EFFICIENT MULTI-AGENT REINFORCEMENT LEARNING UNDER SPARSE BINARY TEAM REWARDS WITH FEW SUBOPTIMAL PRIOR TRAJECTORIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning efficient cooperative policies under sparse binary team rewards is particularly challenging when only a few suboptimal prior trajectories are available. Binary terminal feedback collapses trajectories of different qualities into identical outcomes, obscuring value estimation, while representative methods struggle to recover these hidden distinctions. We therefore propose TRACE, an online MARL framework that restores within-outcome reward resolution under binary feedback and converts the resulting trajectory-level supervision into discriminative transition-level learning signals. TRACE requires only the meaning of the original sparse binary reward and available terminal-state variables. Given this limited information and the proposed evaluator template, a fixed terminal-state graph evaluator can be instantiated either manually or with LLM assistance. The evaluator provides discriminative trajectory-quality scores within each outcome class while preserving success-failure separation. Uniform discount-aware relabeling then propagates this quality-derived terminal reward to preceding transitions without changing the induced policy ordering. Progressive prior-guided co-training integrates limited suboptimal priors with online experience while gradually reducing prior dependence. Across discrete-action SMAC v1/v2 and continuous-action MPE benchmarks, TRACE achieves the best overall training efficiency and performance. Further results show improved value discrimination, and robustness to prior quantity.

1 INTRODUCTION

Multi-agent reinforcement learning (MARL) has achieved substantial progress in intelligent transportation, cooperative robotics, energy systems, and networked resource scheduling (Feng et al., 2023; Heuthe et al., 2024; Ma et al., 2024). However, designing dense, generalizable process-level rewards becomes increasingly costly as task complexity and agent interactions grow (Mu et al., 2024). In many practical systems, only a sparse two-class team reward is available, with $+1$ for success and -1 for failure, together with a few suboptimal trajectories collected from early logs, human demonstrations, or existing controllers (Hoang et al., 2024). We study how to learn sample-efficient cooperative policies in this low-data, sparse-feedback setting.

For this learning problem, adversarial imitation learning constructs discriminator-based rewards from demonstrations (Ho & Ermon, 2016; Song et al., 2018; Wang et al., 2023; Yu et al., 2025), but may introduce preferences inconsistent with the original objective and typically requires sufficient demonstration coverage. Offline-to-online MARL transfers pretrained policies or value functions (Jiang & Lu, 2023; Zhong et al., 2025), yet a few suboptimal trajectories may yield unreliable initialization, while subsequent learning still relies on binary feedback. Exploration-based MARL improves state-action coverage and successful experience acquisition (Liu et al., 2023; Li et al., 2024), but the original binary reward still provides no direct supervision for distinguishing trajectories sharing the same outcome.

Binary feedback can therefore assign identical values to joint actions whose resulting trajectories differ substantially in within-outcome quality. This many-to-one mapping removes the supervision required for discriminative value estimation, and our diagnosis shows that representative meth-

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.

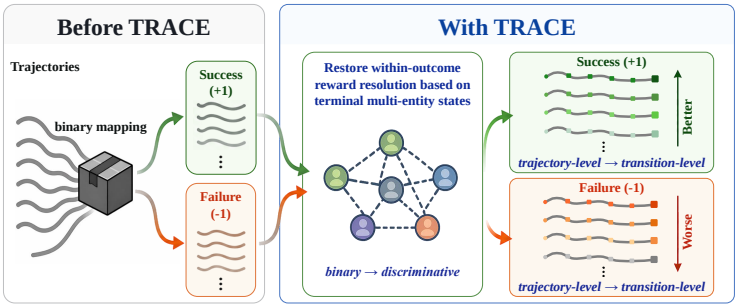


Figure 1: TRACE restores within-outcome trajectory-quality distinctions obscured by binary team rewards using a fixed evaluator over terminal multi-entity states, and propagates the resulting trajectory-level supervision into transition-level learning signals.

ods struggle to recover the lost distinctions. We term this bottleneck *intra-outcome-class reward-resolution collapse*.

Because the binary mapping discards within-outcome information, the lost distinctions are not identifiable from binary labels alone. We use only information already associated with the learning problem: the meaning of the sparse binary reward and the available terminal-state variables. Given this limited information and a common evaluator template, a fixed trajectory-quality evaluator can be instantiated either manually or with LLM assistance. The evaluator only needs to provide a coarse ordering among trajectories within the same outcome class.

To this end, we propose TRACE, short for trajectory-quality recovery and reward relabeling with prior-guided co-training for efficient MARL. TRACE first instantiates a fixed terminal multi-entity graph evaluator under the proposed template and uses it to assign discriminative trajectory-quality scores within each binary outcome class while preserving success-failure separation. It then applies uniform discount-aware relabeling to propagate the resulting quality-derived terminal reward to preceding transitions without changing the induced policy ordering. Finally, progressive prior-guided co-training integrates a few suboptimal trajectories with online experience and gradually reduces prior dependence.

The main contributions are summarized as follows:

- We identify intra-outcome-class reward-resolution collapse as a distinct bottleneck under sparse binary team rewards. Theoretically, binary feedback can assign identical values to actions with different within-outcome trajectory qualities. Empirically, representative methods struggle to recover these lost distinctions.
- We propose TRACE, which uses only the meaning of the original binary reward and terminal-state variables to instantiate a fixed evaluator for within-outcome trajectory-quality discrimination. Policy-ordering-preserving relabeling and progressive prior-guided co-training then propagate and exploit this supervision.
- Extensive experiments on discrete-action SMAC v1/v2 and continuous-action MPE demonstrate that TRACE achieves strong overall training efficiency and performance over representative baselines. Further analyses show improved within-outcome value discrimination and robustness to the number of prior trajectories.

2 Related Work

Adversarial Imitation Learning. Adversarial imitation learning (AIL) derives policy-optimization rewards from discriminators trained on demonstrations and learner behaviors. Multi-agent extensions include MAGAIL (Song et al., 2018), MA-AIRL (Yu et al., 2019), and IGASIL (Hao et al., 2019). DM^2 (Wang et al., 2023) decomposes joint distribution matching into agent-wise objectives, while PegMARL (Yu et al., 2025) learns cooperation from personalized demonstrations through dual discriminators. These methods provide process-level supervision but depend on demonstration quality and coverage, and do not explicitly distinguish trajectories sharing the same binary outcome. In contrast, TRACE uses a few suboptimal demonstrations only as weak guidance and jointly exploits prior and online experience to recover discriminative trajectory rewards without behavior matching.

Offline-to-Online MARL. Offline-to-online MARL transfers policies, value functions, or data from offline training to online fine-tuning. Online Tuning for Offline Decentralized MARL (Jiang & Lu, 2023) prioritizes offline transitions according to their similarity to online samples, while Selective Reincarnation (Formanek et al., 2023) transfers selected pretrained agents. OVMSE (Zhong et al., 2025) retains an offline value network for online target estimation and combines it with sequential exploration. Unlike these methods, TRACE randomly initializes the online learner and uses prior trajectories only as weak guidance.

Exploration-Based MARL. Exploration-based MARL introduces intrinsic objectives to improve state-action coverage under sparse rewards. Lazy Agents (Liu et al., 2023) strengthens individual incentives under weak team feedback. Related methods use influence-aware credit assignment (Han et al., 2025), contribution-based exploration (Li et al., 2024), self-supervised goal reaching (Nimonkar et al., 2025), accumulated entropy rewards (He et al., 2024), or human-informed intrinsic objectives (Wu et al., 2025). While these methods improve experience acquisition, trajectories with the same binary outcome still receive identical environment returns. TRACE instead directly restores intra-outcome reward resolution and converts the recovered quality into denser learning signals.

3 PRELIMINARIES

We formulate cooperative MARL as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Bernstein et al., 2013), $\langle N, \mathcal{S}, \mathcal{A}, P, R, Z, \mathcal{O}, \gamma, H \rangle$. Here, N is the number of agents, $s_t \in \mathcal{S}$ is the global state, and $\boldsymbol{a}_t = (a_t^1, \ldots, a_t^N) \in \mathcal{A}$ is the joint action, with $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_N$. The transition and reward functions are $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ and $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, respectively. Under partial observability, agent i selects actions according to $\pi_i(a_i^t \mid \tau_i^t)$, where $\tau_i^t = (o_i^0, a_i^0, \ldots, o_i^t)$ is its local action–observation history generated by the observation function Z over the observation space $\mathcal{O}$. The joint policy is $\boldsymbol{\pi} = \{\pi_i\}_{i=1}^N$. Given $\gamma \in (0, 1]$ and horizon H, the objective is $J(\boldsymbol{\pi}) = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{t=0}^{H-1} \gamma^t R_{t+1} \right]$. The state-joint-action value function under $\boldsymbol{\pi}$ is defined as $Q^{\boldsymbol{\pi}}(s_t, \boldsymbol{a}_t) = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{k=t}^{H-1} \gamma^{k-t} R_{k+1} \,\middle|\, s_t, \boldsymbol{a}_t \right]$.

4 RETHINKING LEARNING SIGNALS UNDER SPARSE BINARY TEAM REWARDS

Existing methods mitigate sparse-reward MARL through auxiliary rewards, warm-started initialization, or enhanced exploration, but do not directly restore trajectory-quality distinctions obscured by binary outcomes. We identify this unresolved bottleneck as *intra-outcome-class reward-resolution collapse* and examine it theoretically and empirically.

Intra-Outcome-Class Reward-Resolution Collapse. Let $G_t(\tau) = \gamma^{H-t-1} y(\tau)$ denote the discounted binary return of trajectory τ, where $y(\tau) \in \{-1, 1\}$ is its outcome. Because $G_t(\tau)$ depends only on the outcome and terminal time, trajectories with different task-relevant qualities can receive identical returns.

Consider two joint actions $\boldsymbol{a}$ and $\bar{\boldsymbol{a}}$ at the same state s_t under a common continuation policy. If $\mathbb{E}[G_t(\tau) \mid s_t, \boldsymbol{a}] = \mathbb{E}[G_t(\tau) \mid s_t, \bar{\boldsymbol{a}}]$, then $Q(s_t, \boldsymbol{a}) = Q(s_t, \bar{\boldsymbol{a}})$, even when their class-conditional expected trajectory qualities differ, i.e., $\Delta_c^q \neq 0$ for some nonempty outcome class $c \in \{-1, 1\}$. Binary feedback therefore provides no direct supervision for recovering this within-outcome distinction. We call this many-to-one loss of reward resolution *intra-outcome-class reward-resolution collapse*. Formal definitions and proofs are provided in Appendix D.

4.1 EMPIRICAL DIAGNOSIS

We focus our empirical diagnosis on failed trajectories, for which an external post-hoc quality reference is available. Successful trajectories are excluded because they share the same dense episode return in SMAC and thus provide no measurable within-success variation. On the super-hard 6h_vs_8z task, we examine whether failed trajectories receiving the same binary reward exhibit meaningful quality variation, whether learned values recover the resulting ordering, and whether this recovery is reflected in online performance.

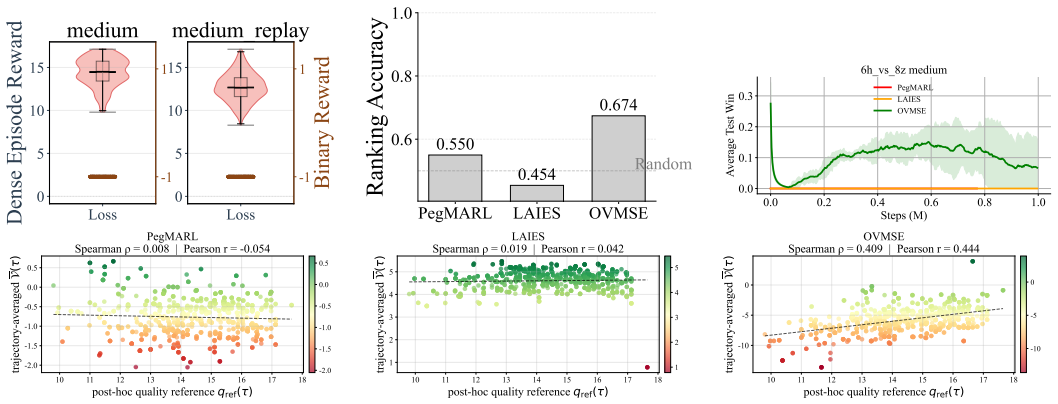


Figure 2: Diagnosis of within-failure reward resolution and value-quality alignment on 6h_vs_8z. Top row: post-hoc trajectory-quality variation, pairwise ranking accuracy, and online performance. Bottom row: value-quality alignment of PegMARL, LAIES, and OVMSE.

