = 1$ when both agents are within distance 1.5 of the same fruit for at least 10% of valid timesteps. z^2_{ind} averages $(1 + d^{\mathrm{af}}_t)^{-1}$, where d^{af}_t is the distance between the fruit and the farther agent.
3. **Valid load attempts.** A valid attempt occurs when an agent within distance 1.2 of the fruit executes LOAD. $z^3_{\mathrm{pf}} = 1$ when such attempts occur in at least 10% of valid timesteps, while z^3_{ind} records their temporal frequency.
4. **Successful pickups.** A pickup is detected when the fruit position changes between consecutive observations due to respawning. $z^4_{\mathrm{pf}} = 1$ when pickups occur in at least 15% of valid noninitial timesteps, while z^4_{ind} records the number of successful pickups.

LBF therefore evaluates TdMARL under sparse rewards with auxiliary test constraints. The environmental reward specifies only successful collection. In contrast, $\mathbf{z}_{\mathrm{pf}}$ encodes behavioral requirements over complete segments, while $\mathbf{z}_{\mathrm{ind}}$ measures the corresponding coordination progress.

G.1.2 MULTI-AGENT PARTICLE ENVIRONMENT

The Multi-Agent Particle Environment (MPE) (Lowe et al., 2017) contains continuous two-dimensional cooperative tasks with partial observations. All episodes last at most 80 timesteps, and the controlled agents receive a common team reward. Separately, the tester maps each sampled segment $\boldsymbol{\tau}$ to the pass-fail outcomes $\mathbf{z}_{\mathrm{pf}}$ and indicative outcomes $\mathbf{z}_{\mathrm{ind}}$ used for preference construction. Only valid timesteps are evaluated, and $T_{\boldsymbol{\tau}}$ denotes the valid segment length.

Cooperative Navigation. Three agents must coordinate to cover three landmarks. Each agent has radius 0.15 and maximum speed 1.3. A landmark is covered when an agent lies within a radius of 0.3. For return learning, each 80-timestep episode is divided into four contiguous segments, and the return-buffer capacity is set to 256, yielding the 256×4 configuration. Agents and non-overlapping landmarks are initialized uniformly in $[-2.5, 2.5]^2$.

In dense setting, let $d_{i\ell,t}$ denote the distance from agent i to landmark ℓ, and let $\mathcal{M}_t$ be their greedy balanced assignment. Since the task contains equal numbers of agents and landmarks, each landmark accepts at most one assigned agent. Before reward normalization, the dense team reward is:

$$r^{\mathrm{dense}}_t = -\sum_{(i,\ell)\in\mathcal{M}_t} d_{i\ell,t} + \sum_{\ell=1}^{3} \mathbf{1}\{c_{\ell,t} = 1\} - 2N_{\mathrm{col},t}, \tag{46}$$

where $c_{\ell,t}$ is the number of agents within the coverage radius and $N_{\mathrm{col},t}$ counts non-exempt agent pairs separated by less than 0.375. The reward therefore combines assignment distance, balanced landmark coverage, and collision avoidance.

The tester implements the trajectory evaluation in Section 4.2. Each sampled segment $\boldsymbol{\tau}$ is mapped to two pass-fail outcomes and two indicative outcomes:

$$\begin{aligned} \mathbf{z}_{\mathrm{pf}}(\boldsymbol{\tau}) &= \left(z^1_{\mathrm{pf}}, z^2_{\mathrm{pf}}\right), \\ \mathbf{z}_{\mathrm{ind}}(\boldsymbol{\tau}) &= \left(z^1_{\mathrm{ind}}, z^2_{\mathrm{ind}}\right). \end{aligned} \tag{47}$$

These outcomes evaluate landmark coverage and safe separation:

1. **Landmark coverage.** Agents and landmarks are matched using the same global greedy assignment as the environment reward, with each agent and landmark used at most once. Let $\bar{d}_{\text{start}}$ and $\bar{d}_{\text{end}}$ denote the mean assigned distances at the first and last valid timesteps. The indicative outcome is $z^1_{\text{ind}}(\boldsymbol{\tau}) = \left[\frac{2}{1+\bar{d}_{\text{end}}} - \frac{1}{1+\bar{d}_{\text{start}}}\right]_+$, where $[x]_+ = \max(x, 0)$. It combines the final landmark proximity with the improvement achieved over the segment. A landmark is considered covered when exactly one agent is within distance 0.3. The pass-fail outcome $z^1_{\text{pf}} = 1$ when at least two landmarks are each continuously covered for five or more valid timesteps.
2. **Safe separation.** Two agents collide when their distance is below 2.5 times the agent radius, giving a collision threshold of 0.375. The indicative outcome $z^2_{\text{ind}}(\boldsymbol{\tau}) = T_{\boldsymbol{\tau}}^{-1} \sum_t \mathbf{1}\{\text{no agent collision at } t\}$ measures the fraction of valid timesteps without collisions. The pass-fail outcome $z^2_{\text{pf}} = 1$ only when the segment remains collision-free throughout.

Thus, z^1_{pf} measures persistent landmark coverage and z^2_{pf} enforces collision-free coordination, while z^1_{ind} and z^2_{ind} provide graded measures of the corresponding behaviors.

Predator–Prey. Three controlled predators cooperate to capture one faster rule-based prey in an environment containing two fixed obstacles. Predators have radius 0.12, maximum speed 1.0, and acceleration 5.0. The prey has radius 0.10, maximum speed 1.3, and acceleration 7.0. It follows a refuge-seeking policy that moves toward the field center and rests there when unthreatened. When a predator enters a radius of 1.5, this attraction is combined with inverse-square repulsion from the nearest predator. If at least two predators approach within 0.8, the prey instead moves perpendicular to their connecting line to escape through the pincer gap. A capture occurs when the distance between any predator and the prey is below 0.22. Each episode is retained as a single segment for return learning, with a return-buffer capacity of 512, yielding the 512×1 configuration.

In dense setting, let $d_{\min,t}$ denote the minimum predator–prey distance and let $I_{\text{cap},t}$ indicate whether a capture occurs. Before common-reward aggregation and normalization, each controlled predator receives:

$$r_t^{\text{dense}} = 10 I_{\text{cap},t} - 0.025 d_{\min,t}. \tag{48}$$

The tester implements the trajectory evaluation in Section 4.2. Each sampled segment $\boldsymbol{\tau}$ is mapped to two pass-fail outcomes and two indicative outcomes:

$$\begin{aligned} \mathbf{z}_{\text{pf}}(\boldsymbol{\tau}) &= \left(z^1_{\text{pf}}, z^2_{\text{pf}}\right), \\ \mathbf{z}_{\text{ind}}(\boldsymbol{\tau}) &= \left(z^1_{\text{ind}}, z^2_{\text{ind}}\right). \end{aligned} \tag{49}$$

Let $d_{\min,t} = \min_i \|\mathbf{p}_t^i - \mathbf{p}_t^{\text{prey}}\|_2$ be the distance between the prey and its nearest predator. A prey-contact timestep occurs when $d_{\min,t} < 0.22$, where 0.22 is the sum of the predator and prey radii. The outcomes evaluate approach quality and repeated capture:

1. **Prey approach.** The indicative outcome $z^1_{\text{ind}}(\boldsymbol{\tau}) = T_{\boldsymbol{\tau}}^{-1} \sum_t \frac{1}{1+d_{\min,t}}$ measures the average proximity between the prey and its nearest predator over valid timesteps. The pass-fail outcome $z^1_{\text{pf}} = 1$ when at least one prey-contact timestep occurs in the segment.
2. **Repeated prey contact.** Let $I_{\text{cap},t} = \mathbf{1}\{d_{\min,t} < 0.22\}$. The indicative outcome $z^2_{\text{ind}}(\boldsymbol{\tau}) = \sum_t I_{\text{cap},t}$ counts prey-contact timesteps over the segment. Each prey-contact timestep contributes one to the count, regardless of whether contact timesteps are consecutive. The corresponding pass-fail outcome is $z^2_{\text{pf}}(\boldsymbol{\tau}) = \mathbf{1}\left\{\sum_t I_{\text{cap},t} \geq 3\right\}$.

