BALANCED MULTI-AGENT REINFORCEMENT LEARNING VIA DESIRABILITY-GATED PSEUDO-EXPERT REGULARIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

The exploitation–exploration trade-off remains a central challenge in complex cooperative multi-agent tasks, where agents must discover coordinated joint behaviors from local observations. Entropy- or divergence-regularized actor-critic methods partly alleviate this challenge, but they often provide only weak guidance for learning complex team behaviors. Exploiting expert behaviors via behavior cloning can reproduce such behaviors, yet it usually assumes access to expert demonstrations and is therefore difficult to apply in online settings. To address this issue, we propose Balanced Multi-agent Reinforcement Learning (BMARL), a desirability-gated policy-regularization framework that balances exploration and exploitation using a pseudo-expert learned online from desirable trajectories. BMARL introduces a hybrid divergence-regularized objective: on desirable trajectories, the learning policy is regularized toward the pseudo-expert prior, while on undesirable trajectories, the objective reverts to maximum-entropy exploration. The framework is compatible with standard MARL backbones and mixer choices, and we provide theoretical analyses of policy evaluation and improvement. The proposed method is evaluated on various MARL tasks, including dense- and sparse-reward settings, and achieves further performance improvements over state-of-the-art methods.

1 INTRODUCTION

Cooperative multi-agent reinforcement learning (MARL) is typically formulated as a decentralized partially observable Markov decision process (Dec-POMDP), where agents must learn high-return joint behaviors from local histories under partial observability while exploring a large joint action space and coping with non-stationarity induced by interactions among agents (Oliehoek et al., 2008; Oliehoek & Amato, 2016; Gupta et al., 2017; Lowe et al., 2017). These challenges make MARL training difficult; consequently, learned policies often converge to local optima and fail to achieve successful coordination (Mahajan et al., 2019; Zhang et al., 2021; Na et al., 2024).

Under centralized training with decentralized execution (CTDE), value function factorization (Sunehag et al., 2017; Rashid et al., 2018; Son et al., 2019; Wang et al., 2020) and actor-critic learning (Zhang et al., 2021; Su & Lu, 2022; Wang et al., 2022; Yu et al., 2022) have substantially improved scalability in this setting. Yet, challenging cooperative benchmarks in SMAC Samvelyan et al. (2019) and GRF Kurach et al. (2020) continue to show that ***exploration and exploitation trade-off*** remains a core bottleneck, especially when successful team behavior requires long-horizon (Samvelyan et al., 2019) or the reward signal is sparsely provided (Jeon et al., 2022; Na & Moon, 2024).

In actor-critic MARL settings, one way to improve exploration is to regularize the policy with entropy or divergence terms. Multi-agent soft Q-learning, maximum-entropy factorization, and related methods all follow this general direction (Wei et al., 2018; Zhang et al., 2021; Liu et al., 2023). These methods broaden the action support of the policy and often improve optimization stability. However, entropy alone is often too weak to reliably discover complex coordinated behaviors (Su & Lu, 2022; Wang et al., 2022). In this regard, Su & Lu (2022) introduces divergence regularization with a target policy, which implicitly encourages the learned policy to exploit behaviors favored by the target policy. However, this approach often fails to learn in sparse-reward and long-horizon tasks.

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.

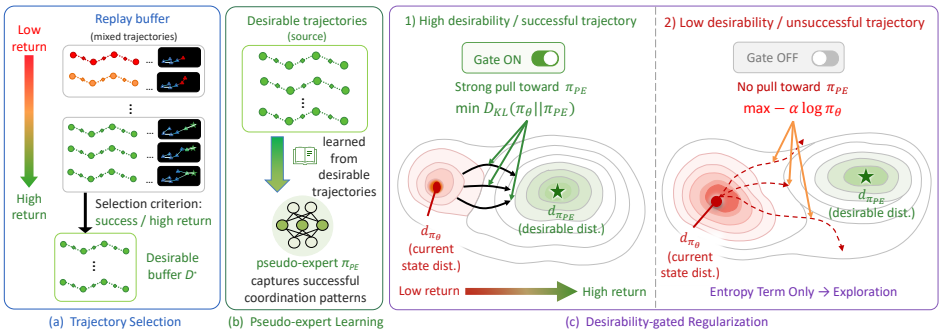


Figure 1: Illustration of the overall procedure of BMARL.

Behavior cloning (BC) offers complementary merits and thus has been widely adopted in robot learning (Argall et al., 2009; Mandlekar et al., 2021; Jang et al., 2022) and offline MARL settings (Pan et al., 2022; Zhang et al., 2023). However, the practical adoption of BC typically requires access to expert trajectories, and such expert supervision is usually unavailable in online MARL. Moreover, high-return trajectories that emerge during online training are often not explicitly utilized. Naively imitating these trajectories can cause exploration to collapse prematurely or lead the policy to overfit to behaviors observed during early training. Thus, balancing exploration and exploitation remains critical for efficient training.

This paper starts from the observation that successful self-generated trajectories can serve as a useful intermediate signal between pure exploration and pure imitation. We therefore propose **B**alanced **M**ulti-**A**gent **R**einforcement **L**earning (BMARL), a desirability-gated framework that converts desirable experience into a ***pseudo-expert prior***. BMARL constructs a desirability-gated prior and the corresponding policy-learning objective as follows: for desirable trajectories, the current policy is regularized toward a pseudo-expert learned from successful trajectories; for undesirable trajectories, the objective reverts to maximum-entropy exploration. In this way, BMARL explicitly balances exploitation of informative successful behaviors and continued exploration elsewhere. Table 1 compares the characteristics of BMARL to existing similar methods, and Figure 1 illustrates the overall procedure of the BMARL framework.

Table 1: Comparison of exploration and additional exploitation control.

Method	FOP (Zhang et al. (2021))	DMAC (Su & Lu (2022))	BMARL **(ours)**
Exploration Control	✓	✗	✓
Additional Exploitation Control	✗	✓	✓
Automatic Interchangeability	✓	✓	✓

Contribution. Our main contributions are as follows:

- ***Hybrid divergence-regularized objective construction:*** BMARL constructs the hybrid divergence-regularized objective for policy learning in cooperative MARL tasks, which automatically interchanges between pseudo-expert-guided exploitation on desirable trajectories and maximum-entropy exploration on undesirable trajectories, illustrated in Figure 1(c).
- ***Desirability-gated prior policy:*** BMARL conditionally utilizes a pseudo-expert trained online on successful trajectories (Figure 1(a) and (b)), enabling balanced exploration and exploitation without requiring external expert demonstrations.
- ***Theoretical analysis and comprehensive experiments:*** We provide policy evaluation and policy improvement analyses for the fixed-prior regularized control problem underlying BMARL. We also conduct extensive experiments, including key parameter studies, ablation studies, and qualitative analyses, to clarify the mechanism of BMARL.

2 Preliminaries

2.1 Dec-POMDP and CTDE

A general cooperative multi-agent task can be defined with Dec-POMDP (Oliehoek et al., 2008; Oliehoek & Amato, 2016) with state space $\mathcal{S}$, joint action space $\mathcal{A} = \prod_{i\in\mathcal{I}} \mathcal{A}_i$, transition probability $P(\cdot \mid s, \mathbf{a})$, team reward $r(s, \mathbf{a})$, and discount factor $\gamma \in [0, 1)$. Additionally, O is the observation

function generating an observation for each agent $o_i \in \Omega$. Under centralized training with decentralized execution (CTDE), we utilize global information as well as agents' observations $o_i \in \Omega$ during training, but in the execution phase, each agent i executes a decentralized policy $\pi_i(a_i \mid \tau_i)$ solely based on its action-observation history $\tau_i \in \mathcal{T}_i \equiv (\Omega \times A)$. The joint policy factorizes as

$$\pi(\mathbf{a} \mid \boldsymbol{\tau}) = \prod_{i \in \mathcal{I}} \pi_i(a_i \mid \tau_i), \text{ where } \boldsymbol{\tau} \triangleq (\tau_1, \ldots, \tau_n), \ \mathbf{a} \triangleq (a_1, \ldots, a_n). \tag{1}$$

2.2 Desirability in Cooperative MARL

In cooperative multi-agent tasks, we regard a trajectory as desirable when the agents successfully complete the given task. More technically, ***desirability*** can be defined as Definition 2.1. We assume that an episode-level binary desirability signal $d \in \{0, 1\}$ is available, such as a "win" signal in SMAC-style benchmarks (Samvelyan et al., 2019; Ellis et al., 2024) or a "goal-scoring" signal in GRF (Kurach et al., 2020). Thus, when we draw off-policy samples during training, each sample is annotated with a desirability label. However, to account for tasks without an explicit success signal, we define the desirability and desirable trajectories as follows.