Hidden Within-Failure Quality Variation. We analyze failed trajectories from the CFCQL medium and medium-replay datasets. The dense episode return is used only as an unavailable post-hoc quality reference: $q_{\text{ref}}(\tau) = \sum_{t=0}^{H-1} r_{t+1}^{\text{dense}}$. It is never observed during training. As shown in Fig. 2, failed trajectories exhibit substantial variation in $q_{\text{ref}}(\tau)$ despite receiving the same binary reward of -1, confirming hidden within-failure quality differences.

Within-Failure Value-Quality Alignment. We evaluate PegMARL, LAIES, and OVMSE as representative AIL-based, exploration-based, and offline-to-online MARL methods. For each failed trajectory τ, we compute $\overline{\mathcal{V}}(\tau)$ by averaging the estimated values over its transitions, and measure its Spearman and Pearson correlations with $q_{\text{ref}}(\tau)$. We also sample $M = 500$ failed-trajectory pairs with distinct reference qualities and report the fraction whose ordering under $\overline{\mathcal{V}}$ agrees with that under q_{ref}. A score of 0.5 corresponds to random ordering.

As shown in Fig. 2, PegMARL and LAIES exhibit negligible correlations and near- or below-random ranking accuracy. OVMSE achieves moderate positive alignment, but still misranks nearly one-third of the sampled pairs. Representative methods therefore recover the hidden within-outcome quality structure only weakly or partially. The recovered ordering is qualitatively consistent with online performance. PegMARL and LAIES remain near zero win rate, whereas OVMSE achieves limited but unstable improvement alongside its more informative, though imperfect, within-failure ordering. These findings suggest that effective learning requires restoring meaningful within-outcome quality distinctions, rather than reconstructing a complete dense reward function. Such distinctions provide more discriminative supervision.

5 Method

5.1 Problem Setting

We consider cooperative tasks with sparse binary team rewards, where a trajectory τ receives $+1$ upon success and -1 upon failure, only at episode termination. In addition to online interaction, we assume access to a small prior dataset $\mathcal{D}_{\text{prior}} = \{\tau_{\text{p}}^{(k)}\}_{k=1}^{K}$ generated by a suboptimal policy satisfying $J(\pi_{\text{rand}}) < J(\pi_{\text{prior}}) < J(\pi^{\star})$. We focus on a low-data regime with only a few dozen prior trajectories, whose transitions are far fewer than the online interaction budget.

5.2 Overall Framework

As illustrated in Fig. 3, TRACE contains three modules. First, graph-based trajectory-quality recovery instantiates a fixed terminal multi-entity graph evaluator from the meaning of the binary reward and available terminal-state variables, and uses it to construct discriminative trajectory-quality scores within each binary outcome class. Second, policy-ordering-preserving relabeling propagates each

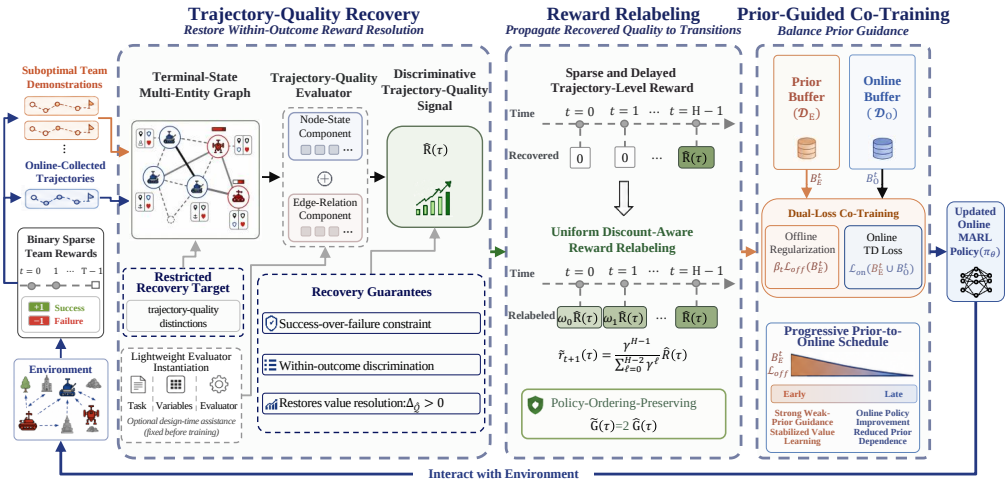


Figure 3: Illustration of TRACE's three-module framework.

recovered trajectory-quality score to its preceding transitions. Third, prior-guided co-training combines limited prior trajectories with online experience and progressively reduces prior dependence.

5.3 Terminal-State Graph-Based Trajectory-Quality Recovery

TRACE separates the recovery problem into two restricted questions: where the lost within-outcome distinctions remain observable and how they can be converted into useful learning supervision. Rather than defining rewards over arbitrary intermediate states and joint actions, TRACE constructs a scalar trajectory-quality score from the terminal multi-entity state using a fixed evaluator, and subsequently propagates this terminal supervision to the transitions that produced that state.

Binary outcomes preserve the success-failure distinction but remove quality differences within each outcome class. We assume that terminal states retain information about these differences: $I(q(\tau); s_T(\tau) \mid y(\tau) = c) > 0$, for each nonempty outcome class c, where $q(\tau)$ denotes task-consistent trajectory quality, and $s_T(\tau)$ denotes the terminal state.

We represent the terminal state as a weighted attributed graph $\mathcal{G}_T(\tau) = (X_T, W_T)$. Its node features describe terminal entity conditions, while its edge features describe inter-entity relations. The evaluator maps these structured terminal variables to a scalar trajectory-quality score. Specifically, its node-feature matrix is $\boldsymbol{X}_T = [\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_{N_T}^\top]^\top$, where each node $\boldsymbol{x}_i = \xi_v(s_T(\tau), i)$. The relation matrix is $\boldsymbol{W}_T = [w_{ij}]_{i,j=1}^{N_T}, w_{ij} = \xi_e\left(s_T(\tau), \boldsymbol{x}_i, \boldsymbol{x}_j\right)$.

The evaluator assigns a scalar recovered trajectory-quality score

$$\widehat{q}(\tau) = \kappa_{y(\tau)}\left[f_V(X_T) + 0.1 * f_E(W_T)\right], \tag{1}$$

where $f_V(\boldsymbol{X}_T) > 0$ and $f_E(\boldsymbol{W}_T) > 0$ aggregate node and relation information, respectively. The resulting score is used as the terminal reward, $\widehat{r}_H(\tau) = \widehat{q}(\tau)$, while all preceding rewards remain zero before relabeling. The outcome-conditioned mapping $\kappa_{y(\tau)}$ preserves the original success-failure separation while allowing trajectories within the same outcome class to receive different scores. We use $\kappa_1(z) = z$ and $\kappa_{-1}(z) = -2z$. The mappings ξ_v, ξ_e, f_V, and f_E are fixed deterministic functions instantiated from available terminal-state under the proposed evaluator template. In SMAC, node terms summarize terminal unit conditions such as health, while edge terms summarize pairwise spatial relations. The resulting $\widehat{R}(\tau)$ is a constructed trajectory reward for value learning, rather than an estimate of a uniquely defined latent or process-level reward function.

For an evaluator satisfying outcome separation, and within-outcome consistency, $\widehat{R}(\tau)$ restores a value distinction removed by binary feedback. For the joint actions $\boldsymbol{a}$ and $\bar{\boldsymbol{a}}$ considered in Sec. 4, let p_c denote their common probability of outcome c, and let $\widehat{\Delta}_c$ denote the difference in their class-conditional expected discounted recovered rewards. A sufficient condition for the recovered rewards to distinguish the two actions is $\widehat{\Delta}_c \geq 0$ for every nonempty outcome class, with a strict inequality for at least one class. Under this condition, $\widehat{\Delta}_Q = \widehat{Q}(s_t, \boldsymbol{a}) - \widehat{Q}(s_t, \bar{\boldsymbol{a}}) = \sum_{c \in \{-1,1\}} p_c \widehat{\Delta}_c > 0$.

This observation suggests that directionally consistent within-outcome rewards can introduce a value distinction between joint actions associated with different task-consistent trajectory qualities. If each learned action value deviates from its recovered-reward value by at most ϵ_Q, this distinction is preserved whenever $\epsilon_Q < \widehat{\Delta}_Q/2$. See Appendix E for details.

Lightweight Evaluator Instantiation. The evaluator is parameterized by $\Omega = (\xi_v, \xi_e, f_V, f_E)$ and instantiated once before policy training. Under the proposed template, evaluator construction uses the meaning of the original binary reward and the available terminal-state variables. It can be instantiated either manually or with LLM assistance. In our experiments, the LLM is provided with these two sources of information together with the evaluator template, and proposes deterministic node features, pairwise relation functions, and aggregation rules. No dense environment reward, trajectory-quality annotation, expert trajectory ranking, or intermediate-state/action preference is provided.

Programmatic checks verify variable availability and numerical validity, and the resulting deterministic rules are verified before being converted into executable functions. Once instantiated, the evaluator is fixed throughout policy learning, and the LLM is never queried during training. A single evaluator is reused across all SMAC v1/v2 maps, and another is reused across all MPE Simple Spread settings, without map-specific adaptation, or prior-specific modification. Thus, the LLM serves only as an instantiation tool within the proposed evaluator template; manual construction follows the same information interface. Full prompts and formulas are provided in Appendix B.3.

5.4 Policy-Ordering-Preserving Reward Relabeling

Although $\widehat{R}(\tau)$ distinguishes trajectories within each outcome class, it remains sparse and delayed. We therefore propagate it to preceding transitions while preserving the policy ordering induced by the recovered objective.

Following the standard episodic replay protocol in MARL, complete trajectories are collected before policy updates. The recovered reward can therefore be computed at termination and relabeled over the stored transitions before replay, changing only their learning targets without altering the environment dynamics or behavior policy.

For a trajectory $\tau = (s_0, \boldsymbol{a}_0, \ldots, s_H)$ with $H > 1$ transitions, we retain $\widehat{R}(\tau)$ at the terminal transition and assign $\widetilde{r}_{t+1}(\tau) = \frac{\gamma^{H-1}}{\sum_{\ell=0}^{H-2}\gamma^{\ell}}\widehat{R}(\tau)$, to every preceding transition. Its total discounted contribution is $\sum_{t=0}^{H-2}\gamma^t\widetilde{r}_{t+1}(\tau) = \gamma^{H-1}\widehat{R}(\tau)$. Together with the retained terminal reward, the relabeled return becomes $\widetilde{G}(\tau) = 2\gamma^{H-1}\widehat{R}(\tau) = 2\widehat{G}(\tau)$, where $\widehat{G}(\tau) = \gamma^{H-1}\widehat{R}(\tau)$.

Thus, relabeling applies the same positive scaling to every recovered trajectory return:

$$\widetilde{J}(\pi) = 2\widehat{J}(\pi), \arg\max_{\pi}\widetilde{J}(\pi) = \arg\max_{\pi}\widehat{J}(\pi). \tag{2}$$

It therefore introduces no additional change to the policy ordering induced by the recovered objective while providing direct supervision at every transition. See Appendix E.3 for details.

5.5 Prior-Guided Co-Training

The recovered and relabeled rewards improve supervision, but a randomly initialized learner may still rarely encounter informative trajectories. TRACE therefore uses a few suboptimal prior trajectories to provide early guidance.