Thus, z^1_{pf} identifies whether the predators reach the prey, while z^2_{pf} requires prey contact at three or more timesteps.

G.1.3 StarCraft Multi-Agent Challenge v2

The StarCraft Multi-Agent Challenge v2 (SMACv2) (Ellis et al., 2023) evaluates decentralized micromanagement under randomized team compositions and initial positions. The main experiments

consider three race-specific 5-vs-5 scenarios at difficulty level 7. Each allied unit selects movement, stop, and target-specific attack or healing actions from its local observations, while the global state is available only during centralized training. Team compositions are sampled independently for each episode. Protoss teams contain stalkers, zealots, and colossi with probabilities 0.45, 0.45, and 0.10. Terran teams contain marines, marauders, and medivacs with probabilities 0.45, 0.45, and 0.10. Zerg teams contain zerglings, hydralisks, and banelings with probabilities 0.45, 0.45, and 0.10. Initial formations are randomized between surrounded and reflected layouts. Each battle trajectory is treated as a single segment for return learning, with a return-buffer capacity of 512, yielding the 512×1 configuration.

In **sparse setting**, the principal SMACv2 evaluation uses a fully sparse environment reward. All intermediate combat rewards are removed, and feedback is provided only when the battle terminates:

$$r_t^{\text{sparse}} = \begin{cases} 1, & \text{if the allied team wins at } t, \\ -1, & \text{if the allied team is defeated at } t, \\ 0, & \text{otherwise.} \end{cases} \tag{50}$$

A battle reaching the episode limit without either outcome receives no terminal reward. This setting provides no feedback for dealing damage, eliminating individual enemies, or preserving allied units.

The tester implements the trajectory evaluation in Section 4.2. Unit survival is determined from the health feature in the global state, and only valid timesteps are evaluated. Each battle trajectory $\boldsymbol{\tau}$ is mapped to two pass-fail outcomes and two indicative outcomes:

$$\begin{aligned} \mathbf{z}_{\text{pf}}(\boldsymbol{\tau}) &= \left(z_{\text{pf}}^1, z_{\text{pf}}^2\right), \\ \mathbf{z}_{\text{ind}}(\boldsymbol{\tau}) &= \left(z_{\text{ind}}^1, z_{\text{ind}}^2\right). \end{aligned} \tag{51}$$

Let A_t and E_t denote the numbers of allied and enemy units alive at timestep t, respectively, and let N_{e} be the initial number of enemies. The outcomes evaluate allied survival and enemy elimination:

1. **Allied survival.** The indicative outcome $z_{\text{ind}}^1(\boldsymbol{\tau}) = T_{\boldsymbol{\tau}}^{-1} \sum_t A_t$ is the time-averaged number of surviving allied units. An allied unit surviving longer therefore contributes more to the trajectory evaluation. The pass-fail outcome $z_{\text{pf}}^1 = 1$ when at least one allied unit remains alive at the final valid timestep.
2. **Enemy elimination.** The indicative outcome $z_{\text{ind}}^2(\boldsymbol{\tau}) = T_{\boldsymbol{\tau}}^{-1} \sum_t (N_{\text{e}} - E_t)$ is the time-averaged number of eliminated enemies. An enemy eliminated earlier contributes over more subsequent timesteps and therefore produces a higher score. The pass-fail outcome $z_{\text{pf}}^2 = 1$ when all N_{e} enemies have been eliminated at the final valid timestep.

Accordingly, the pass-fail thresholds are set to $[1, N_{\text{e}}]$: $[1, 5]$, $[1, 10]$, and $[1, 20]$ for the 5-vs-5, 10-vs-10, and 20-vs-20 scenarios, respectively. The pass-fail outcomes encode successful battle completion, while the indicative outcomes provide graded supervision for preserving allied units and eliminating enemies earlier.

QMIX-Dense and MAPPO-Dense are included as oracle references using the standard dense SMACv2 reward. Let $\Delta H_t^{\text{enemy}}$ denote the reduction in total enemy health and shields, let K_t be the number of enemies eliminated at timestep t, and let $I_{\text{win},t}$ indicate victory. The unscaled dense reward is:

$$\widetilde{r}_t^{\text{dense}} = \max(\Delta H_t^{\text{enemy}} + 10K_t, 0) + 200 I_{\text{win},t}. \tag{52}$$

Because only positive rewards are enabled, allied damage and deaths do not incur penalties, and defeat contributes no additional reward. For the 5-vs-5 scenarios, the dense reward follows the standard environment normalization:

$$r_t^{\text{dense}} = \frac{20}{H_{\text{max}}^{\text{enemy}} + 5 \times 10 + 200} \widetilde{r}_t^{\text{dense}}, \tag{53}$$

where $H_{\text{max}}^{\text{enemy}}$ denotes the total maximum enemy health and shields used to initialize the normalization constant. These dense variants provide oracle references with direct damage, elimination, and victory feedback. The main evaluation focuses on the 5-vs-5 scenarios, while results and analyses for the larger 10-vs-10 and 20-vs-20 settings are provided in Appendix H.4.

Table 2: Key hyperparameters for the QMIX and MAPPO backbones.

Parameter	**QMIX**	**MAPPO**
Discount factor γ	0.99	0.99
Training batch size	32 episodes	32 episodes
Buffer capacity	5,000 episodes	32 episodes
Learning rate	3×10^{-4}	3×10^{-4}
Recurrent hidden dimension	128	64
Gradient-norm threshold	10	10
Action selection	ϵ-greedy	Policy sampling
TD(λ)	0.6	–
Mixing embedding dimension	32	–
Hypernetwork embedding dimension	64	–
PPO clipping parameter	–	0.2
PPO epochs	–	5
Number of mini-batches	–	1
Recurrent chunk length	–	10
GAE parameter	–	0.95
Entropy coefficient	–	0.005
Huber-loss threshold	–	10
Value normalization	–	Enabled
Advantage normalization	–	Enabled

Table 3: Key FoX and LAIES settings on SMACv2.

LAIES		FoX	
Parameter	Value	Parameter	Value
Predictor epochs	25	Auxiliary batch	16
Target-update interval	200	Predictor epochs	25
Annealing scale	2×10^6	Predictor width	128
Sample size	16	(β_1, β_2)	$(0.01, 0.02)$
Clipping thresholds	$(0.002, 1)$	`round`	3
(β_1, β_2)	$(100, 0.2)$	α	0.3
Enemy-state dimension	3		

G.2 Implementation Details

The backbone implementations retain their standard optimization structures. QMIX uses off-policy replay with value decomposition and ϵ-greedy exploration, while MAPPO uses on-policy rollouts with recurrent actors and a centralized critic. Within each backbone family, the test-driven variant and its baseline use the same policy architecture and optimization settings, allowing the comparison to isolate the effect of test-driven reward construction. The key parameters are summarized in Table 2. All methods run eight environments in parallel and are evaluated every 25,000 environment steps using 32 greedy episodes.

FoX and LAIES provide QMIX-based and MAPPO-based sparse-reward exploration baselines, respectively. Table 3 summarizes their baseline-specific settings on SMACv2. Both use the parallel-run and evaluation protocol described above.

The additional parameters shared by Td-QMIX and Td-MAPPO are reported in Table 4. The return model maps indicative outcomes to segment-level scores, while the reward model decomposes these scores into per-step rewards from global states and joint actions. Both models use ensembles whose mean predictions provide the final return and reward estimates. Training begins after the warm-up period, and both models are updated whenever 32 new episodes have been collected.

Table 4: Key hyperparameters for the test-driven components.