Definition 2.1 (Desirability and Desirable Trajectory)**.** With a return threshold value R_{thr} and an episodic sequence $\mathcal{T} := \{s_0, \boldsymbol{a}_0, r_0, s_1, \boldsymbol{a}_1, r_1, \ldots, s_T\}$, if $\sum_{t=0}^{T-1} r_t \geq R_{\text{thr}}$, we define $\mathcal{T}$ as a desirable trajectory and denote such a trajectory with a desirability signal $d = 1$.

2.3 Entropy- and Divergence-Regularized MARL

Entropy regularization encourages stochastic policies by augmenting the reward with a policy entropy bonus, or equivalently, by using the per-step objective $r(s_t, a_t) - \omega \log \pi(a_t \mid s_t)$, where ω is a scaling factor, as in SAC (Haarnoja et al., 2018). In cooperative MARL, FOP (Zhang et al., 2021) extends this idea to a factorized joint policy $\boldsymbol{\pi}(\boldsymbol{a} \mid \boldsymbol{\tau}) = \prod_i \pi_i(a_i \mid \tau_i)$, leading to the regularized reward $r(s_t, \boldsymbol{a}_t) - \omega \sum_i \log \pi_i(a_{i,t} \mid \tau_{i,t})$. Divergence regularization further generalizes entropy regularization by introducing a target policy or prior $\boldsymbol{\rho}$. DMAC (Su & Lu, 2022) uses the per-step objective $r(s_t, \boldsymbol{a}_t) - \omega \sum_i \log \frac{\pi_i(a_{i,t}|\tau_{i,t})}{\rho_i(a_{i,t}|\tau_{i,t})}$ via policy decomposition, which penalizes the KL divergence between $\boldsymbol{\pi}$ and $\boldsymbol{\rho}$. Thus, divergence regularization encourages behavior patterns preferred by the prior policy ρ. Appendix A provides further related work and discussions regarding the novelty of BMARL.

3 Desirability-Gated Divergence-Regularized MARL

This section describes the **B**alanced **M**ulti-**A**gent **R**einforcement **L**earning (BMARL) framework. The central idea is to gate the regularization prior by trajectory desirability so that desirable experience is exploited through a pseudo-expert prior policy, while undesirable experience still encourages maximum-entropy exploration. Figure 2 illustrates the overview of the proposed method.

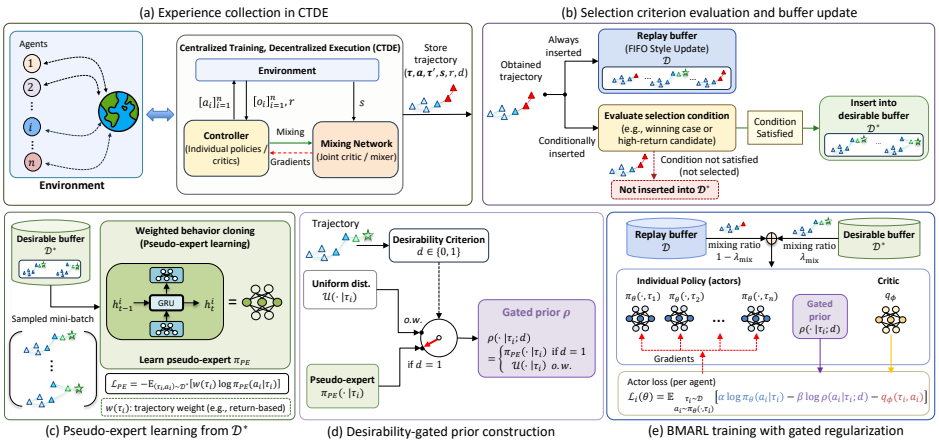


Figure 2: Overview of BMARL framework.

3.1 HYBRID DIVERGENCE-REGULARIZED OBJECTIVE WITH DESIRABILITY-GATED PRIOR

For a given prior policy $\rho(\cdot \mid \cdot; d)$ with a desirability label $d \in \{0, 1\}$, we define the hybrid divergence-regularized return as

$$J_\rho(\pi) \triangleq \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t \Big(r(s_t, \mathbf{a}_t) - \alpha \, \log \pi(\mathbf{a}_t \mid \boldsymbol{\tau}_t) + \beta \, \log \rho(\mathbf{a}_t \mid \boldsymbol{\tau}_t; d_t)\Big)\right], \tag{2}$$

where α and β are scale factors to control each term. Then, the corresponding value functions can be defined as follows:

$$\begin{aligned} Q_\rho^\pi(s, \mathbf{a}) &\triangleq r(s, \mathbf{a}) + \gamma \mathbb{E}_{s' \sim P(\cdot \mid s, \mathbf{a})}[V_\rho^\pi(s')], \\ V_\rho^\pi(s) &\triangleq \mathbb{E}_{\mathbf{a} \sim \pi(\cdot \mid \boldsymbol{\tau})}\Big[Q_\rho^\pi(s, \mathbf{a}) - \alpha \, \log \pi(\mathbf{a} \mid \boldsymbol{\tau}) + \beta \, \log \rho(\mathbf{a} \mid \boldsymbol{\tau}; d)\Big]. \end{aligned} \tag{3}$$

Here, Q_ρ^π and V_ρ^π are joint regularized value functions over the joint action $\mathbf{a}$, and their definition does not require any value decomposition. For fixed Q_ρ^π and prior ρ, the corresponding one-step improvement can be expressed as

$$\pi_\rho^+(\cdot \mid \boldsymbol{\tau}) = \arg\max_{\pi'} \mathbb{E}_{\mathbf{a} \sim \pi'} \left[Q_\rho^\pi(s, \mathbf{a}) - \alpha \log \pi'(\mathbf{a} \mid \boldsymbol{\tau}) + \beta \log \rho(\mathbf{a} \mid \boldsymbol{\tau}; d)\right]. \tag{4}$$

Assuming $\rho(\mathbf{a} \mid \boldsymbol{\tau}; d) > 0$, this yields the prior-weighted Boltzmann policy

$$\pi_\rho^+(\mathbf{a} \mid \boldsymbol{\tau}) = \frac{\rho(\mathbf{a} \mid \boldsymbol{\tau}; d)^{\beta/\alpha} \exp\left(Q_\rho^\pi(s, \mathbf{a})/\alpha\right)}{\sum_{\mathbf{a}' \in \mathcal{A}} \rho(\mathbf{a}' \mid \boldsymbol{\tau}; d)^{\beta/\alpha} \exp\left(Q_\rho^\pi(s, \mathbf{a}')/\alpha\right)}. \tag{5}$$

With this improved policy, the corresponding soft greedy value can be written as

$$V_\rho^+(s) = \alpha \log \sum_{\mathbf{a} \in \mathcal{A}} \rho(\mathbf{a} \mid \boldsymbol{\tau}; d)^{\beta/\alpha} \exp\left(Q_\rho^\pi(s, \mathbf{a})/\alpha\right). \tag{6}$$

Thus, the prior ρ reweights the standard entropy-regularized Boltzmann policy, where letting $\beta = 0$ recovers the maximum-entropy update, i.e., $V_\rho^+(s) = \alpha \log \sum_{\mathbf{a} \in \mathcal{A}} \exp\left(Q_\rho^\pi(s, \mathbf{a})/\alpha\right)$. In CTDE, this closed form is used as an interpretation of the joint objective, while the practical actor update is applied to factorized decentralized policies.