TRACE maintains a prior buffer $\mathcal{D}_{\mathrm{E}}$ and an online buffer $\mathcal{D}_{\mathrm{O}}$. The former is initialized with prior trajectories and retains successful online trajectories whose recovered rewards exceed the buffer minimum. The remaining online trajectories are stored in $\mathcal{D}_{\mathrm{O}}$. At update step t, the coefficient $\beta_t = \beta_{\mathrm{start}}\max\left(0, 1 - \frac{t}{T_{\mathrm{d}}}\right)$ progressively reduces prior-buffer sampling and offline regularization. The training objective is $\mathcal{L}_{\mathrm{total}}(t) = \mathcal{L}_{\mathrm{on}}\left(\mathcal{B}_{\mathrm{E}}^t \cup \mathcal{B}_{\mathrm{O}}^t\right) + \beta_t\mathcal{L}_{\mathrm{off}}\left(\mathcal{B}_{\mathrm{E}}^t\right)$, where $\mathcal{B}_{\mathrm{E}}^t$ and $\mathcal{B}_{\mathrm{O}}^t$ are sampled from the two buffers.

For a value-based MARL backbone, the online temporal-difference (TD) loss is $\mathcal{L}_{\mathrm{on}} = \mathbb{E}\left[\left(Q_{\mathrm{tot}}(s, \boldsymbol{a}) - \widetilde{r} - \gamma\max_{\boldsymbol{a}'}\overline{Q}_{\mathrm{tot}}(s', \boldsymbol{a}')\right)^2\right]$, where $\overline{Q}_{\mathrm{tot}}$ is the target Q_{tot}.

To stabilize learning on the limited prior-buffer distribution, we adopt the counterfactual conservative regularizer (Shao et al., 2023): $\mathcal{L}_{\text{off}} = \mathbb{E}_{s\sim\mathcal{B}_{\text{E}}^{t}}\Big[\sum_{i=1}^{N}\lambda_i(s)\mathbb{E}_{\boldsymbol{a}^{-i}\sim\boldsymbol{\eta}^{-i}}\big[\log\sum_{a^i}\exp Q_{\text{tot}}\big(s,(a^i,\boldsymbol{a}^{-i})\big)\big] - \mathbb{E}_{\boldsymbol{a}\sim\boldsymbol{\eta}}\left[Q_{\text{tot}}(s,\boldsymbol{a})\right]\Big]$, where $\boldsymbol{\eta}$ is the behavior policy estimated from prior-buffer samples, and $\lambda_i(s) = \frac{\exp\big(\kappa\mathbb{E}_{\pi_i}[\pi_i(s)/\eta_i(s)]\big)}{\sum_{j=1}^{N}\exp\big(\kappa\mathbb{E}_{\pi_j}[\pi_j(s)/\eta_j(s)]\big)}$. The first term suppresses unsupported counterfactual values, while the second anchors value estimates to prior-buffer actions. Overall, prior-guided co-training stabilizes early learning with limited suboptimal data and gradually shifts optimization toward online experience, enabling improvement beyond the prior.

6 Experiments

6.1 Baselines and Experimental Setup

We compare TRACE with six representative baselines: the online value-factorization method ft-QMIX (Hu et al., 2023), the AIL-based PegMARL (Yu et al., 2025), the exploration-based LAIES (Liu et al., 2023), and three offline-to-online methods, OVMSE (Zhong et al., 2025), CFCQL (Shao et al., 2023), and MACal-QL (Nakamoto et al., 2023). CFCQL and MACal-QL retain their offline objectives during online fine-tuning, whereas OVMSE switches from a CFCQL-style offline loss to the ft-QMIX online objective.

TRACE uses one fixed trajectory-quality evaluator for all StarCraft Multi-Agent Challenge (SMAC) v1/v2 maps (Samvelyan et al., 2019; Ellis et al., 2023) and another for all Multi-Agent Particle Environment (MPE) Simple Spread settings (Peng et al., 2021). Each evaluator is instantiated once before policy training using only the meaning of the original binary reward and available terminal-state variables under the proposed evaluator template. No map-specific evaluator is designed, and no component or weight is tuned using map-specific downstream policy performance. Full implementation details are provided in Appendix B.3.

We evaluate all methods on three SMAC v1 maps, 5m_vs_6m, 6h_vs_8z, and MMM2, and three SMAC v2 scenarios, Protoss_5v5, Zerg_5v5, and Terran_5v5, under both medium and medium-replay settings. Each task contains only 50 suboptimal team trajectories. All methods use the same prior trajectories, online interaction budget, and evaluation protocol, and results are averaged over five independent seeds.

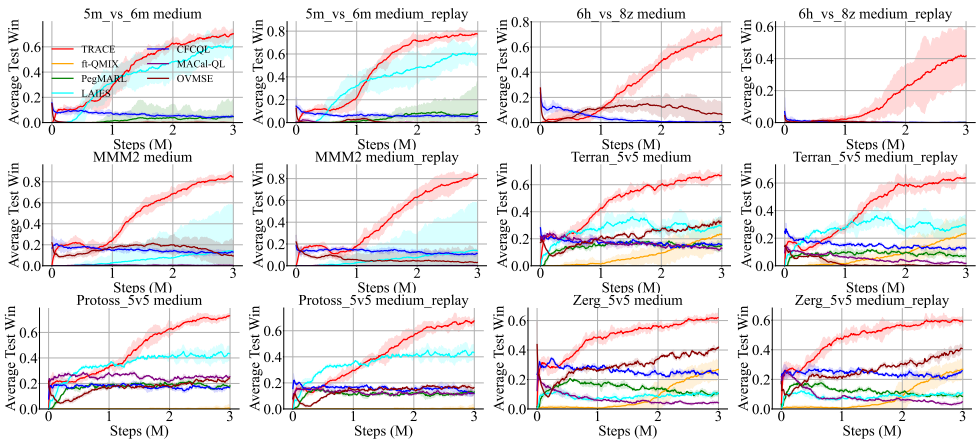


Figure 4: Comparison of TRACE with ft-QMIX and SOTA baselines across various maps. Shaded regions indicate the minimum-to-maximum range across runs.

6.2 Overall Results

As shown in Fig. 4, TRACE achieves the best overall training efficiency and performance under both prior settings. Its advantages are particularly pronounced on 6h_vs_8z, MMM2, and the more challenging SMAC v2 scenarios, where most baselines either remain at low win rates or plateau early. The results also highlight the difficulty of learning from very limited suboptimal data under sparse binary feedback. PegMARL and LAIES achieve limited performance on several challenging tasks, while the offline-to-online methods show limited or unstable improvement in this low-data setting. The online ft-QMIX baseline achieves near-zero win rates on most challenging tasks. As shown in Appendix C.1, TRACE achieves the highest win-rate AUC across all evaluated tasks and consistently maintains a substantial advantage over the representative baselines under both medium and medium-replay settings. Appendix C.3 further shows that TRACE is competitive with the privileged dense-reward reference ft-QMIX-DR and surpasses it on several tasks.

6.3 Value-Function Discrimination

We evaluate whether trajectory-quality recovery improves the alignment between trajectory-averaged Q_{tot} and the post-hoc quality reference $q_{\mathrm{ref}}(\tau)$ within the failure class on 5m_vs_6m-medium and 6h_vs_8z-medium. As shown in Fig. 5, removing trajectory-quality recovery produces weak or negative value-quality correlations. With recovery, both Spearman and Pearson correlations increase substantially, indicating that the learned values better distinguish trajectories sharing the same binary outcome. This result supports improved within-outcome value discrimination.

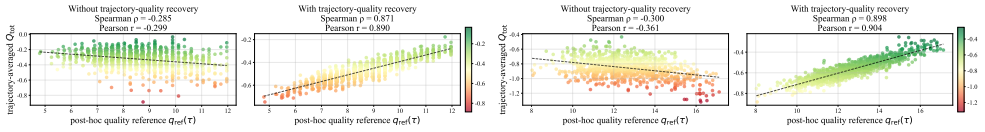


Figure 5: Relationship between trajectory-averaged Q_{tot} and the post-hoc quality reference $q_{\mathrm{ref}}(\tau)$ on 5m_vs_6m-medium (left) and 6h_vs_8z-medium (right).

6.4 Ablation Studies

Module ablation. We compare TRACE with three leave-one-module-out variants: TRACE-noTQR, TRACE-noRelabel, and TRACE-noCoTrain. We also include TRACE-TQR-Terminal, which retains the same evaluator-constructed terminal reward but removes relabeling and prior-guided co-training, to test whether terminal reward construction alone explains TRACE's gains. As shown in Fig. 6, TRACE-noTQR remains near zero win rate, while removing relabeling or co-training also substantially degrades performance. TRACE-TQR-Terminal likewise remains near zero despite using the same terminal reward as full TRACE, showing that sparse and delayed terminal supervision alone is difficult to exploit. Together with Fig. 5, these results show that trajectory-quality recovery enriches the learning signal, while relabeling and prior-guided co-training are critical for translating it into policy improvement.

Trajectory-quality evaluator analysis. We investigate how different trajectory-quality evaluator designs affect learning performance. TRACE-SimpleTQR uses the average health of surviving units, TRACE-RandomTQR assigns random within-class scores while preserving success-failure separation, and TRACE-noTQR retains the original binary reward. As shown in Fig. 6, TRACE-SimpleTQR substantially outperforms TRACE-noTQR, showing that a coarse trajectory-quality evaluator can provide substantial learning benefits. The full graph evaluator performs consistently better and yields more discriminative signals. TRACE-RandomTQR, however, remains close to TRACE-noTQR: random within-class variation does not recover meaningful trajectory-quality distinctions and thus leaves the intra-outcome-class reward-resolution collapse essentially unresolved. Overall, alleviating this collapse is critical for improving training efficiency and performance.

6.5 Effect of Prior-Trajectory Quantity

We examine how TRACE depends on the number of available suboptimal prior trajectories. Fig. 7 varies the prior size over $n \in \{0, 10, 30, 40, 50\}$ across four representative SMAC v1/v2 tasks.

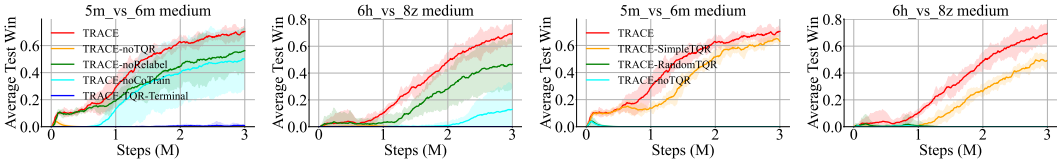


Figure 6: Ablation studies of TRACE. Left two: ablation of the main TRACE modules. Right two: trajectory-quality evaluator analysis.

Without prior trajectories, performance degrades substantially. With only ten trajectories, TRACE already approaches its full-data performance with a modest early-stage slowdown, while results stabilize when $n \geq 30$, showing that only a few dozen suboptimal priors are sufficient. Appendix C.4 shows that TRACE is robust to the co-training hyperparameters β_{start} and T_{d}, respectively.

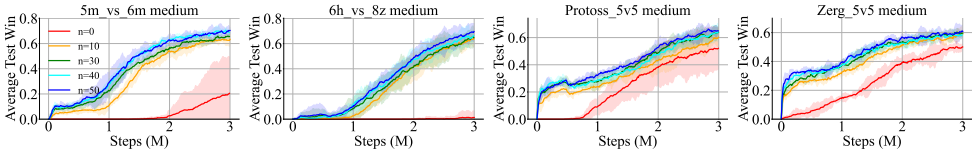


Figure 7: Sensitivity to the number of prior trajectories.

6.6 Generalization to Continuous-Action MARL

We evaluate TRACE on continuous-action MPE Simple Spread with $N \in \{2, 3, 4\}$ agents and N landmarks under the binary sparse team-reward setting defined in Sec. 5.1. MATD3 replaces ft-QMIX as the online backbone and is also used by TRACE and the applicable baselines. Medium and medium-replay priors follow the same construction protocol as in SMAC.

Figure 8 and Appendix C.2 show that TRACE consistently outperforms all baselines across the six tasks. Most competing methods remain near zero success rate as the task scales to three and four agents, whereas TRACE continues to learn effective cooperative policies. These results show that TRACE's effectiveness extends beyond discrete-action SMAC to continuous-action MARL.