Parameter	Value
Warm-up environment steps	10,000
Episodes accumulated per update	32
Return-buffer size	256×4 for LBF and dense Cooperative Navigation; 512×1 for the remaining tasks
Preference sampling	Mixed uniform and targeted sampling
Preference evaluator	Robust hierarchical comparison
Preference tolerance ϵ_{pref}	0.02
Minimum normalization scale $\sigma_{\min}$	0.1
Return ensemble size	3
Reward ensemble size	3
Return-network hidden dimension	64
Reward-network hidden dimension	128
Network architecture	Three-layer MLP with LeakyReLU and Tanh output
Return/reward optimizer	Adam
Return/reward learning rate	10^{-3}
Return-model batch size	128 segment pairs
Reward-model batch size	128 segments
Return-model training passes	10 per update
Reward-model training passes	10 per update
Change-loss gradient coefficient	1.0
Return stability threshold c_{es}	10
Soft uniform prior λ_{step}	1.0
Decomposition discount factor	0.99

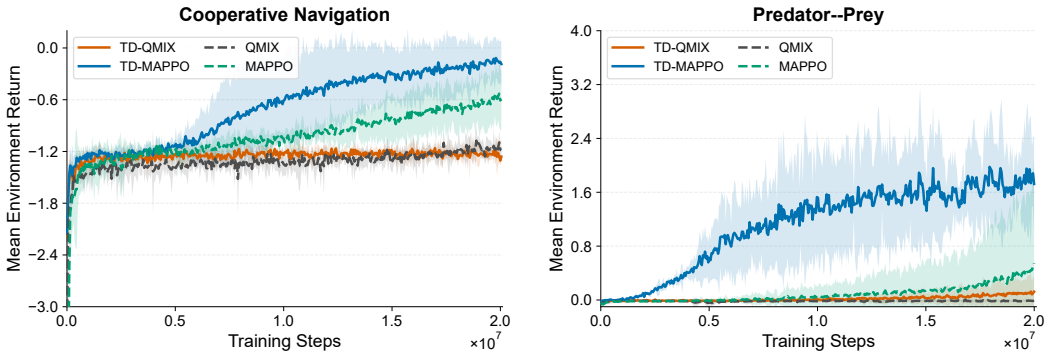

Figure 8: Environment-return training curves on MPE. The left and right panels show Cooperative Navigation and Predator–Prey, respectively.

H Additional Experimental Results

H.1 Training Curves Measured by Environment Rewards

For conventional comparison with the original MARL backbones, this section reports performance measured by the environment-defined rewards. For TdMARL, these rewards are used only for evaluation rather than policy optimization.

Figure 8 reports the dense environment returns on Cooperative Navigation and Predator–Prey. On Cooperative Navigation, Td-MAPPO improves steadily and substantially outperforms MAPPO, whereas both QMIX-based methods remain at relatively low returns. On Predator–Prey, Td-MAPPO again achieves the strongest performance, while Td-QMIX provides a smaller but consistent improvement over QMIX. These conventional return curves agree with the behavioral results in Section 5.2, showing that the test-derived guidance translates into performance under the original dense-

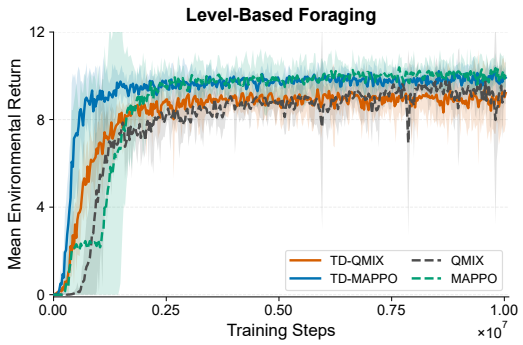

Figure 9: Environment-return training curves on LBF, where the episode return equals the number of successful pickups.

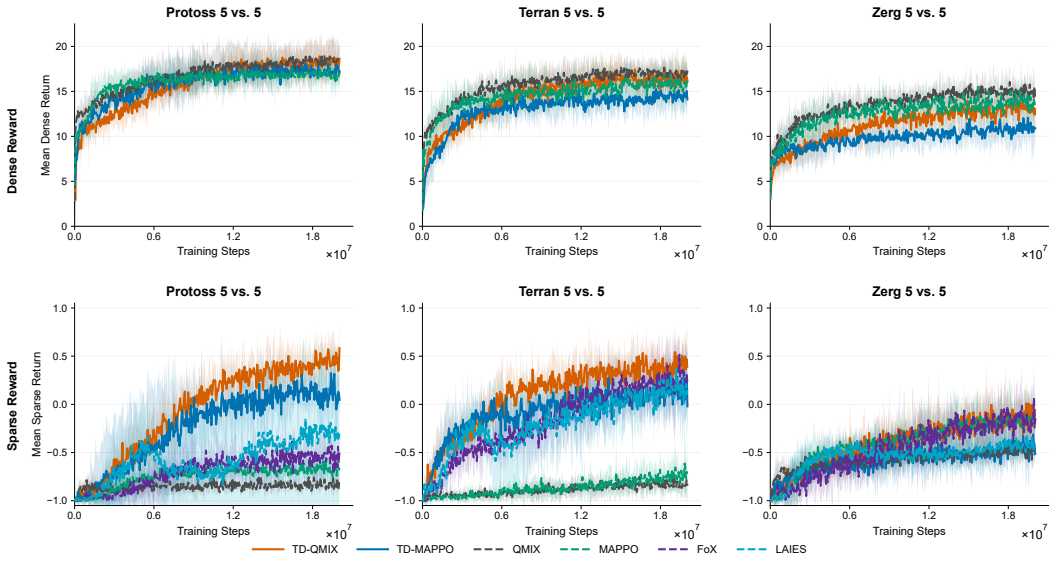

Figure 10: Dense- and sparse-return training curves on the SMACv2 5-vs-5 scenarios.

reward objectives. They also confirm that the resulting improvement remains influenced by the coordination capability of the underlying backbone.

In LBF, the environment return equals the number of successfully collected fruits. Figure 9 shows that all methods eventually achieve approximately nine to ten successful pickups per episode. The TdMARL variants remain comparable to their original backbones in final task performance while generally learning faster during the early stages, particularly for Td-MAPPO. Combined with the improved auxiliary behaviors reported in Section 5.2, these results show that extending the tester with intermediate cooperative criteria does not compromise the original pickup objective.

Figure 10 reports training curves evaluated under both the original dense reward and the sparse terminal reward. Across the three races, Td-QMIX and Td-MAPPO achieve substantially higher sparse returns than their respective backbones and generally approach the dense-reward references, particularly on Protoss and Terran. The exploration baselines also obtain useful learning signals, but their performance varies across scenarios. FoX achieves strong sparse returns on Terran and Zerg, approaching Td-QMIX late in training, while remaining clearly behind on Protoss. LAIES is competitive on Terran but shows greater variability on Protoss and performs poorly on Zerg, where Td-MAPPO retains a clear advantage. Evaluated under the original dense reward, the Td methods also achieve strong returns, showing that test-derived guidance supports the combat progress and

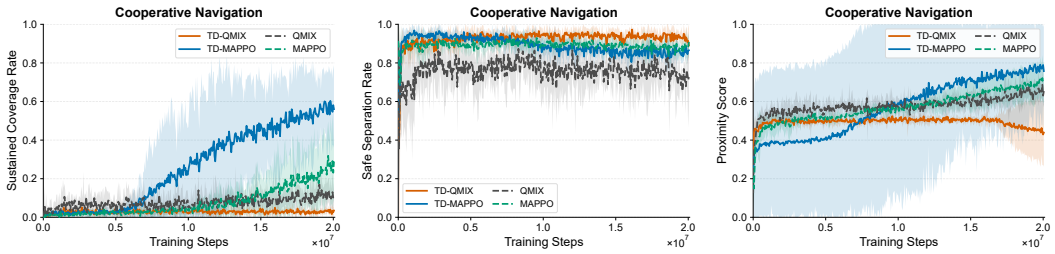

Figure 11: Behavioral performance on Cooperative Navigation. From left to right, the panels report sustained landmark coverage, safe separation, and landmark proximity.