BMARL maintains (i) a learning policy π and (ii) a prior policy ρ used for divergence regularization, and both are factorized across agents, i.e., $\rho(\mathbf{a} \mid \boldsymbol{\tau}; d) \triangleq \prod_{i \in \mathcal{I}} \rho_i(a_i \mid \tau_i; d)$. Then, the per-agent prior policy in BMARL is defined as

$$\rho_i(a_i \mid \tau_i; d) \triangleq \begin{cases} \pi_{PE,i}(a_i \mid \tau_i), & d = 1\,, \\ \mathcal{U}_i(a_i \mid \tau_i), & d = 0 \text{ or } \pi_{PE} \text{ is not ready}, \end{cases} \tag{7}$$

where π_{PE} is a pseudo-expert trained from desirable trajectories, and $\mathcal{U}_i$ is uniform over currently available actions. This desirability-gated prior policy automatically adjusts the regularization term between the entropy regularization as in FOP (Zhang et al., 2021) and the divergence regularization by π_{PE}, according to the desirability d of a given trajectory, $\mathcal{T}$. Section 3.3 further discusses the characteristics of hybrid divergence regularization, and Appendix F.3 presents the details of the readiness of π_{PE}.

3.2 LEARNING A PSEUDO-EXPERT BY WEIGHTED BEHAVIOR CLONING

To make Eq. 2 effective in training, training a proper pseudo-expert policy for ρ is critical. To this end, BMARL introduces a desirable buffer $\mathcal{D}^\star$ of desirable trajectories and then trains π_{PE} via weighted behavior cloning. As illustrated in Figure 2(b), unlike the replay buffer $\mathcal{D}$, the sampled trajectory is conditionally inserted in the desirable buffer $\mathcal{D}^*$. For example, the desirable buffer can be maintained either as a priority queue, where sampled trajectories are inserted based on their episodic returns, or as a FIFO buffer that stores only desirable trajectories. We primarily use the priority queue style based on the return value of a trajectory. Once $\mathcal{D}^*$ is constructed, BMARL periodically updates a pseudo-expert policy π_{PE} via weighted behavior cloning with mini-batch samples, as illustrated in Figure 2(c).

$$\theta_{PE}^{new} \leftarrow \arg\max_{\theta} \; \mathbb{E}_{(\boldsymbol{\tau}, \mathbf{a}) \sim \mathcal{D}^\star}\Big[\sum_{i=1}^{n} w_i(\tau_i, a_i) \, \log \pi_{PE,\theta,i}(a_i \mid \tau_i)\Big] \tag{8}$$

Here, Eq. 8 updates each individual pseudo-expert policy $\pi_{PE,i}$ by per-agent weighted behavior cloning, and for $\pi_{PE,i}$, we use the same network structure as $\pi_{\theta,i}$. Similar to the advantage-weighted policy regression used in AWR (Peng et al., 2019), we use advantage-based weights only as a practical weighting scheme for pseudo-expert learning, assigning larger likelihood weights to agent actions with higher estimated per-agent advantages.

$$A_{\phi,i}(\tau_i, a_i) = q_{\phi,i}(\tau_i, a_i) - \mathbb{E}_{a_i' \sim \pi_{\theta,i}(\cdot|\tau_i)}\left[q_{\phi,i}(\tau_i, a_i')\right],$$
$$w_i(\tau_i, a_i) = \min\left\{\exp\left(\frac{A_{\phi,i}(\tau_i, a_i)}{\beta_{\mathrm{BC}}}\right), w_{\max}\right\}. \tag{9}$$

Here, $q_{\phi,i}$ is the per-agent critic estimate, β_{BC} is the weighting temperature for behavior cloning, and $w_{\max}$ is an optional clipping threshold used to prevent excessively large weights from noisy advantage estimates. During training, we simultaneously update the individual learning policies $\pi_{\theta,i}$ and thus directly using a rapidly varying pseudo-expert for prior construction may introduce instability. Thus, we use a smoothed pseudo-expert for prior construction and update its parameters by Polyak averaging (Polyak & Juditsky, 1992) with the smoothing coefficient, τ_{PE}:

$$\theta_{PE} \leftarrow (1 - \tau_{PE})\theta_{PE} + \tau_{PE}\theta_{PE}^{new}. \tag{10}$$

3.3 Characteristics of the BMARL objective

As discussed in Section 3.1, the prior policy ρ defined in Eq. 7 switches among the pseudo-expert, the target policy, and the uniform distribution according to desirability and pseudo-expert readiness. For example, when the trajectory is desirable, i.e., $d = 1$, and π_{PE} is obtained, then, Eq. 2 with $\alpha = \beta$ simply becomes the value function of divergence regularization with π_{PE}.

$$J_\rho(\pi) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty}\gamma^t\Big(r(s_t, \mathbf{a}_t) - \alpha \log \frac{\pi(\mathbf{a}_t \mid \boldsymbol{\tau}_t)}{\pi_{PE}(\mathbf{a}_t \mid \boldsymbol{\tau}_t)}\Big)\right]. \tag{11}$$

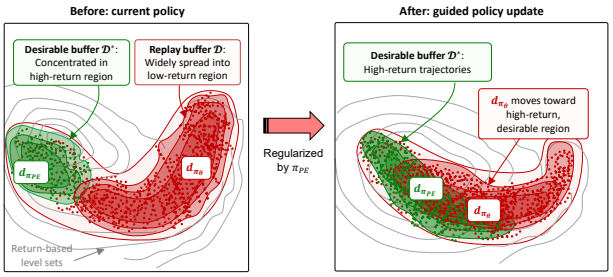


Figure 3: Changes in the state distribution of the learned policy π_θ through regularization by π_{PE}.

Through this regularization, the individual policy π_i adaptively learns the behavior patterns experienced in successful trajectories in $\mathcal{D}^*$, unlike DMAC (Su & Lu, 2022) regularized by the target policy. If the pseudo-expert π_{PE} properly captures behavior patterns in the high-return region or successful region, regularization by such behaviors pulls the state distribution of the current policy d_{π_θ} toward the state distribution of $\mathcal{D}^*$ or $d_{\pi_{PE}}$. Figure 3 illustrates how the state distribution of π_θ changes under divergence regularization toward π_{PE}. When a given trajectory is not desirable, i.e., $d = 0$, then $\rho(\cdot \mid \boldsymbol{\tau}; d) = u(\cdot \mid \boldsymbol{\tau})$ is uniform on available actions. Since $\log u(\mathbf{a} \mid \boldsymbol{\tau})$ is constant in the joint action $\mathbf{a}$ (given the availability mask), the term $\beta\,\mathbb{E}_{\mathbf{a}\sim\pi}[\log u(\mathbf{a} \mid \boldsymbol{\tau})]$ is an additive constant independent of π. Therefore, maximizing Eq. 2 is equivalent to the standard maximum-entropy objective, i.e., $J_\rho(\pi) = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty}\gamma^t\Big(r(s_t, \mathbf{a}_t) + \alpha\, H\big(\pi(\cdot \mid \boldsymbol{\tau}_t)\big)\Big)\Big]$. In this way, BMARL automatically balances between exploitation of desirable experience and exploration elsewhere, as illustrated in Figure 1(c).

3.4 Learning objectives and Desirability Thresholding

For training, BMARL uses a critic $q_{i,\phi}(\tau_i, a_i)$ per agent and combines these utilities with a centralized mixer. For each agent, the regularized local soft value function is defined as

$$v_{\phi,i}(\tau_i; d) \triangleq \alpha \log \sum_{a_i \in \mathcal{A}_i} \exp\left(\frac{q_{\phi,i}(\tau_i, a_i)}{\alpha} + \frac{\beta}{\alpha}\log \rho_i(a_i \mid \tau_i; d)\right). \tag{12}$$

Let $\mathcal{M}_\Phi$ be any mixer that maps local utilities and global context to a joint critic. We write

$$Q^{jt}_{\phi,\Phi}(s,\mathbf{a};d) \triangleq \mathcal{M}_\Phi\Big(s,\mathbf{a},\{q_{\phi,i}(\tau_i,a_i),\, v_{\phi,i}(\tau_i;d)\}_{i=1}^n\Big). \tag{13}$$

This notation keeps the BMARL objective independent of a specific value factorization.