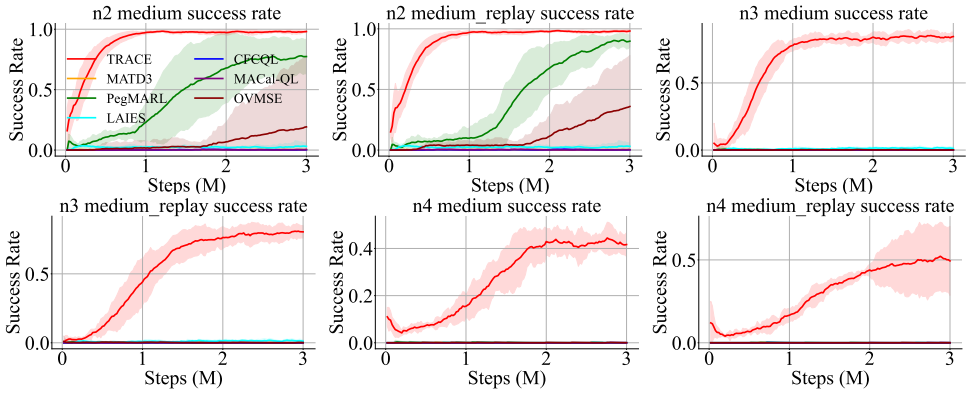


Figure 8: Comparison of TRACE with MATD3 and SOTA baselines on MPE.

7 Conclusion

We identified intra-outcome-class reward-resolution collapse as a key bottleneck in cooperative MARL with sparse binary team rewards. TRACE addresses it by constructing coarse within-outcome trajectory-quality supervision from the meaning of the original binary-reward and available terminal-state, and making this supervision effective through policy-ordering-preserving relabeling and progressive prior-guided co-training. Experiments on SMAC v1/v2 and continuous-action MPE demonstrate strong training efficiency and performance, improved value discrimination, and robustness with only a few suboptimal prior trajectories.

AI USE STATEMENT

Generative AI tools were primarily used to assist with manuscript formatting, language polishing, and code-related tasks, including code analysis, debugging, and implementation support, as well as improving grammar, clarity, concision, and LaTeX presentation. In addition, as explicitly described in the methodology, an LLM was used at design time to assist in instantiating the trajectory-quality evaluator under the proposed template. All AI-assisted outputs, including code, were manually reviewed, verified, and tested by the authors. The authors take full responsibility for the final content of this work, including its technical claims, experimental results, code, text, and artifacts.

ETHICS STATEMENT

This study constitutes algorithmic research conducted entirely in simulated environments, involving no experiments on human or animal subjects and no datasets containing private, or sensitive information. All experiments were performed on standard multi-agent simulation benchmarks, including SMAC v1/v2 and MPE Simple Spread. The suboptimal prior trajectories used in our experiments were generated within these simulation environments and contain no real-world personal data.

Multi-agent reinforcement learning has broad potential for applications such as robotic collaboration, autonomous systems, and distributed decision-making. This work focuses on improving learning efficiency under sparse binary team rewards with only a few suboptimal prior trajectories, and does not directly involve deployment in safety-critical or high-risk real-world settings. Nevertheless, methods that improve autonomous decision-making may have broader downstream implications when transferred to practical systems. Any real-world deployment should therefore undergo application-specific validation of safety, reliability, robustness, and fairness, with appropriate human oversight where necessary, to reduce the risk of unintended or harmful behavior.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility of TRACE and its evaluation under sparse binary team rewards with only a few suboptimal prior trajectories, we have anonymously released our code and experimental implementation[1].

The appendix describes the offline dataset construction, experimental setup, baseline implementations, TRACE implementation details, hyperparameters, software dependencies, additional results, sensitivity analyses, and complete theoretical derivations used in this work. This ensures that other researchers can replicate this study under identical conditions. We sincerely encourage academic peers to utilize these resources in broader multi-agent learning scenarios to further validate and extend the findings of this research.

REFERENCES

Daniel S. Bernstein, Shlomo Zilberstein, and Neil Immerman. The complexity of decentralized control of markov decision processes. *CoRR*, abs/1301.3836, 2013.

Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob N. Foerster, and Shimon Whiteson. SMACv2: an improved benchmark for cooperative multi-agent reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Shuo Feng, Haowei Sun, Xintao Yan, Haojie Zhu, Zhengxia Zou, Shengyin Shen, and Henry X Liu. Dense reinforcement learning for safety validation of autonomous vehicles. *Nature*, 615(7953): 620–627, 2023.

Claude Formanek, Callum Rhys Tilbury, Jonathan Shock, Kale-ab Tessera, and Arnu Pretorius. Selective reincarnation: Offline-to-online multi-agent reinforcement learning. In *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*, 2023.

[1] https://anonymous.4open.science/r/TRACE-rl

Shuai Han, Mehdi Dastani, and Shihan Wang. Credit assignment and efficient exploration based on influence scope in multi-agent reinforcement learning. *ArXiv*, abs/2505.08630, 2025.

Xiaotian Hao, Weixun Wang, Jianye Hao, and Yaodong Yang. Independent generative adversarial self-imitation learning in cooperative multiagent systems. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1315–1323, 2019.

Xin He, Hongwei Ge, Yaqing Hou, and Jincheng Yu. Saeir: Sequentially accumulated entropy intrinsic reward for cooperative multi-agent reinforcement learning with sparse reward. In *IJCAI*, pp. 4107–4115, 2024.

Veit-Lorenz Heuthe, Emanuele Panizon, Hongri Gu, and Clemens Bechinger. Counterfactual rewards promote collective transport using individually controlled swarm microrobots. *Science Robotics*, 9(97):eado5888, 2024.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Huy Hoang, Tien Mai, and Pradeep Varakantham. Sprinql: Sub-optimal demonstrations driven offline imitation learning. *Advances in Neural Information Processing Systems*, 37:136837–136872, 2024.

Jian Hu, Siying Wang, Siyang Jiang, and Musk Wang. Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning. In *The Second Blogpost Track at ICLR 2023*, 2023.

Jiechuan Jiang and Zongqing Lu. Online tuning for offline decentralized multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 8050–8059, 2023.

Xinran Li, Zifan Liu, Shibo Chen, and Jun Zhang. Individual contributions as intrinsic exploration scaffolds for multi-agent reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 28387–28402, 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.

Chengdong Ma, Aming Li, Yali Du, Hao Dong, and Yaodong Yang. Efficient and scalable reinforcement learning for large-scale network control. *Nature Machine Intelligence*, 6(9):1006–1020, 2024.

Tongzhou Mu, Minghua Liu, and Hao Su. Drs: Learning reusable dense rewards for multi-stage tasks. In *The Twelfth International Conference on Learning Representations*, 2024.

Mitsuhiko Nakamoto, Simon Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *Advances in Neural Information Processing Systems*, 36:62244–62269, 2023.

Chirayu Nimonkar, Shlok Shah, Catherine Ji, and Benjamin Eysenbach. Self-supervised goal-reaching results in multi-agent cooperation and exploration. *arXiv preprint arXiv:2509.10656*, 2025.

Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Boehmer, and Shimon Whiteson. FACMAC: Factored multi-agent centralised policy gradients. In *Advances in Neural Information Processing Systems*, volume 34, pp. 12208–12221. Curran Associates, Inc., 2021.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2186–2188, 2019.

Jianzhun Shao, Yun Qu, Chen Chen, Hongchang Zhang, and Xiangyang Ji. Counterfactual conservative q learning for offline multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 36:77290–77312, 2023.

Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. Multi-agent generative adversarial imitation learning. *Advances in neural information processing systems*, 31, 2018.

Caroline Wang, Ishan Durugkar, Elad Liebman, and Peter Stone. Dm^2: Decentralized multi-agent reinforcement learning via distribution matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11699–11707, 2023.

Xuefei Wu, Xiao Yin, Yuanyang Zhu, and Chunlin Chen. Learning individual intrinsic reward in multi-agent reinforcement learning via incorporating generalized human expertise. In *2025 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 4804–4811. IEEE, 2025.

Lantao Yu, Jiaming Song, and Stefano Ermon. Multi-agent adversarial inverse reinforcement learning. In *International conference on machine learning*, pp. 7194–7201. PMLR, 2019.

Peihong Yu, Manav Mishra, Alec Koppel, Carl Busart, Priya Narayan, Dinesh Manocha, Amrit Singh Bedi, and Pratap Tokekar. Beyond joint demonstrations: Personalized expert guidance for efficient multi-agent reinforcement learning. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856.

Hai Zhong, Xun Wang, Zhuoran Li, and Longbo Huang. Offline-to-online multi-agent reinforcement learning with offline value function memory and sequential exploration. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2025.

A Offline Datasets

To ensure the validity, reproducibility, and comparability of the experiments, we construct a diverse set of offline datasets across SMAC v1, v2, and MPE, covering different task complexities, agent scales, and action space dimensions. Detailed explanations are provided below.

A.1 Dataset Construction

We construct all datasets under a unified protocol to ensure fair comparisons. For 5m_vs_6m and 6h_vs_8z, we reuse the publicly released CFCQL datasets[2] and randomly sample 50 successful trajectories from each Medium and Medium_replay subset. For the remaining SMAC maps, we train ft-QMIX and follow the CFCQL collection protocol to construct datasets of the same size. For MPE, we follow the same protocol using MATD3 as the data-collection policy. Two dataset types are considered. **Medium** contains trajectories collected from a fixed policy with intermediate performance between random and near-optimal policies. **Medium_replay** contains trajectories sampled from the replay buffer accumulated from policy initialization until medium-level performance is reached.

B Implementation Details

B.1 Experimental Setup

Training Protocol. All methods share a unified training pipeline to ensure a fair comparison. Each training run proceeds for a fixed interaction budget measured in environment steps, with periodic evaluation episodes executed at regular intervals. Win-rate performance is reported based on these periodic evaluations. To leverage prior knowledge, a fixed set of offline trajectories is loaded once at the beginning of training. All methods use the same prior dataset for each task and data regime, and the online interaction budget is identical across methods.

Every experimental configuration is repeated over 5 independent runs with different random seeds. Seeds are neither manually fixed nor shared across methods, thereby reflecting realistic stochastic

[2] https://github.com/thu-rllab/CFCQL

variation in both environment dynamics and parameter initialization. We report the mean and standard deviation across these runs. All experiments are conducted on computing nodes equipped with a single NVIDIA GeForce RTX 5090 GPU.

B.1.1 DISCRETE-ACTION EXPERIMENTS (SMAC).

The discrete-action implementation of TRACE is built upon the pymarl3 codebase[3], which provides a modular and efficient research framework for multi-agent reinforcement learning. We evaluate TRACE on a set of challenging StarCraft II micromanagement maps, including 5m_vs_6m, 6h_vs_8z, MMM2, Protoss_5v5, Zerg_5v5, and Terran_5v5.

B.1.2 CONTINUOUS-ACTION EXPERIMENTS (MPE).

The continuous-action implementation is based on the HARL codebase[4], with the MPE environment upgraded to Simple Spread V3. To accommodate a sparse binary reward setting, we apply customized modifications to the environment. A task is considered successful if and only if all agents simultaneously cover their corresponding target landmarks (i.e., the distance between each agent and its assigned landmark is less than 0.2). Under this definition, the team receives a reward of $+1$ upon task success and -1 otherwise. We conduct experiments with $N \in \{2, 3, 4\}$ agents to validate the effectiveness of TRACE across varying levels of coordination complexity.

B.2 BASELINE ALGORITHMS

B.2.1 FT-QMIX.

We directly use the pymarl3[3] implementation of ft-QMIX, including all hyperparameters for the various SMAC v1 and v2 maps, strictly following recommended values of hyperparameters provided by the repository. For all methods built upon ft-QMIX, all hyperparameters (e.g., buffer size, learning rate, batch size, and optimizer) follow the ft-QMIX defaults unless explicitly modified by the corresponding method.

B.2.2 MATD3.

We directly employ the built-in MATD3 implementation from the HARL[4] framework, along with its default hyperparameter configurations across the MPE tasks of varying scales (N = 2, 3, 4). In this work, MATD3 serves as the online backbone for all methods operating in continuous action spaces. All MATD3-based approaches adhere to the default buffer size, learning rate, batch size, and optimizer settings, with modifications made only when explicitly required by the respective method.

B.2.3 LAIES.