Table 5: Cooperative Navigation behavioral metrics. Sustained coverage is the primary task metric and reflects persistent landmark occupancy.

Metric	Td-QMIX	Td-MAPPO	QMIX	MAPPO
z^1_{pf}: Sustained coverage rate	0.03 ± 0.00	$\mathbf{0.55 \pm 0.12}$	0.11 ± 0.01	0.23 ± 0.10
z^2_{pf}: Safe separation rate	$\mathbf{0.93 \pm 0.01}$	0.85 ± 0.00	0.74 ± 0.03	0.88 ± 0.02
z^1_{ind}: Proximity score	0.46 ± 0.09	$\mathbf{0.76 \pm 0.18}$	0.65 ± 0.02	0.69 ± 0.06

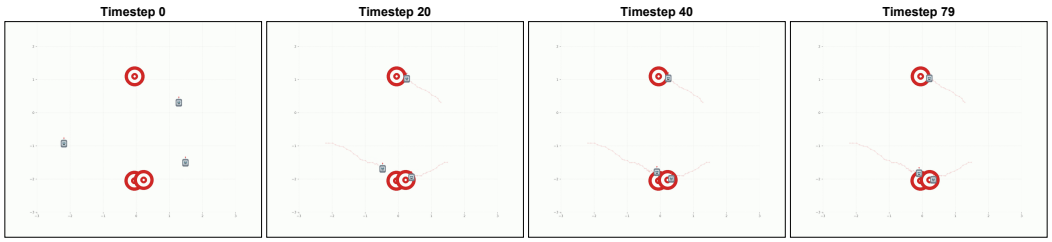

Figure 12: Representative Cooperative Navigation rollout of Td-MAPPO from timestep 0 to 79. Gray squares denote agents, red targets denote landmarks, and traces show the agents' paths.

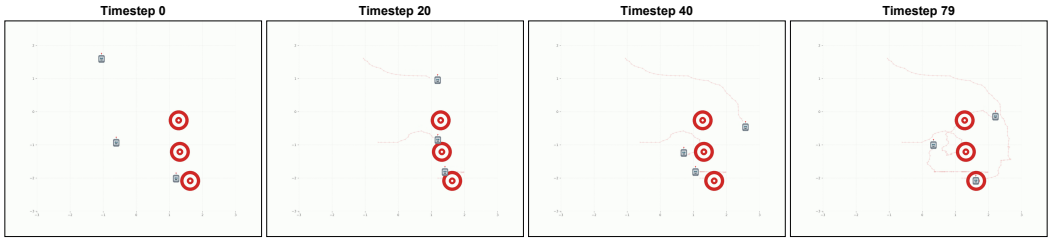

Figure 13: Representative Cooperative Navigation rollout of Td-QMIX from timestep 0 to 79. Gray squares denote agents, red targets denote landmarks, and traces show the agents' paths.

victories measured by that reward. Together with the allied-survival and enemy-elimination metrics in Table 7 in Appendix H.3, these curves show that simple task-level tests provide effective guidance without manually designing dense per-step rewards.

H.2 MPE Performance and Behavioral Analysis

Cooperative Navigation. Figure 11 presents the evolution of sustained landmark coverage, safe separation, and landmark proximity during training. Td-MAPPO exhibits a sustained increase in coverage after approximately 6×10^6 steps, accompanied by a gradual improvement in proximity.

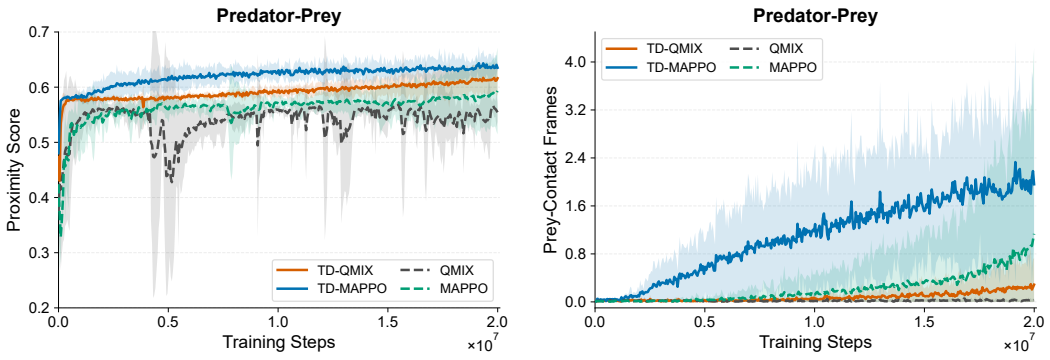

Figure 14: Behavioral performance on Predator–Prey. The left panel reports predator–prey proximity, and the right panel reports the prey-contact frames.

Table 6: Predator–Prey behavioral metrics. The prey-contact frames is the primary task metric and reflects successful prey interception.

Metric	Td-QMIX	Td-MAPPO	QMIX	MAPPO
z^1_{ind}: Predator–prey proximity	0.61 ± 0.02	$\mathbf{0.64 \pm 0.01}$	0.55 ± 0.01	0.58 ± 0.04
z^2_{ind}: Prey-contact frames	0.22 ± 0.27	$\mathbf{1.93 \pm 0.82}$	0.02 ± 0.00	0.77 ± 1.32

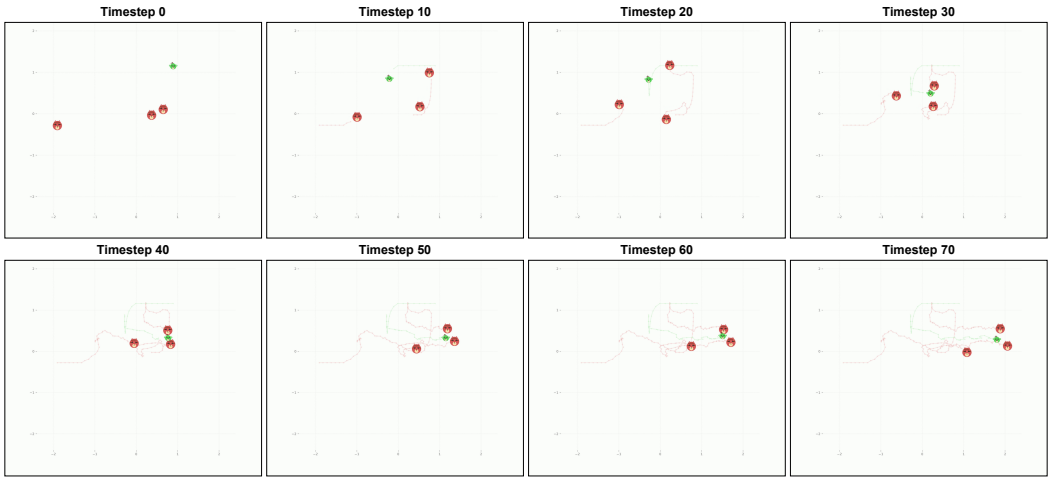

Figure 15: Representative Predator–Prey rollout of Td-MAPPO from timestep 0 to 70. Red agents are predators, the green agent is the prey, and the traces show their paths up to each timestep.

Table 5 shows that it ultimately achieves a coverage rate of 0.55 and a proximity score of 0.76, compared with 0.23 and 0.69 for MAPPO. Its safe separation rate remains high at 0.85, indicating that the improved coverage does not arise simply from uncontrolled agent crowding. In contrast, Td-QMIX obtains the highest safe separation rate of 0.93 but only 0.03 sustained coverage and 0.46 proximity. QMIX similarly reaches only 0.11 coverage. This discrepancy highlights that safe movement or temporary proximity does not necessarily produce persistent and complementary landmark assignments.