The local soft value in Eq. 12 corresponds to the closed-form value of a regularized greedy policy update, following standard soft policy improvement and divergence-regularized policy learning (Haarnoja et al., 2018; Su & Lu, 2022). Thus, for a fixed conditioning variable h, prior $\rho(\cdot \mid h;d)$, and action-value function $Q(h,\cdot)$, we define the closed-form optimal policy for the local regularized improvement problem ν as

$$\nu(a \mid h;d) \triangleq \frac{\rho(a \mid h;d)^{\beta/\alpha}\exp(Q(h,a)/\alpha)}{\sum_{a'\in\mathcal{A}}\rho(a' \mid h;d)^{\beta/\alpha}\exp(Q(h,a')/\alpha)}. \tag{14}$$

The individual actor π is then updated by projecting the parametric policy onto this optimal policy through the KL objective

$$\min_\pi D_{\mathrm{KL}}(\pi(\cdot \mid h)\,\|\,\nu(\cdot \mid h;d))\,, \tag{15}$$

Thus, the actor objective follows the KL projection form of regularized policy improvement (Eq. 15) and becomes Eq. 18 according to Proposition 3.1.

Proposition 3.1 (KL projection form of the regularized actor objective)**.** *For a fixed conditioning variable h (either a joint history or a local agent history), desirability label d, prior policy $\rho(\cdot \mid h;d)$, and action-value function $Q(h,\cdot)$, define*

$$\nu(a \mid h;d) \triangleq \frac{\rho(a \mid h;d)^{\beta/\alpha}\exp\big(Q(h,a)/\alpha\big)}{Z(h;d)},\; Z(h;d) \triangleq \sum_{a'\in\mathcal{A}}\rho(a' \mid h;d)^{\beta/\alpha}\exp\big(Q(h,a')/\alpha\big). \tag{16}$$

Then minimizing $D_{\mathrm{KL}}(\pi(\cdot \mid h)\|\nu(\cdot \mid h;d))$ is equivalent to minimizing

$$\mathcal{J}(\pi;h,d) \triangleq \mathbb{E}_{a\sim\pi(\cdot|h)}\left[\alpha\log\pi(a \mid h) - \beta\log\rho(a \mid h;d) - Q(h,a)\right]. \tag{17}$$

The proof is deferred to Appendix C.1. Applying Proposition 3.1 with $h=\tau_i$ and $Q=q_{i,\phi}$ gives the practical factorized actor loss as

$$L_\pi(\theta) = \mathbb{E}_{(\boldsymbol{\tau},d)\sim\mathcal{D}}\left[\sum_{i=1}^n \mathbb{E}_{a_i\sim\pi_{\theta,i}(\cdot|\tau_i)}\left[\alpha\log\pi_{\theta,i}(a_i \mid \tau_i) - \beta\log\rho_i(a_i \mid \tau_i;d) - q_{i,\phi}(\tau_i,a_i)\right]\right]. \tag{18}$$

This policy objective also accounts for inter-agent coordination through the mixed critic defined in Eq. 13. Again, when $d=0$ and $\rho_i=u_i$, the prior term is constant with respect to the actor, and the update reduces to per-agent maximum-entropy policy optimization. For critic learning, with given sampled transition $(s,\boldsymbol{\tau},\mathbf{a},r,s',\boldsymbol{\tau}',d)$ and the next joint action as $\mathbf{a}'\sim\prod_i\pi_{\theta,i}(\cdot \mid \tau_i')$, we can express the joint value function as

$$\hat{V}^{jt,-}(s',\boldsymbol{\tau}',d) = Q^{jt}_{\phi^-,\Phi^-}(s',\mathbf{a}';d) - \frac{1}{n}\sum_{i=1}^n\Big(\alpha\log\pi_{\theta,i}(a_i' \mid \tau_i') - \beta\log\rho_i(a_i' \mid \tau_i';d)\Big). \tag{19}$$

By setting one-step TD target is $y = r + \gamma\hat{V}^{jt,-}(s',\boldsymbol{\tau}',d)$, the critic and mixer are trained by

$$\mathcal{L}_Q(\phi,\Phi) = \mathbb{E}_{\mathcal{D}}\Big[\big(Q^{jt}_{\phi,\Phi}(s,\mathbf{a};d) - y\big)^2\Big]. \tag{20}$$

With these policy and critic updates, Algorithm 1 in Appendix F.2 describes the overall training algorithm of BMARL. Due to the page limit, we defer the theoretical analysis of policy evaluation and improvement in BMARL to Appendix B.

Desirability Thresholding In many cooperative tasks such as SMAC-style benchmarks and GRF, R_{thr} can be set as the task-dependent maximum return $R_{\max}$. When an explicit success signal from the environment or $R_{\max}$ is unavailable, R_{thr} in Definition 2.1 should be appropriately defined. In BMARL, we set R_{thr} to the return threshold corresponding to the top $k_{\text{elite}}\%$ of trajectories in $\mathcal{D}^*$. As training progresses and better trajectories are obtained, R_{thr} increases adaptively. In this way, BMARL can provide adaptive and meaningful desirability signals even for tasks without explicit success signals. Appendix I provides further details, while Appendix K compares the binary desirability gating used in BMARL with continuous-gating alternatives.

4 EXPERIMENTS

In this section, we evaluate BMARL on cooperative MARL benchmarks such as SMAC, SMACv2 (Ellis et al., 2024), and GRF. Beyond performance evaluation based on win-rate, the main empirical questions are whether desirability-gated pseudo-expert regularization improves sample efficiency, stabilizes off-policy training, and yields a better exploration-exploitation balance than entropy- or divergence-only baselines.

Baseline Methods For actor-critic baseline methods, we consider FOP (Zhang et al., 2021), adopting entropy-maximization in MARL, DMAC (Su & Lu, 2022), introducing divergence regularization, MACPF (Wang et al., 2022), allowing sequential updates among agents, and MAPPO (Yu et al., 2022). In addition, we consider widely adopted value-based approaches such as QMIX (Rashid et al., 2018); RODE (Wang et al., 2021), which decomposes the action space according to designated roles to handle complex tasks; EMC (Zheng et al., 2021), a state-of-the-art algorithm that leverages episodic memory and curiosity-driven exploration; and MASER (Jeon et al., 2022), which utilizes intrinsic rewards to tackle sparse-reward tasks.

For evaluation, we use 4 different consecutive seeds, and in each test phase, 32 samples are used to compute the mean and variance of the win rate. During training, we schedule α and β by annealing α from α_s to α_f and increasing β from β_s to β_f. We use the same value for $\alpha_f = \beta_f$ across all tasks. Please see Appendix D for more details of the experiment and evaluation settings and hyperparameter settings. Due to space limitations, we defer additional experiments to the appendix. Appendix G.4 presents additional experiments on SMAX (Rutherford et al., 2024) using more random seeds. Appendix I describes desirability thresholding for tasks without explicit success signals and reports corresponding results on Overcooked and MABrax in JaxMARL (Rutherford et al., 2024).

4.1 COMPARATIVE EVALUATION ON SMAC AND SMACV2 (DENSE REWARD)

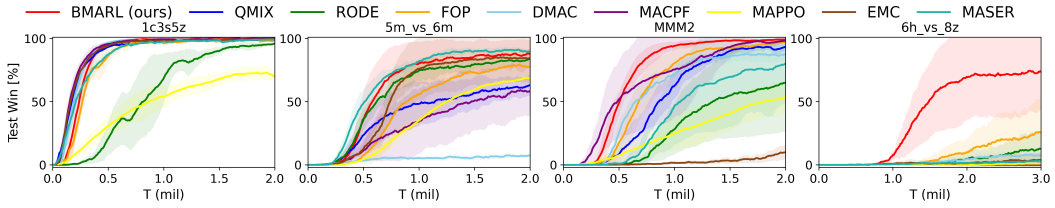


Figure 4: Performance Evaluation on Dense-reward SMAC Tasks.

Figure 4 illustrates the overall performance of BMARL compared to various baseline methods. By comparing BMARL with FOP and DMAC, we can notice whether the desirability-gated prior improves learning beyond entropy- or divergence-only baselines in dense-reward SMAC tasks. The main trend is that BMARL maintains competitive or improved win-rate learning. Especially, in **super hard** tasks such as `MMM2` and `6h_vs_8z`, BMARL shows strong performance compared to all baseline methods, including SOTA value-based approaches. Figure 5 compares the performance of BMARL with other baseline methods on representative SMACv2 tasks. Overall, BMARL shows strong and consistent performance on SMACv2 as well.