On the SMAC benchmark, we directly employ the official source code[5] released by the original authors to conduct our experiments. For maps where the authors provided hyperparameter configurations, we strictly adhere to their original settings to reproduce the reported results. For maps without provided configurations, we perform a degree of hyperparameter tuning based on a thorough understanding of the underlying principles of LAIES, and report the best reproduced results. On the MPE benchmark, owing to the absence of a directly transferable official implementation, we reimplement LAIES upon the HARL codebase. Guided by the definition constraints of the external state specified in the original paper, we design the external state to consist of the distances between each agent and each target landmark, as well as the coverage status of each target landmark. On this basis, we similarly conduct a degree of hyperparameter tuning and report the best results obtained.

[3] https://github.com/tjuHaoXiaotian/pymarl3

[4] https://github.com/PKU-MARL/HARL

[5] https://github.com/Dawn0523/LAIES

B.2.4 PEGMARL.

The experimental procedures of PegMARL on both SMAC and MPE are consistent with those of LAIES: on SMAC, we employ its official source code[6] and strictly adhere to the hyperparameter configurations provided by the original authors; on MPE, we reimplement the algorithm based on the HARL codebase, referencing its source code released for the SMAC environment. In both settings, extensive hyperparameter tuning is conducted, and the best results are ultimately reported.

B.2.5 CFCQL.

The official codebase[2] of CFCQL provides implementations for both discrete and continuous action spaces. On SMAC, we directly adopt the original implementation, with moderate hyperparameter tuning only on maps for which no configurations are provided. On MPE, to facilitate a unified experimental pipeline, we integrate its continuous-action implementation into the HARL framework while preserving the core logic consistent with the original codebase. The training procedure of CFCQL comprises three stages. First, offline data are prepared and written into an offline buffer, based on which the behavior model is pre-trained prior to the onset of training. Subsequently, offline training is conducted. Upon completion of the offline phase, the online training phase commences, during which a new online buffer is instantiated to store trajectories sampled by the behavior policy. At each training iteration, an aggregated sample set is drawn from both the offline and online buffers and used for loss computation and parameter updates. Following common practice in O2O MARL, each training batch contains 30% offline samples and 70% online samples. The hyperparameters across all stages are carefully tuned, and the best results are reported.

B.2.6 MACAL-QL.

For MACal-QL, we referenced the source code of Cal-QL[7] and implemented it based on the MACQL method in CFCQL. First, we calculate the Monte Carlo return of the behavioral strategy as the value of the reference strategy, and then take the maximum value between the Q-value of the action sampled by the strategy and the value of the reference strategy. The offline-to-online design is consistent with that of CFCQL.

B.2.7 OVMSE.

The source code of OVMSE is not publicly available. We therefore reimplemented its two key components, OVM and SE, based on the formulas and pseudocode in the original paper. For the OVM module, we retain the offline pre-trained target value function and compute the target Q-value as the maximum of the offline memory value and the online TD target. The SE module is implemented by reducing the exploration probability proportionally to the number of agents. The transition from the offline to the online phase in our reimplementation follows the design principles described in the original OVMSE paper. During implementation, we found that the decay cycle of OVM has a noticeable impact on OVMSE's performance. Therefore, we tuned this hyperparameter over multiple candidate values and report the best-performing setting. In addition, on exploration-demanding tasks such as Corridor, OVMSE still attains near-zero win rates, likely because its SE module reduces exploration, which is detrimental when tackling tasks that require substantial exploration.

B.3 TRACE IMPLEMENTATION

This section elaborates on the concrete implementation of TRACE, covering the three core modules (trajectory-quality recovery, policy-ordering-preserving reward relabeling, and prior-guided cotraining) along with all relevant hyperparameters and training details.

B.3.1 TERMINAL-STATE GRAPH-BASED TRAJECTORY-QUALITY RECOVERY

LLM-Assisted Evaluator Instantiation. At design time, we use a three-step workflow to instantiate a deterministic evaluator under the proposed evaluator template.

[6] https://openreview.net/forum?id=kzPNHQ8ByY

[7] https://nakamotoo.github.io/Cal-QL

<!-- System Prompt -->

```
You are an expert in Multi-Agent Reinforcement Learning, specializing in designing interpretable trajectory quality evaluation rules based on variables from the terminal state.

Your task is to design a concrete instantiation of a "Terminal-State Graph-Based Trajectory-Quality Recovery" for a given multi-agent environment.

The framework is intended to extract trajectory quality information from the terminal state of each episode (i.e., to distinguish the quality of trajectories that share the same win/loss outcome), while strictly satisfying the success-over-failure constraint—namely, the recovered reward of every successful trajectory must be strictly greater than that of every unsuccessful trajectory.

You are required to output only four programmable symbolic components:
- Node Feature Extraction Function (ξ_v)
- Graph Edge Construction Function (ξ_e)
- Node State Aggregation Function (f_V)
- Edge Relation Aggregation Function (f_E)

All components must use only the explicitly provided state variables. In addition, the outputs of both f_V and f_E must always be positive.

Please provide the result in a structured JSON format, along with a brief explanation of each component.
```

Figure 9: System prompt specifying role, constraints, and required outputs for terminal-state graph-based evaluator generation.

<!-- User Prompt -->

```
### Task Description
The environment is SMAC. If all allied units are eliminated or the time limit is exceeded, the episode is considered a failure. If all enemy units are eliminated, the episode is considered a success.
This is a cooperative multi-agent task with sparse team rewards: success = +1, failure = -1, and all intermediate steps receive a reward of 0.

### Available Terminal-State Variables
For each unit (including both allied and enemy units), you may use the following fields:
- `unit_id`: Unique identifier
- `alliance`: 0 indicates an allied unit, and 1 indicates an enemy unit
- `health`: Current health (0 to maximum health, floating-point value)
- `health_max`: Maximum health
- `x`: x-coordinate (floating-point value)
- `y`: y-coordinate (floating-point value)

### Framework Requirements
You are required to define the following four components. All definitions must be based solely on the available variables listed above, and the outputs of both `f_V` and `f_E` must always be positive.

1. **ξ_v (Node Feature Extraction)**
   Define how to extract a feature vector for each entity (node) from the terminal state. Clearly specify which fields are used and how they are normalized or transformed.

2. **ξ_e (Graph Edge Construction)**
   Define how edges and edge weights are constructed between nodes. This is typically based on spatial distance or other relationships.

3. **f_V (Node State Aggregation Function)**
   The input consists of the features of all nodes (e.g., node health values). The output is a positive scalar representing the overall team quality with respect to the aggregated node properties.

4. **f_E (Edge Relationship Aggregation Function)**
   The input consists of the set of all edge weights. The output is a positive scalar representing the overall quality of the team's formation compactness, cooperative structure, or other relational properties.
```

Figure 10: SMAC task prompt.

<!-- User Prompt -->

```
### Task Description
The environment is the Simple Spread scenario in the Multi-Agent Particle Environment (MPE). It contains N agents and N landmarks. Agents can move (continuous action space), and the objective is to cover all landmarks.
An episode is considered successful when every landmark is covered by at least one agent. The reward is +1 upon success and −1 otherwise. Intermediate steps receive a reward of 0.

### Available Terminal State Variables
For each agent, you may use the following fields:
- `vel`: the agent's own linear velocity
- `pos`: the agent's own position
- `landmark_rel`: relative positions to the N landmarks
- `other_rel`: relative positions to the other agents

### Framework Requirements
Based on the available variables above, define the following four components. Important: Since there are no adversaries in this environment, the quality evaluations for successful and unsuccessful outcomes should be constructed based on the landmark coverage completion state and the remaining coverage gap, respectively.

1. **ξ_v (Node Feature Extraction)**
   Define how to extract a feature vector for each entity (node) from the terminal state. Clearly specify which fields are used and any normalization or transformations applied.

2. **ξ_e (Graph Edge Construction)**
   Define how edges and edge weights are constructed between nodes. This is typically based on spatial distance or other relational information.

3. **f_V (Node-State Aggregation Function)**
   The input is the set of node features, and the output is a positive scalar representing the overall team quality from the node perspective.

4. **f_E (Edge-Relation Aggregation Function)**
   The input is the set of all edge weights, and the output is a positive scalar representing the overall team quality from the perspective of the collaboration structure.
```

Figure 11: MPE Simple Spread prompt.

```
Review the following symbolic evaluator components for a trajectory-quality assessment task.

Task Description: {Consistent with the previous content}
Available Terminal State Variables: {Consistent with the previous content}
Proposed components: {ξ_v, ξ_e, f_V, f_E}

Check each of the following:
1. All referenced variables exist in the whitelist.
2. All operations are deterministic (no sampling or randomness).
3. Output shapes and types are consistent with each component's role.
4. No division-by-zero, NaN, or unbounded values; all outputs respect prescribed bounds.
5. Within-outcome directionality: among successes, larger aggregates mean better quality; among failures, larger aggregates mean worse severity.
6. Components are semantically reasonable for the described task.

For each violation, report the component, the issue, and a minimal fix. If all checks pass, confirm approval.
```

Figure 12: Validation prompt.

```
Convert the following confirmed symbolic components into a single executable Python module using NumPy.

Components: {ξ_v, ξ_e, f_V, f_E}
Available variables: {Consistent with the previous content}

Rules:
- Use only variables from the terminal state. No new constants or external calls beyond NumPy.
- Preserve the exact semantics—do not alter formulas or aggregation order.
- Raise an error if a required variable is missing.
- Keep code minimal and readable.
```

Figure 13: Code generation prompt.

- **Inputs and scope.** We provide the LLM with (i) a system prompt specifying the evaluator template, design constraints, and required outputs (Fig. 9), and (ii) a task-family prompt specifying the meaning of the original binary reward and available terminal-state variables (Fig. 10 for SMAC and Fig. 11 for MPE). No dense reward, trajectory-quality annotation, expert trajectory ranking, intermediate-state/action preference, or downstream learning signal is provided. The LLM is instructed to instantiate four deterministic symbolic components: ξ_v, ξ_e, f_V, and f_E. In Step 2, a separate validation prompt (Fig. 12) directs an independent LLM to review the proposed components. In Step 3, a code-generation prompt (Fig. 13) instructs the LLM to convert the confirmed symbolic components into executable functions.
- **Step 1: Symbolic component generation.** The LLM proposes deterministic rules for node features, relation functions, and node/edge aggregation using the provided terminal-state variables under the evaluator template.
- **Step 2: Programmatic and LLM-assisted validation.** The proposed symbolic rules are first subjected to programmatic checks for variable availability, deterministic operations, output shapes and types, numerical validity, and prescribed bounds. An independent LLM then checks whether the components satisfy the structural constraints of the evaluator template, including success-failure separation and within-outcome consistency. A human reviewer verifies flagged issues and confirms the finalized rules.
- **Step 3: Code generation and implementation testing.** The LLM converts the confirmed symbolic rules into executable functions without introducing new variables, followed by implementation-level validity checks.

SMAC: LLM-Proposed Instantiation

- Node features (ξ_v). For each unit (ally or enemy), use normalized health $h^{\mathrm{norm}} = \mathrm{clip}(h/h_{\max}, 0, 1)$, where h and $h_{\max}$ are the current and maximum health of a unit, respectively.
- Edges (ξ_e). Build a k-nearest-neighbor graph based on 2D position (x, y) by Euclidean distance d; assign Gaussian weights $\exp(-d/\sigma)$ on retained edges, where $\sigma > 0$ is a scale hyperparameter. Set $k = 2$ and $\sigma = 0.5$, shared across all SMAC maps.
- Node aggregation (f_V). Compute average normalized health separately for allies and enemies: $M_V^{\mathrm{ally}} = \frac{1}{|\mathcal{A}|}\sum_{i\in\mathcal{A}} h_i^{\mathrm{norm}}$, $M_V^{\mathrm{enemy}} = \frac{1}{|\mathcal{E}|}\sum_{i\in\mathcal{E}} h_i^{\mathrm{norm}}$, where $\mathcal{A}$ and $\mathcal{E}$ are the sets of ally and enemy units, and $|\cdot|$ denotes set cardinality. The node aggregation f_V selects M_V^{ally} for successful trajectories and M_V^{enemy} for all other trajectories.
- Edge aggregation (f_E). Measure factional cohesion on each induced subgraph by the ratio of weighted to maximal kNN edge sums, clamped to $[0, 1]$: $M_E^{\mathrm{ally}} \in [0, 1]$, $M_E^{\mathrm{enemy}} \in [0, 1]$. Analogously, the edge aggregation f_E selects M_E^{ally} for successful trajectories and M_E^{enemy} for all other trajectories.