The representative rollouts provide a spatial explanation for these results. In Figure 12, the policy learned by Td-MAPPO assigns agents to the landmark regions by timestep 20, with the snapshots at timesteps 40 and 79 showing that these assignments are maintained. By contrast, Figure 13 shows agents approaching the landmark region but continuing to move without establishing stable assign-

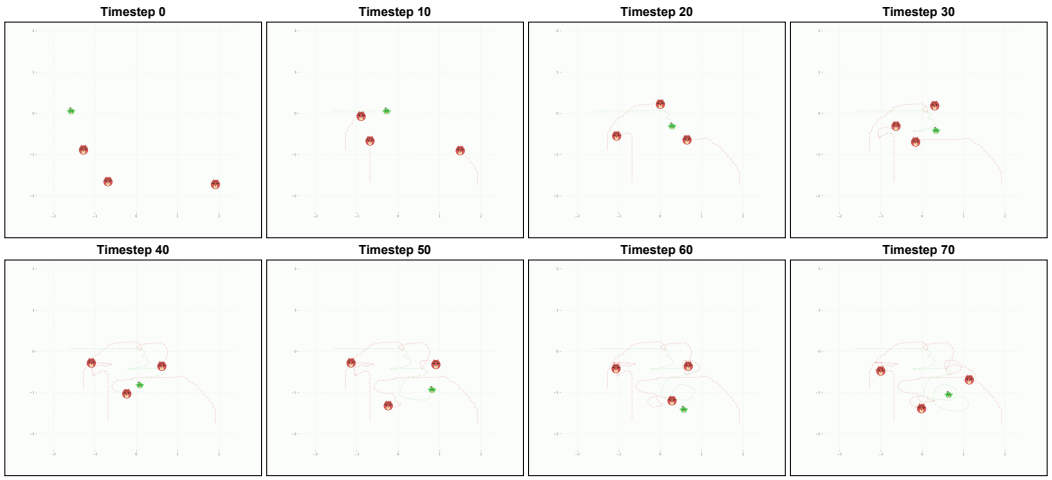

Figure 16: Representative Predator–Prey rollout of Td-QMIX from timestep 0 to 70. Red agents are predators, the green agent is the prey, and the traces show their paths up to each timestep.

ments. The qualitative behavior is therefore consistent with the low coverage rate of Td-QMIX. Together, the results show that Td-MAPPO converts the test specifications into persistent landmark coordination, whereas the weak coverage of both QMIX-based methods reflects the influence of the underlying backbone.

Predator–Prey. Figure 14 reports the evolution of predator–prey proximity and prey-contact frames, while Table 6 summarizes their final values. Both Td variants reduce the distance to the prey relative to their backbones. However, the difference between proximity and contact performance reveals that approaching the prey alone is insufficient for effective capture. Td-QMIX reaches a proximity score of 0.61 but records only 0.22 contact frames, suggesting that its predators approach the prey without consistently restricting its movement. In contrast, Td-MAPPO achieves the highest proximity score of 0.64 and 1.93 contact frames. The snapshots from timesteps 0 to 70 in Figures 15 and 16 further illustrate this distinction. Td-MAPPO approaches from multiple directions and forms an encircling pattern, whereas Td-QMIX exhibits less structured pursuit without stable containment.

H.3 Detailed SMACv2 5-vs-5 Results

Table 7 reports performance across Protoss, Terran, and Zerg scenarios. Win rate is the primary metric, while allied survival and enemy elimination characterize team preservation and combat progress, respectively. Episode length provides a more complete account of the battle trajectories.

H.4 SMACv2 Results with Larger Teams

Figure 17 and Table 8 evaluate TdMARL with larger teams. In the 10-vs-10 scenarios, both Td methods learn effective policies while the sparse-reward backbones remain at zero win rate. On Protoss and Terran, Td-QMIX achieves 0.68 and 0.71, approaching the QMIX-Dense results of 0.77 and 0.74. Td-MAPPO similarly obtains 0.52 and 0.47, closely matching the MAPPO-Dense results of 0.53 and 0.46. Larger gaps appear on Zerg, where Td-QMIX and Td-MAPPO reach 0.42 and 0.23, compared with 0.56 and 0.39 for their dense-reward counterparts.

These performance gaps become more widespread in the 20-vs-20 scenarios. Td-QMIX achieves 0.52, 0.52, and 0.16 on Protoss, Terran, and Zerg, compared with 0.75, 0.67, and 0.49 for QMIX-Dense. Td-MAPPO obtains 0.33, 0.31, and 0.09, compared with 0.42, 0.38, and 0.31 for MAPPO-Dense. Nevertheless, both Td methods continue to learn meaningful policies, whereas their sparse-reward backbones remain at zero win rate. Moreover, several curves, particularly those of Td-QMIX, are still increasing near the 20 million-step boundary. This suggests that larger teams reduce learning efficiency under the fixed training budget rather than preventing test-derived reward learning entirely.

Table 7: Evaluation results on SMACv2 5-vs-5 scenarios. Bold marks the best win rate and behavioral metrics in each scenario.

Method	Win Rate	z^1_{ind}: Allies	z^2_{ind}: Enemies	Length
Protoss 5-vs-5				
QMIX	0.00 ± 0.00	3.16 ± 0.01	0.01 ± 0.00	143.99 ± 0.21
MAPPO	0.00 ± 0.00	3.30 ± 0.01	0.00 ± 0.00	164.98 ± 1.91
FoX	0.23 ± 0.01	3.43 ± 0.01	1.12 ± 0.04	55.90 ± 0.50
LAIES	0.35 ± 0.25	3.42 ± 0.22	1.27 ± 0.83	76.63 ± 14.62
Td-QMIX	$\mathbf{0.72 \pm 0.02}$	$\mathbf{3.95 \pm 0.05}$	$\mathbf{1.80 \pm 0.04}$	67.38 ± 0.91
Td-MAPPO	0.55 ± 0.03	3.72 ± 0.03	1.66 ± 0.04	69.37 ± 0.67
QMIX-Dense	$\mathbf{0.72 \pm 0.03}$	3.80 ± 0.05	1.76 ± 0.02	63.02 ± 0.42
MAPPO-Dense	0.53 ± 0.03	3.52 ± 0.05	1.69 ± 0.03	64.32 ± 1.17
Terran 5-vs-5				
QMIX	0.00 ± 0.00	3.07 ± 0.02	0.03 ± 0.01	113.34 ± 3.27
MAPPO	0.06 ± 0.06	3.08 ± 0.09	0.61 ± 0.22	100.01 ± 12.59
FoX	0.62 ± 0.06	3.94 ± 0.06	1.33 ± 0.03	45.35 ± 0.78
LAIES	0.57 ± 0.04	3.80 ± 0.05	1.49 ± 0.02	51.31 ± 1.11
Td-QMIX	0.69 ± 0.02	$\mathbf{4.13 \pm 0.01}$	$\mathbf{1.78 \pm 0.03}$	56.41 ± 0.85
Td-MAPPO	0.57 ± 0.02	3.83 ± 0.03	1.57 ± 0.05	51.88 ± 0.68
QMIX-Dense	$\mathbf{0.77 \pm 0.01}$	4.05 ± 0.01	1.62 ± 0.02	53.34 ± 1.37
MAPPO-Dense	0.57 ± 0.01	3.76 ± 0.02	1.49 ± 0.02	50.75 ± 0.91
Zerg 5-vs-5				
QMIX	0.00 ± 0.00	2.98 ± 0.02	0.76 ± 0.05	142.28 ± 2.65
MAPPO	0.12 ± 0.08	2.74 ± 0.12	1.70 ± 0.59	148.05 ± 7.95
FoX	0.41 ± 0.01	3.75 ± 0.01	1.22 ± 0.04	29.46 ± 0.28
LAIES	0.02 ± 0.01	2.83 ± 0.22	1.40 ± 0.32	140.16 ± 3.04
Td-QMIX	0.43 ± 0.01	$\mathbf{3.84 \pm 0.00}$	$\mathbf{1.80 \pm 0.02}$	40.14 ± 0.56
Td-MAPPO	0.27 ± 0.02	3.54 ± 0.06	1.78 ± 0.02	38.02 ± 1.11
QMIX-Dense	$\mathbf{0.48 \pm 0.01}$	3.83 ± 0.02	1.54 ± 0.02	31.43 ± 0.21
MAPPO-Dense	0.35 ± 0.01	3.56 ± 0.00	1.67 ± 0.01	30.22 ± 0.33

Table 8: Final win rates on the SMACv2 10-vs-10 and 20-vs-20 scenarios.