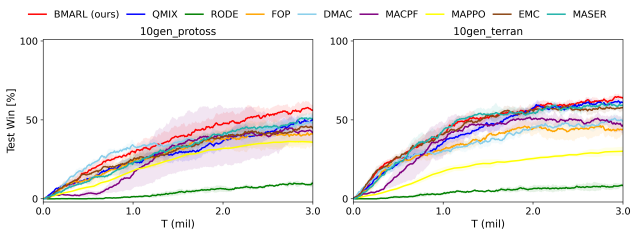
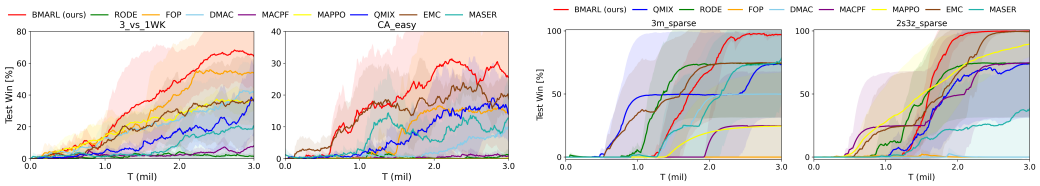


Figure 5: Performance Evaluation on SMACv2 Tasks.

4.2 COMPARATIVE EVALUATION ON SMAC AND GRF (SPARSE REWARD)

Figure 6 summarizes performance evaluations in sparse reward settings in GRF and SMAC. Please see details of experiment settings in Appendix E. These settings emphasize the intended use case of BMARL: when the reward signal is sparse, guidance by a pseudo-expert prior may help behavior learning among agents. As actor-critic (AC) methods require additional neural network training, AC-based models often fail to learn optimal policies without further guidance, as illustrated in Figure 6. On the other hand, BMARL succeeds in learning all seed cases in Figure 6(b).

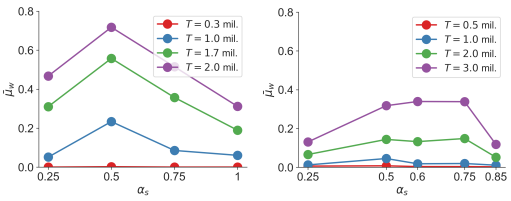


(a) GRF sparse-reward tasks. (b) SMAC sparse-reward tasks.

Figure 6: Sparse-reward performance comparison on GRF and SMAC.

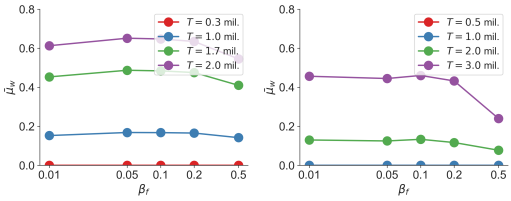
4.3 Parametric Study

In this section, we conduct the parametric study to analyze how the α and β affect the exploration-exploitation balance and thus overall training efficiency. To measure training efficiency, we adopt the *overall win rate* $\bar{\mu}_w$ from Na et al. (2024), defined as the normalized area under the learning curve, where values closer to 1 indicate better performance. Figures 8–9 show how the overall win rate $\bar{\mu}_w$ varies with α_s and β_f at different training times. As α controls the level of exploration, the proper level of α is important in improving learning performance. Thus, if we consider too large or too small α, it will also yield degraded performance compared to the optimal settings. In addition, overly weak regularization ($\beta \to 0$) may reduce BMARL to entropy-style exploration, whereas overly strong guidance ($\beta \gg 0$) may restrict the policy to the given π_{PE}. The results in Figures 8–9 align well with this intuition. Corresponding learning curves, win-rate analysis, and additional parametric studies for the mixing ratio λ_{mix} and the size of desirable buffer $|D^*|$ are presented in Appendix G.



(a) MMM2 (b) 3_vs_1WK

Figure 8: $\bar{\mu}_w$ over α_s and training time T.

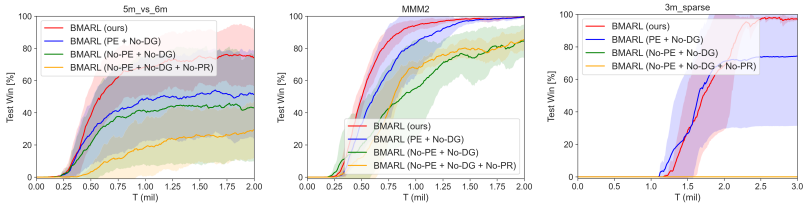


(a) MMM2 (b) 2s3z

Figure 9: $\bar{\mu}_w$ over β_f and training time T.

4.4 Ablation Study

In this study, we evaluate the effect of the proposed components, such as **pseudo-expert (PE)** and **desirability-gated (DG)**. In addition, we study the effect of **policy regularization (PG)**. In the No-PE variant, we use the target policy instead, similar to DMAC (Su & Lu, 2022). In the No-DG variant, we always use the divergence term induced by ρ, regardless of the desirability of a given trajectory. Figure 10 illustrates the performance of each ablated method. Notably, without PE, agents fail to learn coordinated behaviors in `3m_sparse` task, as shown by BMARL (No-PE + No-DG). This emphasizes the importance of guidance from the pseudo-expert. In addition, desirability-gating (DG) is also critical for further improving training, as it automatically balances exploration and exploitation according to desirability d. This is also supported by the high variance in MARL (PE + No-DG) in the `3m_sparse` task.

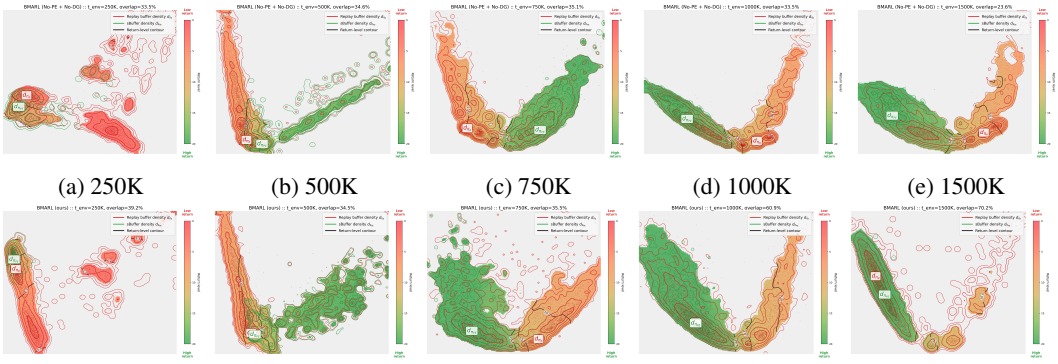


(a) 5m_vs_6m (b) MMM2 (c) 3m_sparse

Figure 10: Ablation study on SMAC MMM2, 5m_vs_6m, 3m_sparse ($\lambda_{mix} = 0.125$).

4.5 Qualitative Analysis

Pseudo-expert Guidance To further examine the role of pseudo-expert guidance, we visualize how the state distribution induced by the learning policy, d_{π_θ}, evolves over training. We estimate d_{π_θ} from



(a) 250K (b) 500K (c) 750K (d) 1000K (e) 1500K

(f) 250K (g) 500K (h) 750K (i) 1000K (j) 1500K

Figure 11: d_{π_θ} and $d_{\pi_{\mathrm{PE}}}$ for BMARL (No-PE + No-DG) (top row) and BMARL (bottom row).

the replay buffer $\mathcal{D}$ and compare it with the pseudo-expert state distribution, $d_{\pi_{\mathrm{PE}}}$, estimated from the desirable buffer $\mathcal{D}^*$. For this study, we consider the `MMM2` test case, using the same seed for BMARL (No-PE + No-DG), which corresponds to divergence regularization by the target policy (Su & Lu, 2022), and for BMARL (ours). The overlap score is computed as $100(1-\mathrm{TV})$ between the normalized PCA-2D state-density histograms of d_{π_θ} and $d_{\pi_{PE}}$, where TV denotes the total variation distance.