MPE (Simple Spread): LLM-Proposed Instantiation

- Node features (ξ_v). For each agent, compute proximity to its nearest landmark: $q_i = \mathrm{clip}(1 - \min_j \|\Delta\ell_{i\to j}\|/d_{\max}, 0, 1)$, and include absolute position, where $\Delta\ell_{i\to j}$ is the displacement vector from agent i to landmark j. Set $d_{\max} = 1.0$.
- Edges (ξ_e). Construct an agent–agent conflict graph, assigning an edge weight of 1 if two agents share the same nearest landmark and 0 otherwise.
- Node aggregation (f_V). Average proximity over agents: $M_V = \frac{1}{N}\sum_i q_i \in [0, 1]$, where N is the number of agents. The node aggregation f_V scales M_V for successful trajectories and $1 - M_V$ for all other trajectories by $S = 100$. Scaling by a positive constant does not change the optimal policy.
- Edge aggregation (f_E). Quantify assignment conflicts among agents. Let c_j denote the number of agents whose nearest landmark is j. The number of conflicting agent pairs is $C_{\mathrm{conf}} = \sum_j \binom{c_j}{2}$, and the maximum possible under N agents is $\binom{N}{2}$. The edge score is

$M_E = 1 - C_{\text{conf}}/\binom{N}{2} \in [0, 1]$, where $M_E = 1$ indicates a perfect one-to-one agent–landmark assignment and $M_E = 0$ indicates complete overlap. Analogously, the edge aggregation f_E scales M_E for successful trajectories and $1 - M_E$ for all other trajectories by the same factor $S = 100$.

Notes

- All aggregations are positive and bounded; only terminal variables listed in the task prompts are used.
- One evaluator is shared across all SMAC v1/v2 maps, and another is shared across all MPE settings.

B.3.2 POLICY-ORDERING-PRESERVING REWARD RELABELING.

The recovered trajectory-level reward $\widehat{R}(\tau)$ is available only at the terminal transition. All non-terminal transitions receive the same relabeled value. In implementation, for each batch element, the terminal step is identified via the done mask and directly assigned $\tilde{R}(\tau)$; all valid preceding steps are assigned the corresponding normalized value. The default discount factor is $\gamma = 0.99$.

B.3.3 PRIOR-GUIDED CO-TRAINING

Although the preceding modules provide discriminative and relatively dense rewards, a randomly initialized learner still requires substantial interaction to encounter informative or successful trajectories. A few suboptimal prior trajectories can initially provide local task-relevant coverage, enabling TRACE to fully capitalize on the relabeled reward signals through dual-buffer co-training.

Dual Buffer Management. TRACE maintains a prior buffer $\mathcal{D}_{\text{E}}$ and an online buffer $\mathcal{D}_{\text{O}}$. The prior buffer is initialized with the available prior trajectories. Before $\mathcal{D}_{\text{E}}$ reaches its capacity, only successful online trajectories are admitted. Once it is full, a completed online trajectory τ is admitted only if its recovered trajectory reward $\widehat{R}(\tau)$ exceeds the minimum reward currently stored in $\mathcal{D}_{\text{E}}$, in which case it replaces the lowest-reward trajectory. All online trajectories not admitted to $\mathcal{D}_{\text{E}}$ are stored in $\mathcal{D}_{\text{O}}$. This rule progressively maintains a curated set of high-quality trajectories in the prior buffer while preserving the remaining online experience for policy learning.

Table 1: Key training hyperparameters for SMAC and MPE.

		SMAC	**MPE**
Training	Backbone	ft-QMIX	MATD3
	Batch Size B	128	1,000
	Online Buffer Capacity $\lvert\mathcal{D}_{\text{O}}\rvert$	5,000 trajectories	1,000,000 transitions
	Total Timesteps $T_{\max}$	3,000,000	3,000,000
	Parallel Environments	8	20
Optimization	ft-QMIX Learning Rate	5×10^{-4}	–
	MATD3 Actor Learning Rate	–	5×10^{-4}
	MATD3 Critic Learning Rate	–	1×10^{-3}
	Optimizer	Adam	Adam

Table 2: TRACE-specific hyperparameters.

Task Setting	β_{start}	α_{start}	$\lvert\mathcal{D}_{\text{E}}\rvert$	Storage Unit	T_{d}
SMAC v1	3	0.7	500	Trajectories	500,000
SMAC v2	1	0.7	500	Trajectories	200,000
MPE	0.1	0.7	200,000	Transitions	100,000

Progressive Scheduling. To smoothly transition from prior-dependent learning to online self-improvement, we introduce a progressive schedule that controls two coupled quantities: (i) the prior-buffer sampling ratio, which determines the fraction of each mini-batch drawn from $\mathcal{B}_{\mathrm{E}}$, and (ii) the regularization weight. Both quantities share a linear decay over a fixed schedule horizon T_{d}: $\alpha(t) = \alpha_{\text{start}} \cdot \max\left(0,\, 1 - \frac{t}{T_{\mathrm{d}}}\right)$, $\beta(t) = \beta_{\text{start}} \cdot \max\left(0,\, 1 - \frac{t}{T_{\mathrm{d}}}\right)$, where $\alpha(t)$ controls the proportion of samples drawn from the prior buffer, and $\beta(t)$ weights the offline regularization loss. Here, t is the current training step, T_{d} is the decay horizon, and α_{start} and β_{start} are their respective initial values. Both coefficients decay linearly to zero, progressively shifting learning toward online experience.

B.4 Hyperparameters

Table 1 summarizes the key training hyperparameters for SMAC and MPE. All methods use the same interaction budget, evaluation protocol, and task-specific prior data. For SMAC, we adopt the ft-QMIX backbone with a batch size of 128, an online replay-buffer capacity of 5,000 trajectories, 3M environment timesteps, and 8 parallel environments. The agent Q-networks and mixing network are jointly optimized with a learning rate of 5×10^{-4}. For MPE, we adopt the MATD3 backbone with a batch size of 1,000, an online replay-buffer capacity of 1M transitions, 3M environment timesteps, and 20 parallel environments. The actor and critic learning rates are 5×10^{-4} and 1×10^{-3}, respectively. Adam is used for optimization in both environments.

TRACE-Specific Hyperparameters. TRACE-specific hyperparameters are primarily associated with the prior-guided co-training module, as summarized in Table 2. Consistent with the corresponding replay implementations, SMAC buffers store complete trajectories, whereas MPE buffers store individual transitions.

B.5 Software Dependencies and Reproducibility

To lower the barrier to reproducing our results, the experimental code is standardized and packaged as follows:

- **Environment Compatibility:** All software dependencies are verified for compatibility with the specified environment versions (i.e., SMAC v1/v2 and Simple Spread V3).
- **Configuration Scripts:** Standardized configuration scripts are provided, documenting all hyperparameter settings in detail, including learning rates, discount factors, buffer sizes, and network architectures.
- **One-Click Execution:** A complete one-click execution script is included, enabling researchers to rapidly deploy and reproduce the main results reported in this paper.

C Additional Experimental Results

C.1 Baseline performance supplement: SMAC win-rate AUC

To provide an aggregate comparison of sample efficiency, we report the win-rate AUC (area under the curve), computed as the area under the test win-rate curve with respect to environment interaction steps. Under a fixed interaction budget, a larger AUC indicates that a method reaches and maintains higher win rates earlier, reflecting more efficient conversion of environment interactions into policy improvement. As shown in Table 3, TRACE achieves the highest win-rate AUC on all tasks. Its average AUC reaches 46.38 and 41.08 on medium and medium-replay, respectively, compared with 19.52 for the strongest baseline under both settings, corresponding to improvements of 137.60% and 110.45%. These results further demonstrate that TRACE effectively leverages only a few suboptimal demonstrations to improve the sample efficiency of MARL under binary sparse team rewards.

C.2 Baseline performance supplement: MPE success-rate AUC

As reported in Table 4, TRACE achieves average success-rate AUCs of 64.98 and 61.10 on medium and medium-replay, respectively, compared with 16.35 and 15.56 for the strongest baseline, corre-

Table 3: Win-rate AUC: Comparison of TRACE with ft-QMIX and SOTA baselines across various maps.

Method	5m_vs_6m	6h_vs_8z	MMM2	Protoss_5v5	Zerg_5v5	Terran_5v5	Average
				medium			
TRACE	$\mathbf{45.07 \pm 2.79}$	$\mathbf{33.31 \pm 2.81}$	$\mathbf{51.60 \pm 2.11}$	$\mathbf{48.78 \pm 1.47}$	$\mathbf{50.20 \pm 0.76}$	$\mathbf{49.34 \pm 0.74}$	**46.38**
ft-QMIX	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.13 ± 0.13	8.80 ± 3.42	8.24 ± 3.20	2.86
PegMARL	2.38 ± 2.38	0.00 ± 0.00	0.00 ± 0.00	17.77 ± 0.74	14.05 ± 0.27	14.86 ± 1.02	8.18
LAIES	37.90 ± 5.81	0.00 ± 0.00	6.36 ± 6.36	36.50 ± 1.46	8.87 ± 0.36	27.52 ± 4.03	19.52
CFCQL	6.34 ± 0.17	3.43 ± 0.32	14.80 ± 0.39	17.51 ± 0.32	26.13 ± 0.68	16.91 ± 0.49	14.19
MACal-QL	0.02 ± 0.01	0.01 ± 0.01	0.49 ± 0.16	25.30 ± 0.19	6.35 ± 0.47	16.56 ± 1.30	8.12
OVMSE	0.10 ± 0.10	9.73 ± 2.74	15.59 ± 3.12	16.95 ± 0.52	30.16 ± 0.13	22.38 ± 0.58	15.82
				medium_replay			
TRACE	$\mathbf{48.96 \pm 2.85}$	$\mathbf{16.04 \pm 5.73}$	$\mathbf{45.49 \pm 2.66}$	$\mathbf{43.49 \pm 2.21}$	$\mathbf{49.50 \pm 1.59}$	$\mathbf{42.99 \pm 1.78}$	**41.08**
ft-QMIX	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.13 ± 0.13	8.80 ± 3.42	8.24 ± 3.20	2.86
PegMARL	4.38 ± 3.08	0.00 ± 0.00	0.00 ± 0.00	11.75 ± 0.89	11.90 ± 0.21	8.66 ± 0.97	6.11
LAIES	37.90 ± 5.81	0.00 ± 0.00	6.36 ± 6.36	36.50 ± 1.46	8.87 ± 0.36	27.52 ± 4.03	19.52
CFCQL	6.32 ± 0.14	0.33 ± 0.11	13.05 ± 0.24	15.72 ± 1.14	24.15 ± 0.63	14.80 ± 0.28	12.39
MACal-QL	0.19 ± 0.08	0.01 ± 0.01	0.12 ± 0.07	12.67 ± 0.39	7.67 ± 0.20	5.10 ± 0.19	4.29
OVMSE	0.55 ± 0.24	0.30 ± 0.14	5.45 ± 3.88	13.88 ± 0.51	27.19 ± 1.14	1.73 ± 0.33	8.18

Table 4: Success-Rate AUC comparison on MPE.

Dataset	Method	n2	n3	n4	Average
medium	TRACE	**93.03±1.54**	**72.70±1.25**	**29.20±3.62**	**64.98**
	MATD3	0.26±0.03	0.07±0.06	0.01±0.01	0.12
	PegMARL	48.83±16.94	0.17±0.09	0.05±0.02	16.35
	LAIES	2.49±0.49	1.11±0.07	0.02±0.02	1.20
	CFCQL	0.19±0.06	0.00±0.00	0.00±0.00	0.06
	MACal-QL	0.13±0.06	0.05±0.03	0.01±0.01	0.06
	OVMSE	6.93±6.93	0.14±0.12	0.05±0.03	2.37
medium_replay	TRACE	**92.98±2.07**	**58.30±6.59**	**32.00±4.01**	**61.10**
	MATD3	0.26±0.03	0.07±0.06	0.01±0.01	0.12
	PegMARL	46.50±8.57	0.14±0.06	0.03±0.02	15.56
	LAIES	2.49±0.49	1.11±0.07	0.02±0.02	1.20
	CFCQL	0.21±0.04	0.00±0.00	0.01±0.01	0.07
	MACal-QL	0.25±0.10	0.02±0.02	0.02±0.02	0.10
	OVMSE	12.74±11.08	0.19±0.06	0.01±0.01	4.31

sponding to improvements of 297.4% and 292.7%. These results demonstrate that TRACE's advantages extend consistently to continuous-action MARL.