	10-vs-10			**20-vs-20**		
Method	Protoss	Terran	Zerg	Protoss	Terran	Zerg
QMIX	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
MAPPO	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Td-QMIX	0.68 ± 0.05	0.71 ± 0.04	0.42 ± 0.01	0.52 ± 0.09	0.52 ± 0.09	0.16 ± 0.06
Td-MAPPO	0.52 ± 0.14	0.47 ± 0.02	0.23 ± 0.03	0.33 ± 0.01	0.31 ± 0.06	0.09 ± 0.02
QMIX-Dense	$\mathbf{0.77 \pm 0.02}$	$\mathbf{0.74 \pm 0.02}$	$\mathbf{0.56 \pm 0.02}$	$\mathbf{0.75 \pm 0.03}$	$\mathbf{0.67 \pm 0.02}$	$\mathbf{0.49 \pm 0.03}$
MAPPO-Dense	0.53 ± 0.12	0.46 ± 0.04	0.39 ± 0.04	0.42 ± 0.09	0.38 ± 0.04	0.31 ± 0.04

Improving the efficiency of TdMARL in complex environments with more agents therefore remains an important limitation and direction for future work.

H.5 LBF Performance and Behavioral Analysis

Figure 18 reports successful pickups and the three auxiliary behavioral indicators throughout training, while Table 9 summarizes their final values. Td-QMIX and Td-MAPPO achieve pickup performance comparable to their respective backbones. Their most consistent improvement appears in agent pairing, with smaller gains in simultaneous fruit coverage and similar loading-attempt frequencies.

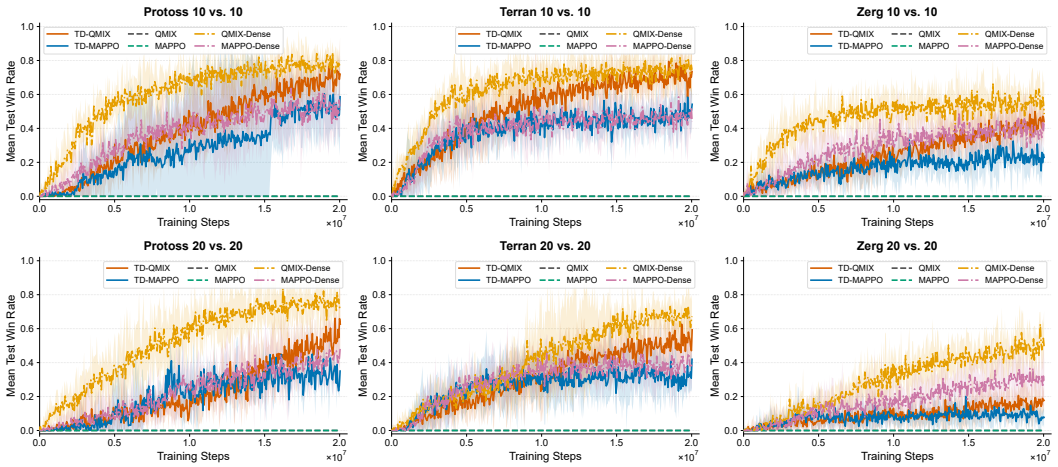

Figure 17: Win rates on SMACv2 with 10 and 20 agents per team.

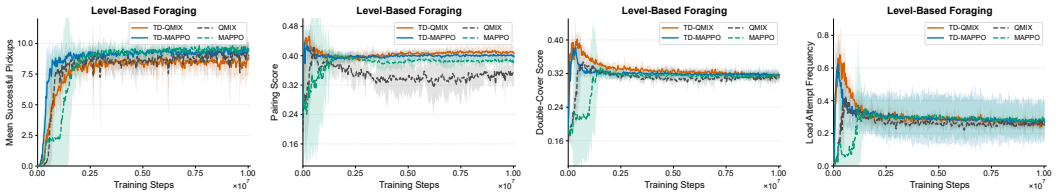

Figure 18: LBF performance and behavioral metrics. From left to right: mean successful pickups, agent-pairing score, double-cover score, and load-attempt frequency.

Table 9: LBF behavioral metrics. Mean successful pickups is the primary task metric and reflects cooperative task completion.

Metric	QMIX	Td-QMIX	MAPPO	Td-MAPPO
ind_1: Agent pairing score	0.35 ± 0.01	$\mathbf{0.41 \pm 0.01}$	0.39 ± 0.01	0.40 ± 0.01
ind_2: Double-cover score	0.31 ± 0.01	$\mathbf{0.32 \pm 0.01}$	0.31 ± 0.00	0.32 ± 0.01
ind_3: Load-attempt frequency	0.26 ± 0.01	0.26 ± 0.01	$\mathbf{0.28 \pm 0.07}$	0.27 ± 0.08
ind_4: Mean successful pickups	9.16 ± 0.38	8.93 ± 0.76	$\mathbf{10.02 \pm 0.34}$	9.84 ± 0.19

The proximity- and attempt-based indicators peak early before converging to stable values, whereas successful pickups continue to increase. LBF episodes retain a fixed horizon, and each collected fruit is respawned at a new location. As policies become more efficient, agents spend less time remaining beside a fruit or repeatedly executing LOAD and more time moving toward subsequent fruits. The decreasing auxiliary scores are therefore consistent with reduced redundant behavior rather than declining pickup ability.

Figures 19 and 20 compare representative rollouts from timesteps 7 to 14. QMIX agents separate more widely while relocating after a pickup, whereas Td-QMIX maintains closer pairing when approaching the next fruit. This comparison illustrates how the auxiliary pairing specification changes the learned coordination pattern while preserving the original pickup objective.

I ADDITIONAL ABLATION RESULTS

Figure 21 presents the complete learning curves, while Table 10 reports performance over the final 10% of training. Four variants isolate the main methodological choices. *TdRL preference* replaces the multi-agent preference evaluator with the original TdRL comparison rule. *TdRL sampling* re-

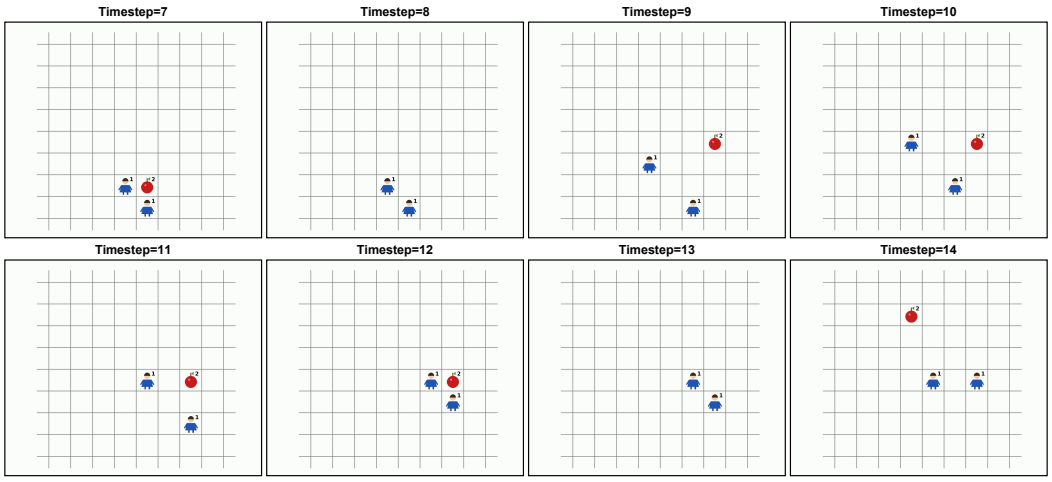

Figure 19: Representative QMIX rollout in LBF from timesteps 7 to 14.