Figure 11 shows PCA-based state distributions of the final states and the corresponding return-level sets, similar to Figure 3. In the BMARL (No-PE + No-DG) case, as shown in the top row of Figure 3, d_{π_θ} does not consistently align with $d_{\pi_{\mathrm{PE}}}$ and often remains spread over low-return regions. In contrast, in the BMARL case, the overlap between d_{π_θ} and $d_{\pi_{\mathrm{PE}}}$ gradually increases as the pseudo-expert becomes the effective guidance. This indicates that the pseudo-expert prior learned from successful trajectories provides a meaningful exploitation signal that pulls the learning policy toward high-return regions. Figure 12 provides an additional qualitative analysis showing how successful and failed test cases are projected onto the evaluated state distribution.

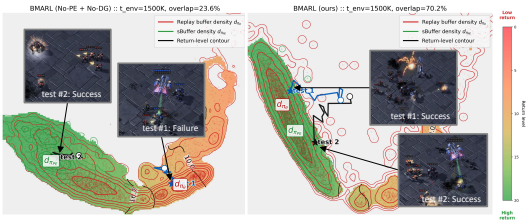


Figure 12: Projection of test cases onto the final-state distributions of BMARL (No-PE + No-DG) (left) and BMARL (ours) (right).

Evolving Pseudo-expert One important mechanism of BMARL is the evolving pseudo-expert induced by the desirable buffer $\mathcal{D}^*$, as *successful trajectories or those with higher returns progressively replace older ones* in $\mathcal{D}^*$ during training. Table 2 represents the KL-divergence between the reference (π_{ref}) and the candidate (π_-) policies, implying that the π_{PE} provides evolving guidance to π_θ. Here, † indicates self-KL divergence, and superscripts (t) and (T_f) denote the current timestep (corresponding to the training time shown in each column) and the final timestep, respectively. Appendix J provides further details of this analysis.

Table 2: KL-divergence between policies. Trajectories in $\mathcal{D}^*$ at the end of training are used.

$D_{\mathrm{KL}}(\pi_{\mathrm{ref}}\|\pi_-)$	500K	1000K	1500K	2000K
$D_{\mathrm{KL}}(\pi_\theta^{(T_f)}\|\pi_\theta^{(t)})$	1.576	0.190	0.114	$0.000^\dagger$
$D_{\mathrm{KL}}(\pi_\theta^{(T_f)}\|\pi_{\mathrm{PE}}^{(t)})$	1.593	0.293	0.227	0.175
$D_{\mathrm{KL}}(\pi_{\mathrm{PE}}^{(T_f)}\|\pi_\theta^{(t)})$	0.874	0.099	0.092	0.073

5 Conclusion

This paper presents BMARL, a desirability-gated divergence-regularized framework for cooperative MARL. BMARL uses a hybrid divergence-regularized objective, in which a pseudo-expert learned from successful self-generated trajectories serves as a prior policy for desirable trajectories, whereas the objective reduces to maximum-entropy exploration for undesirable trajectories. This design provides a way to balance the exploitation of informative successful behavior with continued exploration in complex MARL tasks, without requiring external expert demonstrations. Thus, BMARL offers a practical recipe for injecting structured guidance into online cooperative MARL.

AI USE STATEMENT

Generative AI tools were used for English grammar correction, readability improvements, and assistance with organizing and presenting experimental results. The authors reviewed and verified all AI-assisted outputs and take full responsibility for the final manuscript and reported results.

REFERENCES

Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009. doi: 10.1016/j.robot.2008.10.024.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. *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, 2024.

C Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021.

Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *International conference on autonomous agents and multiagent systems*, pp. 66–83. Springer, 2017.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pp. 1352–1361. PMLR, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *conference on Robot Learning*, pp. 991–1002. PMLR, 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.

Karol Kurach, Anton Raichuk, Piotr Stanczyk, Michal Zajkc, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 4501–4510, 2020.

Iou-Jen Liu, Unnat Jain, Raymond A Yeh, and Alexander Schwing. Cooperative exploration for multi-agent deep reinforcement learning. In *International Conference on Machine Learning*, pp. 6826–6836. PMLR, 2021.

Jiarong Liu, Yifan Zhong, Siyi Hu, Haobo Fu, Qiang Fu, Xiaojun Chang, and Yaodong Yang. Maximum entropy heterogeneous-agent reinforcement learning. *arXiv preprint arXiv:2306.10715*, 2023.

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

Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. Maven: Multi-agent variational exploration. *Advances in Neural Information Processing Systems*, 32, 2019.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

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

Hyungho Na, Yunkyeong Seo, and Il-chul Moon. Efficient episodic memory utilization of cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2403.01112*, 2024.

Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.

Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

Ling Pan, Longbo Huang, Tengyu Ma, and Huazhe Xu. Plan better amid conservatism: Offline multi-agent reinforcement learning with actor rectification. In *International conference on machine learning*, pp. 17221–17237. PMLR, 2022.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. *Advances in neural information processing systems*, 34:12208–12221, 2021.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International conference on machine learning*, pp. 4295–4304. PMLR, 2018.

Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 33:10199–10210, 2020.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Garðar Ingvarsson, Timon Willi, Ravi Hammond, Akbir Khan, Christian S de Witt, et al. Jaxmarl: Multi-agent rl environments and algorithms in jax. *Advances in Neural Information Processing Systems*, 37: 50925–50951, 2024.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*, pp. 5887–5896. PMLR, 2019.

Kefan Su and Zongqing Lu. Divergence-regularized multi-agent actor-critic. In *International Conference on Machine Learning*, pp. 20580–20603. PMLR, 2022.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

Jiangxing Wang, Deheng Ye, and Zongqing Lu. More centralized training, still decentralized execution: Multi-agent conditional policy factorization. *arXiv preprint arXiv:2209.12681*, 2022.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020.

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

Ermo Wei, Drew Wicke, David Freelan, and Sean Luke. Multiagent soft q-learning. *arXiv preprint arXiv:1804.09817*, 2018.

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.

Fuxiang Zhang, Chengxing Jia, Yi-Chen Li, Lei Yuan, Yang Yu, and Zongzhang Zhang. Discovering generalizable multi-agent coordination skills from multi-task offline data. In *The Eleventh International Conference on Learning Representations*, 2023.

Tianhao Zhang, Yueheng Li, Chen Wang, Guangming Xie, and Zongqing Lu. Fop: Factorizing optimal joint policy of maximum-entropy multi-agent reinforcement learning. In *International conference on machine learning*, pp. 12491–12500. PMLR, 2021.

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.

A Related Work and Discussions

BMARL is related to three lines of research: cooperative MARL with centralized training and decentralized execution (CTDE), entropy- and divergence-regularized MARL, and imitation or successful-trajectory reuse. These lines address complementary challenges in cooperative learning. Value-factorized CTDE methods improve scalable joint credit assignment, regularized MARL methods encourage exploration or stable policy improvement, and imitation-style approaches exploit informative behaviors when high-quality data are available. BMARL connects these directions by learning a pseudo-expert prior from the agents' own desirable trajectories and using it as a desirability-gated regularizer, without requiring external demonstrations.

A.1 Cooperative MARL and value factorization

Cooperative MARL commonly adopts the CTDE paradigm, where agents use centralized information during training while executing decentralized policies at test time (Gupta et al., 2017; Lowe et al., 2017). A major branch of CTDE methods learns a factorized representation of the joint action-value function to make credit assignment tractable in large cooperative action spaces. Representative approaches include VDN, QMIX, QTRAN, Weighted QMIX, and QPLEX (Sunehag et al., 2017; Rashid et al., 2018; Son et al., 2019; Rashid et al., 2020; Wang et al., 2020). These methods differ in the expressivity and constraints of the value factorization, but share the goal of enabling decentralized action selection from a centralized training signal. By satisfying the individual global maximization condition (IGM), these methods maintain the consistency between agent-wise greedy action and joint greedy action, and thus yield decentralized execution.

Among actor–critic approaches, FOP (Zhang et al., 2021) introduces value factorization under the maximum-entropy framework by leveraging soft value functions. In addition, conditional policy-factorization methods such as MACPF model inter-agent dependencies during centralized training while extracting decentralized local policies (Wang et al., 2022).

BMARL is orthogonal to these factorization-based approaches. Rather than proposing a new mixer, policy factorization, or value-factorization constraint, BMARL focuses on how to construct and use a regularization prior from self-generated successful experience. Consequently, the proposed mechanism can be combined with different CTDE backbones and mixer designs.