C.3 Comparison with a Dense-Reward Reference

We compare TRACE with ft-QMIX trained using the original dense environment rewards, denoted as ft-QMIX-DR. For a fair comparison, ft-QMIX-DR uses the same suboptimal prior trajectories as TRACE. It serves as a privileged reference rather than a standard baseline because dense rewards are unavailable in our target setting. As shown in Fig. 14, TRACE outperforms ft-QMIX-DR on several tasks and remains competitive on the others. Table 5 reports average AUCs of 46.38 versus 36.84 on medium and 41.08 versus 37.04 on medium-replay, corresponding to improvements of 25.9% and 10.9%, respectively. TRACE therefore achieves higher average sample efficiency than this privileged dense-reward reference using only sparse binary rewards, highlighting its practical value when handcrafted dense feedback is costly or unavailable.

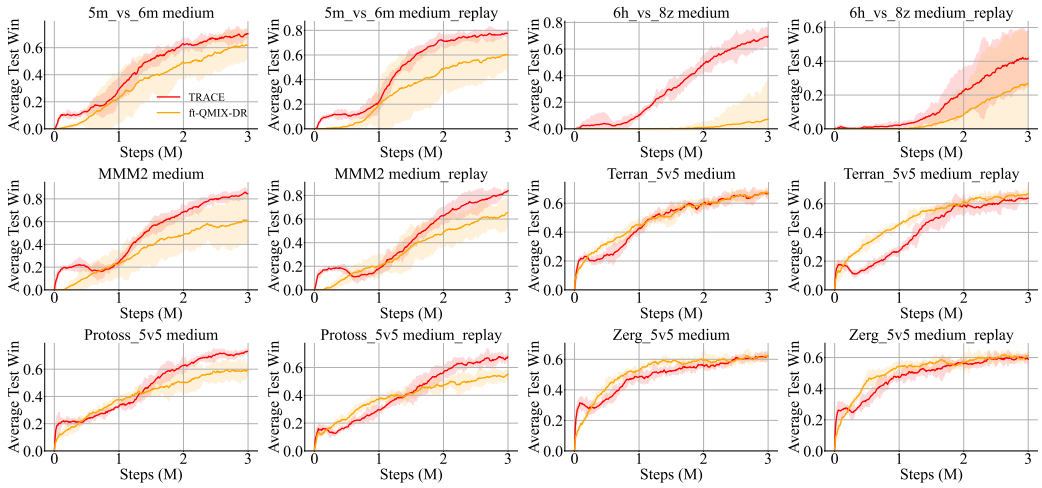


Figure 14: Comparison of TRACE with ft-QMIX-DR across various maps.

Table 5: Win-rate AUC: Comparison of TRACE with ft-QMIX-DR across various maps. Values after $\pm$ denote standard deviations.

Method	5m_vs_6m	6h_vs_8z	MMM2	Protoss_5v5	Zerg_5v5	Terran_5v5	Average
				medium			
TRACE	$\mathbf{45.07 \pm 2.79}$	$\mathbf{33.31 \pm 2.81}$	$\mathbf{51.60 \pm 2.11}$	$\mathbf{48.78 \pm 1.47}$	50.20 ± 0.76	49.34 ± 0.74	**46.38**
ft-QMIX-DR	35.66 ± 6.81	1.51 ± 2.97	36.75 ± 11.42	43.08 ± 2.58	$\mathbf{53.31 \pm 0.71}$	$\mathbf{50.76 \pm 1.31}$	36.84
				medium_replay			
TRACE	$\mathbf{48.96 \pm 2.85}$	$\mathbf{16.04 \pm 5.73}$	$\mathbf{45.49 \pm 2.66}$	$\mathbf{43.49 \pm 2.21}$	49.50 ± 1.59	42.99 ± 1.78	**41.08**
ft-QMIX-DR	33.36 ± 8.33	8.02 ± 7.25	36.13 ± 9.78	40.48 ± 2.08	$\mathbf{52.76 \pm 1.48}$	$\mathbf{51.48 \pm 0.71}$	37.04

C.4 Sensitivity Analysis

C.4.1 Sensitivity Analysis for Initial Regularization Weight

We study the initial offline-regularization weight β_{start}. As shown in Fig. 15, we test $\beta_{\text{start}} \in \{1, 3, 5, 7, 10\}$ on 5m_vs_6m-medium and Zerg_5v5-medium. The learning curves remain close under these values, and no setting causes clear instability or collapse. This indicates that TRACE is robust to β_{start} within a reasonable range. The prior-buffer regularization mainly provides early stabilization, while its influence is gradually reduced by the prior-to-online schedule.

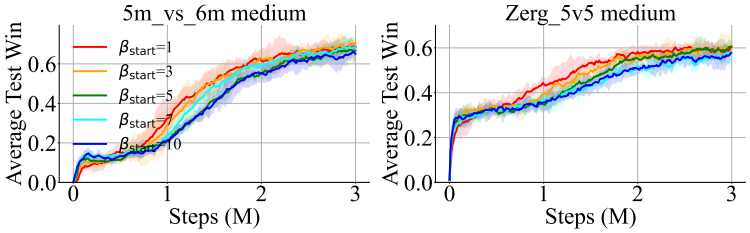


Figure 15: Sensitivity analysis of the optimization curriculum hyperparameter β_{start}.

C.4.2 Sensitivity Analysis for Decay Period

We analyze the decay period T_{d}, which controls how quickly TRACE shifts from prior-guided co-training to online-dominated learning. As shown in Fig. 16, we vary T_{d} over $\{0.1\text{M}, 0.3\text{M}, 0.5\text{M}, 0.7\text{M}, 1.0\text{M}\}$ environment steps. For $0.1\text{M} \leq T_{\text{d}} \leq 0.7\text{M}$, the overall learning

curves remain largely similar. When $T_{\mathrm{d}} = 1.0\mathrm{M}$, the performance shows a slight degradation. This is consistent with the role of the prior-to-online schedule: prior-buffer data and offline regularization should stabilize early learning, but an overly long decay period may keep training biased toward prior-guided updates for too long.

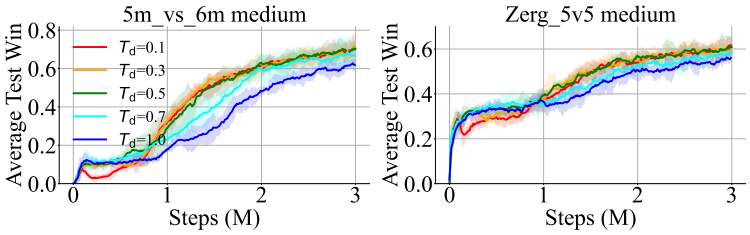


Figure 16: Sensitivity analysis of the optimization curriculum hyperparameter T_{d}.

D Theoretical Analysis of Intra-Outcome-Class Reward-Resolution Collapse

Fix a state s_t and consider two joint actions $\boldsymbol{a}$ and $\bar{\boldsymbol{a}}$, followed by the same continuation policy. Let $\tau_{\boldsymbol{a}}$ and $\tau_{\bar{\boldsymbol{a}}}$ denote their resulting trajectories. For a trajectory τ terminating after H transitions, define its discounted binary return from decision time t as

$$G_t(\tau) = \gamma^{H-t-1} y(\tau), \tag{3}$$

where $y(\tau) \in \{-1, 1\}$ denotes the binary outcome of τ, with -1 and 1 corresponding to failure and success, respectively, and $\gamma \in (0, 1]$ is the discount factor. Here, H denotes the terminal time of the realized trajectory and may be trajectory-dependent when episodes have variable lengths.

Because $G_t(\tau)$ depends only on the binary outcome and terminal time, two trajectories with the same outcome and terminal time receive identical binary returns even when their terminal multi-entity states reflect different task-relevant qualities. The corresponding state–joint-action value is

$$Q(s_t, \boldsymbol{a}) = \mathbb{E}\left[G_t(\tau_{\boldsymbol{a}}) \mid s_t, \boldsymbol{a}\right]. \tag{4}$$

To characterize the quality information omitted by binary feedback, let $c \in \{-1, 1\}$ index an outcome class that occurs with positive probability under both actions. For each such class, define

$$\begin{aligned}\Delta_c^q = \mathbb{E}\left[q(\tau_{\boldsymbol{a}}) \mid s_t, \boldsymbol{a}, y(\tau_{\boldsymbol{a}}) = c\right] \\ - \mathbb{E}\left[q(\tau_{\bar{\boldsymbol{a}}}) \mid s_t, \bar{\boldsymbol{a}}, y(\tau_{\bar{\boldsymbol{a}}}) = c\right],\end{aligned} \tag{5}$$

where $q(\tau)$ denotes task-consistent trajectory quality within the same outcome class, as reflected in terminal multi-entity states. It does not represent a unique latent reward implied by the binary outcome. Instead, Δ_c^q measures the class-conditional quality difference between trajectories produced by the two actions that is not explicitly represented by their binary rewards.

Proposition 1 (Intra-Outcome-Class Reward-Resolution Collapse)**.** *Suppose that the two joint actions have identical expected discounted binary returns:*

$$\mathbb{E}\left[G_t(\tau_{\boldsymbol{a}}) \mid s_t, \boldsymbol{a}\right] = \mathbb{E}\left[G_t(\tau_{\bar{\boldsymbol{a}}}) \mid s_t, \bar{\boldsymbol{a}}\right]. \tag{6}$$

Then

$$Q(s_t, \boldsymbol{a}) = Q(s_t, \bar{\boldsymbol{a}}), \tag{7}$$

even when $\Delta_c^q \neq 0$ for one or more nonempty outcome classes.

Moreover, consider the population squared-return objectives

$$\mathcal{L}_{\boldsymbol{a}}(v) = \mathbb{E}\left[\left(v - G_t(\tau_{\boldsymbol{a}})\right)^2 \mid s_t, \boldsymbol{a}\right], \tag{8}$$

$$\mathcal{L}_{\bar{\boldsymbol{a}}}(v) = \mathbb{E}\left[\left(v - G_t(\tau_{\bar{\boldsymbol{a}}})\right)^2 \mid s_t, \bar{\boldsymbol{a}}\right], \tag{9}$$

where $v \in \mathbb{R}$ is a tabular value estimate. Although the two objectives need not be identical, they have the same population minimizer:

$$\arg\min_{v \in \mathbb{R}} \mathcal{L}_{\boldsymbol{a}}(v) = \arg\min_{v \in \mathbb{R}} \mathcal{L}_{\bar{\boldsymbol{a}}}(v) = Q(s_t, \boldsymbol{a}) = Q(s_t, \bar{\boldsymbol{a}}). \tag{10}$$

Proof. By the definition of the state–joint-action value and Eq. (6),

$$\begin{aligned} Q(s_t, \boldsymbol{a}) &= \mathbb{E}\left[G_t(\tau_{\boldsymbol{a}}) \mid s_t, \boldsymbol{a}\right] \\ &= \mathbb{E}\left[G_t(\tau_{\bar{\boldsymbol{a}}}) \mid s_t, \bar{\boldsymbol{a}}\right] \\ &= Q(s_t, \bar{\boldsymbol{a}}). \end{aligned} \tag{11}$$

This equality constrains only the expected binary returns and does not require the two actions to induce identical trajectory or return distributions. In particular, it imposes no equality on the class-conditional trajectory qualities in Eq. (5). Hence, Δ_c^q may remain nonzero even though the two actions have identical binary values.