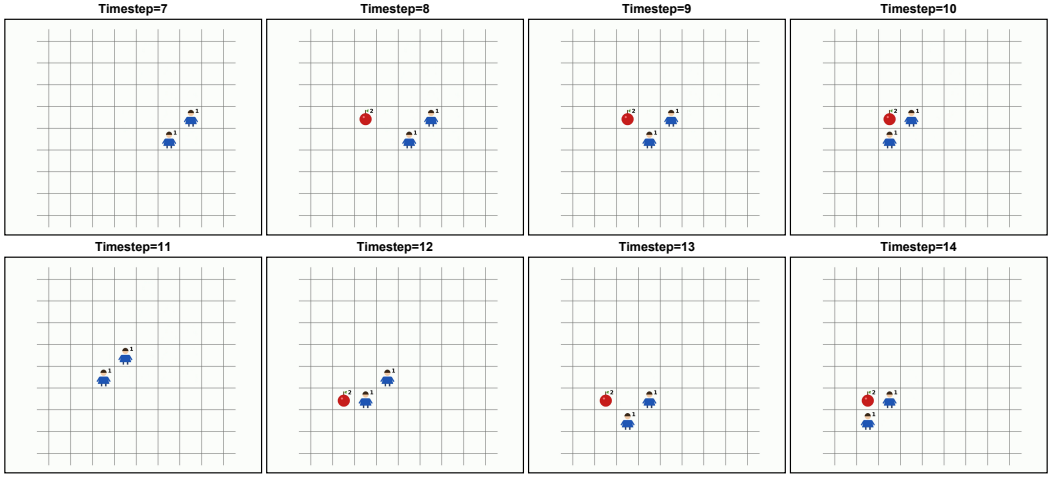

Figure 20: Representative Td-QMIX rollout in LBF from timesteps 7 to 14.

Table 10: Ablation performance on Cooperative Navigation, LBF, and Protoss 5-vs-5, measured by sustained coverage, successful pickups, and win rate, respectively.

	Cooperative Navigation		LBF		Protoss 5-vs-5	
Variant	Td-QMIX	Td-MAPPO	Td-QMIX	Td-MAPPO	Td-QMIX	Td-MAPPO
TdMARL (full)	$\mathbf{0.03 \pm 0.01}$	$\mathbf{0.55 \pm 0.12}$	8.92 ± 1.20	$\mathbf{9.84 \pm 0.30}$	$\mathbf{0.72 \pm 0.03}$	0.55 ± 0.05
TdRL preference	0.00 ± 0.00	0.01 ± 0.02	8.94 ± 1.40	9.04 ± 2.63	0.71 ± 0.01	$\mathbf{0.59 \pm 0.05}$
TdRL sampling	0.01 ± 0.01	0.31 ± 0.49	8.39 ± 0.49	8.25 ± 0.57	0.67 ± 0.05	0.39 ± 0.05
Direct reward training	0.02 ± 0.01	0.02 ± 0.00	$\mathbf{9.13 \pm 0.40}$	7.88 ± 1.47	0.65 ± 0.03	0.53 ± 0.02
w/o soft uniform prior	0.01 ± 0.02	0.09 ± 0.16	7.75 ± 0.78	7.96 ± 2.03	0.20 ± 0.28	0.09 ± 0.15

places the targeted sampler with uniformly sampled trajectory pairs. *Direct reward training* removes the separate return-learning and return-decomposition stages. Finally, *w/o soft uniform prior* sets the coefficient of the per-step decomposition prior to zero.

Preference evaluation. The original TdRL preference rule produces its largest degradation on Cooperative Navigation, where Td-MAPPO decreases from 0.55 sustained coverage to 0.01. For Td-

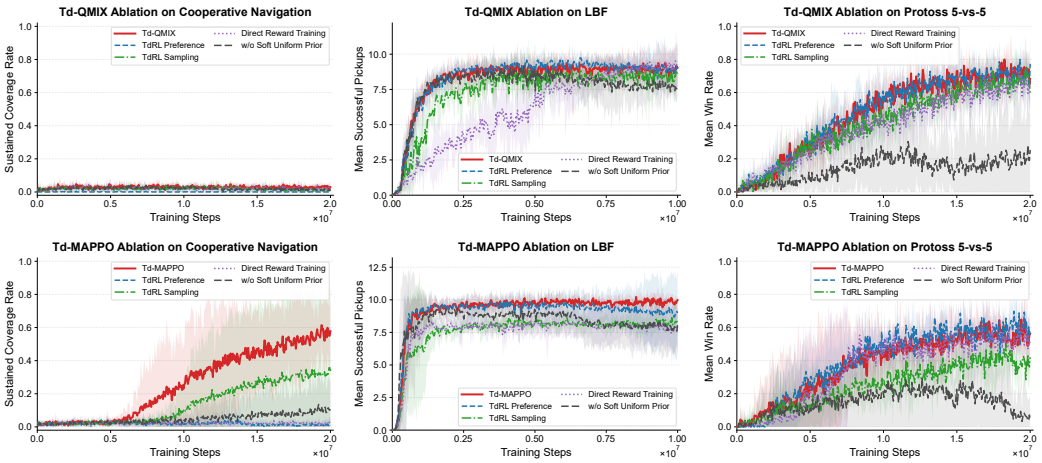

Figure 21: Ablation results on Cooperative Navigation, LBF, and Protoss 5-vs-5.

QMIX, the corresponding score decreases from 0.03 to 0.00, while its LBF pickups and Protoss win rate remain close to the complete method at 8.94 and 0.71, respectively. The original rule also remains competitive for Td-MAPPO on LBF and Protoss. These results suggest that the proposed evaluator is particularly beneficial on Cooperative Navigation, where retaining indicative comparisons after identical successful pass-fail outcomes and suppressing noise-driven preferences provide more reliable supervision.

Sampling strategy. Uniform TdRL sampling reduces performance for both backbones. Td-QMIX decreases from 0.03 to 0.01 on Cooperative Navigation, from 8.92 to 8.39 on LBF, and from 0.72 to 0.67 on Protoss. The reductions are larger for Td-MAPPO, whose performance decreases from 0.55 to 0.31, from 9.84 to 8.25, and from 0.55 to 0.39, respectively. By including pairs with different outcomes on the least-passed test, the targeted sampler focuses return learning on the cooperative requirement that the current policy most frequently fails to satisfy.

Direct reward training. Following the direct reward-learning approach in TdRL (Yu et al., 2026), this variant removes the intermediate return model f_ψ. For each segment, the discounted sum of per-step reward predictions defines its segment score:

$$\widehat{G}_\phi(\boldsymbol{\tau}) = \sum_{t=0}^{T_{\boldsymbol{\tau}}-1} \gamma^t r_\phi(s_t, \mathbf{a}_t). \tag{54}$$

For each trajectory pair, the two scores are passed through log-softmax and optimized directly against the test-induced preference label y_{12}. The reward network is therefore trained end to end without first learning $\widehat{R}_\psi$ or performing return-to-reward decomposition. For Td-QMIX, direct training changes Cooperative Navigation coverage from 0.03 to 0.02, increases LBF pickups from 8.92 to 9.13, and reduces the Protoss win rate from 0.72 to 0.65. For Td-MAPPO, it reduces Cooperative Navigation coverage from 0.55 to 0.02 and LBF pickups from 9.84 to 7.88, while remaining close on Protoss at 0.53 versus 0.55. Although competitive in several settings, direct training is less consistent across tasks and MARL backbones than the explicit return construction and decomposition.