A.2 Entropy- and divergence-regularized MARL

Entropy regularization has been widely used to encourage stochastic exploration and improve policy robustness in reinforcement learning (Haarnoja et al., 2017; 2018). In cooperative MARL, this idea has been extended through multi-agent soft learning and factorized maximum-entropy policy optimization (Wei et al., 2018; Zhang et al., 2021). DMAC further generalizes the regularized-control perspective by replacing a pure entropy bonus with a divergence regularizer toward a reference policy, thereby supporting more stable policy improvement (Su & Lu, 2022).

BMARL builds on this regularized MARL viewpoint but differs in how the reference policy is constructed and applied. In FOP, the regularization mainly promotes maximum-entropy exploration, whereas in DMAC, the reference policy is used for divergence-regularized exploitation. BMARL instead learns a pseudo-expert prior online from desirable trajectories and activates the corresponding policy regularization through a desirability gate. This allows agents to retain broad exploration on undesirable or uncertain trajectories while exploiting informative successful experience when it becomes available.

A.3 Imitation, behavior cloning, and successful-trajectory reuse

Behavior cloning and imitation learning provide a simple and effective way to reproduce structured behaviors from demonstrations, and have been widely used in robot learning (Argall et al., 2009; Mandlekar et al., 2021; Jang et al., 2022). Related ideas also appear in offline MARL, where policies can benefit from high-quality offline data, actor rectification, or reusable coordination skills (Pan et al., 2022; Zhang et al., 2023). In online sparse-reward settings, informative experience can be exploited through curiosity-driven exploration, cooperative exploration, or episodic-memory-based reuse (Pathak et al., 2017; Burda et al., 2018; Liu et al., 2021; Na et al., 2024).

A.4 Distinct Characteristics of BMARL

BMARL learns from self-generated online experience without external demonstrations. In this setting, an unconditional actor objective such as $\mathcal{L}_{\mathrm{RL}} + \lambda \mathcal{L}_{\mathrm{BC}}$ can keep reinforcing early successful but suboptimal coordination. BMARL instead uses weighted behavior cloning to distill a separate pseudo-expert, stabilizes that prior through temporal smoothing, and controls its influence through trajectory desirability. The actor continues to optimize the reward-dependent objective in Eq. 2, and thus *it is not trained solely to reproduce the pseudo-expert.*

The analysis in Appendices B and C establishes policy evaluation, policy improvement, and convergence for the idealized fixed-prior subproblem under the stated assumptions. These results do not establish global convergence of the full deep-learning procedure in which the prior, actor, critic, and data distribution change simultaneously. The component ablations and the diagnostics in Appendix J provide complementary empirical evidence about the roles of prior learning and gating.

B Fixed-prior policy evaluation and improvement

The following analysis summarizes the regularized control problem induced by BMARL when the gated prior is held fixed. This fixed-prior view applies to each snapshot of the algorithm after choosing a prior according to Eq. 7.

Definition B.1 (Divergence policy evaluation operator)**.** Given fixed π and ρ, define Γ_ρ^π on any $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ by

$$(\Gamma_\rho^\pi Q)(s, \mathbf{a}) \triangleq r(s, \mathbf{a}) + \gamma\, \mathbb{E}_{s', \mathbf{a}'} \Big[Q(s', \mathbf{a}') - \alpha \log \pi(\mathbf{a}' \mid \boldsymbol{\tau}') + \beta \log \rho(\mathbf{a}' \mid \boldsymbol{\tau}'; d') \Big], \tag{21}$$

where $s' \sim P(\cdot \mid s, \mathbf{a})$ and $\mathbf{a}' \sim \pi(\cdot \mid \boldsymbol{\tau}')$.

Lemma B.2 (Contraction of policy evaluation)**.** *For any fixed π and ρ, Γ_ρ^π is a γ-contraction in the sup norm. Hence, $Q_{k+1} = \Gamma_\rho^\pi Q_k$ converges to the unique fixed point Q_ρ^π.*

Given $Q_\rho^{\pi_{\mathrm{old}}}$, we can define the prior-weighted Boltzmann target as

$$\nu_{\mathrm{old}}(\mathbf{a} \mid \boldsymbol{\tau}) \propto \rho(\mathbf{a} \mid \boldsymbol{\tau}; d)^{\beta/\alpha} \exp\left(\frac{1}{\alpha} Q_\rho^{\pi_{\mathrm{old}}}(s, \mathbf{a}) \right). \tag{22}$$

Then, the policy improvement can be shown by Lemma B.3.

Lemma B.3 (Divergence policy improvement)**.** *If π_{new} satisfies, for all $\boldsymbol{\tau}$,*

$$\pi_{\mathrm{new}}(\cdot \mid \boldsymbol{\tau}) \in \arg\min_{\pi}\ D_{\mathrm{KL}}(\pi(\cdot \mid \boldsymbol{\tau}) \,\|\, \nu_{\mathrm{old}}(\cdot \mid \boldsymbol{\tau}))\,, \tag{23}$$

then $Q_\rho^{\pi_{\mathrm{new}}}(s, \mathbf{a}) \geq Q_\rho^{\pi_{\mathrm{old}}}(s, \mathbf{a})$ for all $(s, \mathbf{a})$.

With Lemma B.2 and B.3, the following policy iteration can be derived. Proofs of Lemma B.2, Lemma B.3, and Theorem B.4 are provided in Appendix C.

Theorem B.4 (Divergence-regularized policy iteration)**.** *Under the fixed prior assumption, starting from any $\pi^{(0)}$, iterating policy evaluation $Q^{(k)} \leftarrow Q_\rho^{\pi^{(k)}}$ and policy improvement via Eq. 23 yields $J_\rho(\pi^{(k+1)}) \geq J_\rho(\pi^{(k)})$. Moreover, the sequence converges to an optimal policy for Eq. 2.*

Although Theorem B.4 assumes a fixed prior ρ, this can be applied to the BMARL case. As training proceeds, the successful trajectories accumulate to $\mathcal{D}^*$. Then, π_{PE} converges to policies that prevails in $\mathcal{D}^*$. In this way, π_{PE} can work as a fixed prior that guarantees desirability as training proceeds. The qualitative analysis in Section 4.5 and further experiments in Appendix J illustrate how π_θ converges to π_{PE}.

C Omitted Proofs

In this section, we provide the omitted proofs for propositions, Lemmas, and Theorems discussed in Section 3.4 and Appendix B.

C.1 Proof of Proposition 3.1

Proposition 3.1 (KL projection form of the regularized actor objective). For a fixed conditioning variable h (either a joint history or a local agent history), desirability label d, prior policy $\rho(\cdot \mid h; d)$, and action-value function $Q(h, \cdot)$, define

$$\nu(a \mid h; d) \triangleq \frac{\rho(a \mid h; d)^{\beta/\alpha} \exp\big(Q(h,a)/\alpha\big)}{Z(h; d)},\ Z(h; d) \triangleq \sum_{a' \in \mathcal{A}} \rho(a' \mid h; d)^{\beta/\alpha} \exp\big(Q(h,a')/\alpha\big). \tag{24}$$

Then minimizing $D_{\mathrm{KL}}(\pi(\cdot \mid h) \| \nu(\cdot \mid h; d))$ is equivalent to minimizing

$$\mathcal{J}(\pi; h, d) \triangleq \mathbb{E}_{a \sim \pi(\cdot|h)} \left[\alpha \log \pi(a \mid h) - \beta \log \rho(a \mid h; d) - Q(h, a)\right]. \tag{25}$$

Proof. For brevity, write $\nu(a) = \nu(a \mid h; d)$, $\rho(a) = \rho(a \mid h; d)$, and $Q(a) = Q(h, a)$. Starting from the KL divergence,

$$D_{\mathrm{KL}}(\pi \| \nu) = \sum_{a \in \mathcal{A}} \pi(a \mid h) \log \frac{\pi(a \mid h)}{\nu(a)} \tag{26}$$

$$= \sum_{a \in \mathcal{A}} \pi(a \mid h) \log \pi(a \mid h) - \sum_{a \in \mathcal{A}} \pi(a \mid h) \log \nu(a). \tag{27}$$

By Eq. 24,

$$\log \nu(a) = \frac{\beta}{\alpha} \log \rho(a) + \frac{1}{\alpha} Q(a) - \log Z(h; d). \tag{28}$$

Substituting this expression gives

$$D_{\mathrm{KL}}(\pi \| \nu) = \frac{1}{\alpha}\, \mathbb{E}_{a \sim \pi(\cdot|h)}[\alpha \log \pi(a \mid h) - \beta \log \rho(a \mid h; d) - Q(h, a)] + \log Z(h; d). \tag{29}$$

The final term is independent of π, and multiplying by the positive constant α does not change the minimizer. Thus, minimizing the KL projection objective is equivalent to minimizing Eq. 25. □

C.2 Detailed proof of Lemma B.2

Lemma B.2 (Contraction of policy evaluation). For any fixed π and ρ, Γ_ρ^π is a γ-contraction in the sup norm. Hence $Q_{k+1} = \Gamma_\rho^\pi Q_k$ converges to the unique fixed point Q_ρ^π.