For the first action, the bias–variance decomposition gives

$$\begin{aligned} \mathcal{L}_{\boldsymbol{a}}(v) = \; & \mathrm{Var}\left[G_t(\tau_{\boldsymbol{a}}) \mid s_t, \boldsymbol{a}\right] \\ & + \left(v - \mathbb{E}\left[G_t(\tau_{\boldsymbol{a}}) \mid s_t, \boldsymbol{a}\right]\right)^2. \end{aligned} \tag{12}$$

Therefore,

$$\arg\min_{v \in \mathbb{R}} \mathcal{L}_{\boldsymbol{a}}(v) = Q(s_t, \boldsymbol{a}). \tag{13}$$

Similarly,

$$\arg\min_{v \in \mathbb{R}} \mathcal{L}_{\bar{\boldsymbol{a}}}(v) = Q(s_t, \bar{\boldsymbol{a}}). \tag{14}$$

Because the two action values are equal, the two squared-return objectives have the same population minimizer. Their minimum values or overall functional forms may differ because the corresponding return variances and higher-order distributional properties need not be the same. □

Proposition 1 shows that two joint actions can be assigned the same population value target under binary supervision even when their resulting trajectories differ systematically in task-consistent quality within the same outcome class. Binary supervision therefore provides no direct value target for recovering the distinction represented by Δ_c^q. Function approximation, exploration, or auxiliary objectives may produce indirect value differences, but such differences are not supervised by the original binary return. This many-to-one loss of reward resolution constitutes intra-outcome-class reward-resolution collapse.

E Theoretical Analysis of TRACE

We next analyze the trajectory-quality recovery and reward relabeling modules of TRACE. The analysis establishes two properties: the recovered trajectory reward restores a positive within-outcome value gap, and transition-level reward relabeling preserves the policy ordering induced by the recovered objective.

E.1 Recovery Setup and Conditions

For the two joint actions considered in Appendix D, equality of their discounted binary-return distributions implies equality of their outcome probabilities. Let

$$p_c = \Pr\left(y(\tau_{\boldsymbol{a}}) = c \mid s_t, \boldsymbol{a}\right) = \Pr\left(y(\tau_{\bar{\boldsymbol{a}}}) = c \mid s_t, \bar{\boldsymbol{a}}\right) \tag{15}$$

denote their common probability of outcome $c \in \{-1, 1\}$. Let

$$\mathcal{C} = \{c \in \{-1, 1\} : p_c > 0\} \tag{16}$$

denote the set of nonempty outcome classes.

For each $c \in \mathcal{C}$, define the class-conditional discounted recovered-reward gap as

$$\begin{aligned} \widehat{\Delta}_c = \; & \mathbb{E}\left[\gamma^{T(\tau_{\boldsymbol{a}})-t-1} \widehat{R}(\tau_{\boldsymbol{a}}) \mid s_t, \boldsymbol{a}, y(\tau_{\boldsymbol{a}}) = c\right] \\ & - \mathbb{E}\left[\gamma^{T(\tau_{\bar{\boldsymbol{a}}})-t-1} \widehat{R}(\tau_{\bar{\boldsymbol{a}}}) \mid s_t, \bar{\boldsymbol{a}}, y(\tau_{\bar{\boldsymbol{a}}}) = c\right]. \end{aligned} \tag{17}$$

The analysis uses the following recovery conditions.

Assumption 1 (Outcome Separation)**.** *The outcome-dependent mapping preserves the original success-over-failure preference:*

$$\sup_{\tau:y(\tau)=-1} \widehat{R}(\tau) < \inf_{\tau:y(\tau)=1} \widehat{R}(\tau). \tag{18}$$

Assumption 2 (Within-Outcome Consistency)**.** *For each nonempty outcome class $c \in \mathcal{C}$, the recovered reward preserves the direction of the task-consistent class-conditional quality difference:*

$$\Delta_c^q \geq 0 \quad \Longrightarrow \quad \widehat{\Delta}_c \geq 0, \tag{19}$$

and

$$\Delta_c^q > 0 \quad \Longrightarrow \quad \widehat{\Delta}_c > 0. \tag{20}$$

Assumption 3 (Bounded Evaluator Output)**.** *There exists a finite constant $B_{\widehat{R}}$ such that*

$$\left|\widehat{R}(\tau)\right| \leq B_{\widehat{R}} \qquad \textit{for every trajectory } \tau. \tag{21}$$

Assumption 1 ensures that introducing within-outcome distinctions does not reverse the original preference for success over failure. Assumption 2 is the expectation-level form of within-outcome discrimination: the recovered reward assigns a nonnegative advantage to the action producing no worse trajectories within each outcome class. Assumption 3 follows from the bounded node, relation, and aggregation functions used by the instantiated graph evaluator.

E.2 Within-Outcome Value-Resolution Restoration

Let $\widehat{Q}$ denote the action-value function induced by the recovered trajectory reward:

$$\widehat{Q}(s_t, \boldsymbol{a}) = \mathbb{E}\left[\gamma^{T(\tau_{\boldsymbol{a}})-t-1}\widehat{R}(\tau_{\boldsymbol{a}}) \mid s_t, \boldsymbol{a}\right]. \tag{22}$$

Proposition 2 (Within-Outcome Value-Resolution Restoration)**.** *Suppose that*

$$\Delta_c^q \geq 0 \qquad \textit{for every } c \in \mathcal{C}, \tag{23}$$

with a strict inequality for at least one nonempty outcome class. Under Assumption 2, the recovered action-value gap

$$\widehat{\Delta}_Q = \widehat{Q}(s_t, \boldsymbol{a}) - \widehat{Q}(s_t, \bar{\boldsymbol{a}}) \tag{24}$$

satisfies

$$\widehat{\Delta}_Q = \sum_{c\in\mathcal{C}} p_c \widehat{\Delta}_c > 0. \tag{25}$$

Thus, the recovered trajectory reward breaks the value tie induced by the original binary reward.

Proof. Applying the law of total expectation to Eq. (22) gives

$$\widehat{Q}(s_t, \boldsymbol{a}) = \sum_{c\in\mathcal{C}} p_c \mathbb{E}\Big[\gamma^{T(\tau_{\boldsymbol{a}})-t-1}\widehat{R}(\tau_{\boldsymbol{a}}) \mid s_t, \boldsymbol{a}, y(\tau_{\boldsymbol{a}}) = c\Big]. \tag{26}$$

The corresponding expression for $\widehat{Q}(s_t, \bar{\boldsymbol{a}})$ has the same outcome probabilities p_c. Subtracting the two expressions yields

$$\widehat{\Delta}_Q = \sum_{c\in\mathcal{C}} p_c \widehat{\Delta}_c. \tag{27}$$

By Assumption 2, $\widehat{\Delta}_c \geq 0$ for every $c \in \mathcal{C}$, and $\widehat{\Delta}_c > 0$ for at least one class. Because every $c \in \mathcal{C}$ satisfies $p_c > 0$, the weighted sum is strictly positive. □

Proposition 2 connects trajectory-level discrimination to value learning. Although the two actions have identical values under the original binary reward, the recovered trajectory reward produces a positive value gap in favor of the action yielding consistently better within-outcome trajectories.

Corollary 1 (Robust Learned Value Discrimination)**.** *Let $\widehat{Q}_\theta$ denote the learned estimate of $\widehat{Q}$. Suppose that its absolute estimation error is bounded by ϵ_Q for each compared action:*

$$\left|\widehat{Q}_\theta(s_t, \boldsymbol{a}) - \widehat{Q}(s_t, \boldsymbol{a})\right| \leq \epsilon_Q, \tag{28}$$

and

$$\left|\widehat{Q}_\theta(s_t, \bar{\boldsymbol{a}}) - \widehat{Q}(s_t, \bar{\boldsymbol{a}})\right| \leq \epsilon_Q, \tag{29}$$

where $\epsilon_Q \geq 0$ is the per-action value-estimation error bound. Then

$$\widehat{Q}_\theta(s_t, \boldsymbol{a}) - \widehat{Q}_\theta(s_t, \bar{\boldsymbol{a}}) \geq \widehat{\Delta}_Q - 2\epsilon_Q. \tag{30}$$

Consequently, the recovered ordering remains identifiable whenever

$$\widehat{\Delta}_Q > 2\epsilon_Q. \tag{31}$$

Proof. The assumed error bounds imply

$$\widehat{Q}_\theta(s_t, \boldsymbol{a}) \geq \widehat{Q}(s_t, \boldsymbol{a}) - \epsilon_Q \tag{32}$$

and

$$\widehat{Q}_\theta(s_t, \bar{\boldsymbol{a}}) \leq \widehat{Q}(s_t, \bar{\boldsymbol{a}}) + \epsilon_Q. \tag{33}$$

Subtracting the second inequality from the first gives Eq. (30). □

E.3 Policy-Ordering Preservation of Reward Relabeling

We finally analyze the transition-level reward relabeling module. Consider a completed trajectory

$$\tau = (s_0, \boldsymbol{a}_0, \ldots, s_H) \tag{34}$$

containing $H \geq 2$ transitions. Before relabeling, the recovered trajectory reward is assigned only to the terminal transition:

$$\widehat{r}_{t+1}(\tau) = \begin{cases} 0, & 0 \leq t < H-1, \\ \widehat{R}(\tau), & t = H-1. \end{cases} \tag{35}$$

The resulting discounted trajectory return is

$$\widehat{G}(\tau) = \sum_{t=0}^{H-1} \gamma^t \widehat{r}_{t+1}(\tau) = \gamma^{H-1} \widehat{R}(\tau). \tag{36}$$

TRACE retains the recovered terminal reward and assigns the following relabeled reward to every preceding transition:

$$\widetilde{r}_{t+1}(\tau) = \begin{cases} \dfrac{\gamma^{H-1}}{\sum_{\ell=0}^{H-2} \gamma^\ell} \widehat{R}(\tau), & 0 \leq t < H-1, \\ \widehat{R}(\tau), & t = H-1. \end{cases} \tag{37}$$

The relabeled discounted return is

$$\widetilde{G}(\tau) = \sum_{t=0}^{H-1} \gamma^t \widetilde{r}_{t+1}(\tau). \tag{38}$$

For a joint policy π, define

$$\widetilde{J}(\pi) = \mathbb{E}_{\tau \sim \pi}\left[\widetilde{G}(\tau)\right], \tag{39}$$

where $\widehat{J}(\pi) = \mathbb{E}_{\tau \sim \pi}\left[\widehat{G}(\tau)\right]$ denotes the recovered objective.

Proposition 3 (Policy-Ordering Preservation)**.** *For every trajectory τ,*

$$\widetilde{G}(\tau) = 2\widehat{G}(\tau). \tag{40}$$

Consequently, for every joint policy π,

$$\widetilde{J}(\pi) = 2\widehat{J}(\pi). \tag{41}$$

Therefore, for any two policies π_1 and π_2,

$$\operatorname{sign}\left[\widetilde{J}(\pi_1) - \widetilde{J}(\pi_2)\right] = \operatorname{sign}\left[\widehat{J}(\pi_1) - \widehat{J}(\pi_2)\right], \tag{42}$$

and

$$\arg\max_{\pi} \widetilde{J}(\pi) = \arg\max_{\pi} \widehat{J}(\pi). \tag{43}$$

Proof. Substituting Eq. (37) into Eq. (38) gives

$$\begin{aligned}\widetilde{G}(\tau) &= \sum_{t=0}^{H-2} \gamma^t \frac{\gamma^{H-1}}{\sum_{\ell=0}^{H-2}\gamma^\ell}\widehat{R}(\tau) + \gamma^{H-1}\widehat{R}(\tau) \\ &= \gamma^{H-1}\widehat{R}(\tau) + \gamma^{H-1}\widehat{R}(\tau) \\ &= 2\widehat{G}(\tau).\end{aligned} \tag{44}$$

Taking expectations over trajectories generated by π gives $\widetilde{J}(\pi) = 2\widehat{J}(\pi)$. Multiplication by the positive constant 2 preserves every pairwise policy comparison and the complete set of optimal policies. □

Proposition 3 shows that reward relabeling converts the recovered trajectory-level signal into transition-level rewards without introducing an additional policy preference. This separates the effects of the two stages: trajectory-quality recovery introduces a bounded modification of the original binary objective, whereas reward relabeling exactly preserves the policy ordering induced by that recovered objective.