Soft uniform prior. Removing the prior produces the most consistent degradation across tasks and backbones. On Protoss, the win rate decreases from 0.72 to 0.20 for Td-QMIX and from 0.55 to 0.09 for Td-MAPPO. LBF pickups similarly decline from 8.92 to 7.75 and from 9.84 to 7.96, respectively. On Cooperative Navigation, Td-MAPPO coverage falls from 0.55 to 0.09. The corresponding Td-QMIX scores are 0.03 and 0.01, although all QMIX-based variants perform poorly on this task. These results show that the prior stabilizes temporal reward decomposition and improves robustness across MARL backbones.

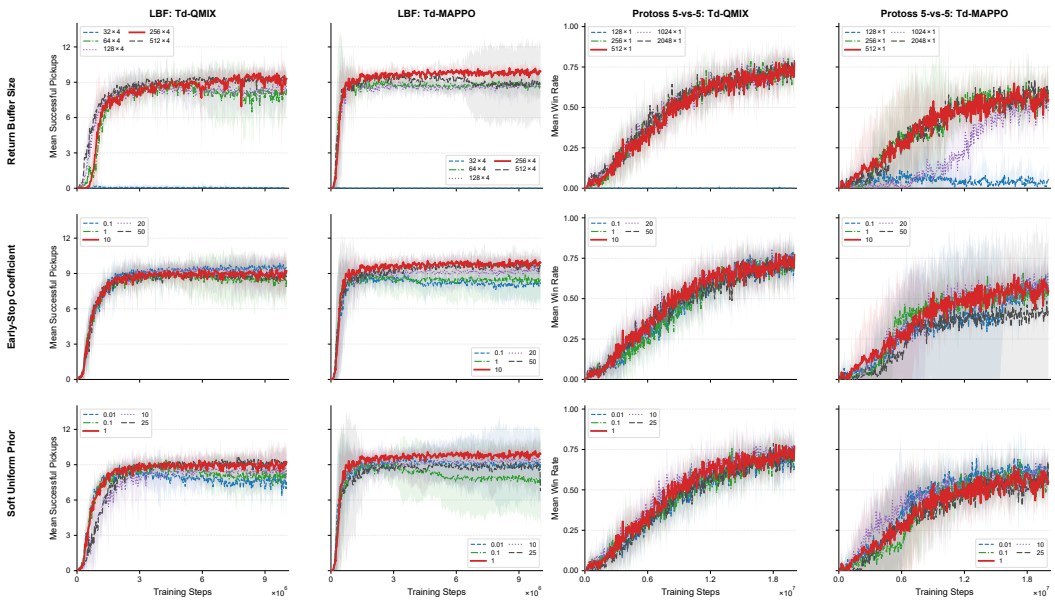

Figure 22: Sensitivity profiles for buffer size, early-stop, and prior coefficients on LBF and Protoss 5-vs-5.

Table 11: Hyperparameter sensitivity on LBF and Protoss 5-vs-5. *denotes settings used for performance evaluation.

LBF			Protoss 5-vs-5		
Setting	Td-QMIX	Td-MAPPO	Setting	Td-QMIX	Td-MAPPO
Return Buffer Size					
32×4	0.03 ± 0.11	0.00 ± 0.00	128×1	0.00 ± 0.00	0.04 ± 0.06
64×4	7.92 ± 1.77	8.72 ± 0.71	256×1	0.72 ± 0.02	0.57 ± 0.06
128×4	8.34 ± 1.65	8.77 ± 0.95	$512 \times 1^*$	0.72 ± 0.03	0.55 ± 0.05
$256 \times 4^*$	8.93 ± 1.21	$\mathbf{9.84 \pm 0.30}$	1024×1	$\mathbf{0.73 \pm 0.02}$	0.52 ± 0.04
512×4	$\mathbf{9.20 \pm 0.81}$	8.89 ± 3.34	2048×1	0.72 ± 0.03	$\mathbf{0.58 \pm 0.07}$
Early-Stop Coefficient					
0.1	$\mathbf{9.50 \pm 0.21}$	8.12 ± 1.43	0.1	0.71 ± 0.04	0.56 ± 0.09
1	8.40 ± 1.59	8.42 ± 1.78	1	0.70 ± 0.04	0.54 ± 0.07
10^*	8.93 ± 1.21	$\mathbf{9.84 \pm 0.30}$	10^*	$\mathbf{0.72 \pm 0.03}$	0.55 ± 0.05
20	9.05 ± 1.31	9.27 ± 1.08	20	0.72 ± 0.03	$\mathbf{0.58 \pm 0.05}$
50	8.71 ± 1.58	9.60 ± 1.14	50	0.70 ± 0.04	0.41 ± 0.44
Soft Uniform Prior					
0.01	7.60 ± 0.74	9.02 ± 3.04	0.01	0.69 ± 0.06	$\mathbf{0.61 \pm 0.04}$
0.1	8.03 ± 1.05	7.71 ± 3.15	0.1	0.70 ± 0.04	0.56 ± 0.03
1^*	8.93 ± 1.21	$\mathbf{9.84 \pm 0.30}$	1^*	$\mathbf{0.72 \pm 0.03}$	0.55 ± 0.05
10	8.54 ± 1.32	9.33 ± 1.77	10	0.72 ± 0.02	0.59 ± 0.05
25	$\mathbf{9.18 \pm 1.12}$	8.75 ± 1.85	25	0.70 ± 0.01	0.55 ± 0.01

J Hyperparameter Sensitivity Analysis

Figure 22 and Table 11 report mean successful pickups on LBF and win rate on Protoss 5-vs-5.

These hyperparameters affect three successive stages of TdMARL: the availability of trajectory data, the stability of return learning, and the temporal decomposition of learned returns.

The return buffer first balances sample diversity against data freshness. A small buffer contains insufficient trajectory variation for constructing informative preference pairs, especially when successful coordination is rare. On LBF, 32×4 yields only 0.03 pickups for Td-QMIX and 0.00 for Td-MAPPO. Similarly, 1×128 produces Protoss win rates of only 0.00 and 0.04. Performance recovers substantially from 64×4 on LBF and 1×256 on Protoss. Conversely, an excessively large buffer retains more trajectories generated by earlier policies. These stale trajectories may become less representative as the joint policy evolves, slowing return-model adaptation. This effect is most visible for Td-MAPPO on LBF, where increasing the buffer from 256×4 to 512×4 reduces pickups from 9.84 to 8.89 and increases the reported uncertainty from ± 0.30 to ± 3.34. Larger buffers remain stable on Protoss, suggesting that the appropriate capacity depends on how rapidly the trajectory distribution changes.

Given sufficient trajectory data, the early-stop coefficient regulates the stability of return-model updates. Performance remains stable for $c_{\text{es}} \in [1, 20]$. Increasing it to 50 is less reliable for Td-MAPPO on Protoss, reducing the win rate to 0.41 ± 0.44 and indicating unstable return learning under this setting.

After learning segment returns, the prior coefficient controls their temporal decomposition into per-step rewards. Weak regularization is less reliable on LBF. Specifically, $\lambda_{\text{step}} = 0.01$ reduces Td-QMIX to 7.60 pickups, while $\lambda_{\text{step}} = 0.1$ gives Td-MAPPO 7.71 ± 3.15. Moderate values between 1 and 10 remain stable across both environments and backbones. Overall, although individual settings occasionally achieve higher means, the defaults provide the most consistent performance without environment-specific tuning.