Proof. Let Q_1, Q_2 be two action-value functions. Subtracting Eq. 21 for Q_1 and Q_2 cancels the reward and regularizer terms. Thus, for any $(s, \mathbf{a})$,

$$\left|(\Gamma_\rho^\pi Q_1)(s, \mathbf{a}) - (\Gamma_\rho^\pi Q_2)(s, \mathbf{a})\right| = \gamma \left|\mathbb{E}\big[Q_1(s', \mathbf{a}') - Q_2(s', \mathbf{a}')\big]\right| \le \gamma \left\|Q_1 - Q_2\right\|_\infty.$$

Taking the supremum over $(s, \mathbf{a})$ yields $\|\Gamma_\rho^\pi Q_1 - \Gamma_\rho^\pi Q_2\|_\infty \le \gamma \|Q_1 - Q_2\|_\infty$. Since $\gamma \in [0, 1)$, Banach's fixed point theorem implies convergence to a unique fixed point. □

C.3 Detailed proof of Lemma B.3

Lemma B.3 (Divergence policy improvement). If π_{new} satisfies, for all $\boldsymbol{\tau}$,

$$\pi_{\mathrm{new}}(\cdot \mid \boldsymbol{\tau}) \in \arg\min_{\pi}\ D_{\mathrm{KL}}(\pi(\cdot \mid \boldsymbol{\tau}) \,\|\, \nu_{\mathrm{old}}(\cdot \mid \boldsymbol{\tau}))\,, \tag{30}$$

then $Q_\rho^{\pi_{\mathrm{new}}}(s, \mathbf{a}) \ge Q_\rho^{\pi_{\mathrm{old}}}(s, \mathbf{a})$ for all $(s, \mathbf{a})$.

Proof. By the definition of ν_{old}, for each $\boldsymbol{\tau}$,

$$\nu_{\mathrm{old}}(\mathbf{a} \mid \boldsymbol{\tau}) = \frac{1}{Z_{\mathrm{old}}(\boldsymbol{\tau})} \exp\left(\frac{1}{\alpha}\left[Q_\rho^{\pi_{\mathrm{old}}}(s, \mathbf{a}) + \beta \log \rho(\mathbf{a} \mid \boldsymbol{\tau}; d)\right]\right),$$

where $Z_{\text{old}}(\boldsymbol{\tau})$ is the normalizing constant independent of π. Therefore,

$$-\alpha\, D_{\text{KL}}(\pi(\cdot \mid \boldsymbol{\tau}) \,\|\, \nu_{\text{old}}(\cdot \mid \boldsymbol{\tau})) = \mathbb{E}_{\mathbf{a}\sim\pi(\cdot|\boldsymbol{\tau})}\Big[Q_\rho^{\pi_{\text{old}}}(s,\mathbf{a}) - \alpha \log \pi(\mathbf{a} \mid \boldsymbol{\tau}) + \beta \log \rho(\mathbf{a} \mid \boldsymbol{\tau}; d)\Big] - \alpha \log Z_{\text{old}}(\boldsymbol{\tau}).$$

Since the last term does not depend on π, minimizing the KL divergence in Eq. 30 is equivalent to maximizing, pointwise in $\boldsymbol{\tau}$,

$$\mathcal{I}(\pi; \boldsymbol{\tau}) \triangleq \mathbb{E}_{\mathbf{a}\sim\pi(\cdot|\boldsymbol{\tau})}\Big[Q_\rho^{\pi_{\text{old}}}(s,\mathbf{a}) - \alpha \log \pi(\mathbf{a} \mid \boldsymbol{\tau}) + \beta \log \rho(\mathbf{a} \mid \boldsymbol{\tau}; d)\Big].$$

Thus, for every $\boldsymbol{\tau}$,

$$\mathcal{I}(\pi_{\text{new}}; \boldsymbol{\tau}) \geq \mathcal{I}(\pi_{\text{old}}; \boldsymbol{\tau}). \tag{31}$$

Now define the one-step soft backup under π_{new} as

$$(\Gamma_\rho^{\pi_{\text{new}}} Q)(s,\mathbf{a}) \triangleq r(s,\mathbf{a}) + \gamma \mathbb{E}_{s',\boldsymbol{\tau}',\mathbf{a}'}\Big[Q(s',\mathbf{a}') - \alpha \log \pi_{\text{new}}(\mathbf{a}' \mid \boldsymbol{\tau}') + \beta \log \rho(\mathbf{a}' \mid \boldsymbol{\tau}'; d)\Big],$$

where s' is sampled from the transition dynamics and $\mathbf{a}' \sim \pi_{\text{new}}(\cdot \mid \boldsymbol{\tau}')$. Using Eq. 31 at the next trajectory $\boldsymbol{\tau}'$, we obtain

$$\begin{aligned}(\Gamma_\rho^{\pi_{\text{new}}} Q_\rho^{\pi_{\text{old}}})(s,\mathbf{a}) &= r(s,\mathbf{a}) + \gamma \mathbb{E}_{s',\boldsymbol{\tau}'}\left[\mathcal{I}(\pi_{\text{new}}; \boldsymbol{\tau}')\right] \\ &\geq r(s,\mathbf{a}) + \gamma \mathbb{E}_{s',\boldsymbol{\tau}'}\left[\mathcal{I}(\pi_{\text{old}}; \boldsymbol{\tau}')\right] \\ &= Q_\rho^{\pi_{\text{old}}}(s,\mathbf{a}).\end{aligned}$$

Hence, $\Gamma_\rho^{\pi_{\text{new}}} Q_\rho^{\pi_{\text{old}}} \geq Q_\rho^{\pi_{\text{old}}}$ holds in a pointwise.

Finally, $\Gamma_\rho^{\pi_{\text{new}}}$ is monotone and is a γ-contraction. Starting from $Q_0 = Q_\rho^{\pi_{\text{old}}}$ and defining $Q_{k+1} = \Gamma_\rho^{\pi_{\text{new}}} Q_k$, monotonicity gives

$$Q_1 \geq Q_0, \quad Q_2 \geq Q_1, \quad \dots$$

and contraction implies that Q_k converges to the unique fixed point of $\Gamma_\rho^{\pi_{\text{new}}}$, namely $Q_\rho^{\pi_{\text{new}}}$. Therefore,

$$Q_\rho^{\pi_{\text{new}}} = \lim_{k\to\infty} (\Gamma_\rho^{\pi_{\text{new}}})^k Q_\rho^{\pi_{\text{old}}} \geq Q_\rho^{\pi_{\text{old}}}.$$

This proves the desired policy improvement result. □

C.4 Detailed proof of Theorem B.4

Theorem B.4 (Divergence-regularized policy iteration). Under the fixed prior assumption, starting from any $\pi^{(0)}$, iterating policy evaluation $Q^{(k)} \leftarrow Q_\rho^{\pi^{(k)}}$ and policy improvement via Eq. 23 yields $J_\rho(\pi^{(k+1)}) \geq J_\rho(\pi^{(k)})$. Moreover, the sequence converges to an optimal policy for Eq. 2.

Proof. Lemma B.2 ensures the evaluation step returns the unique fixed point. Lemma B.3 yields monotonic improvement of $Q_\rho^{\pi^{(k)}}$, hence of $V_\rho^{\pi^{(k)}}$ and J_ρ. Since rewards are bounded and $\gamma < 1$, $J_\rho(\pi)$ is upper bounded, so the monotone sequence converges. Any limit point must be a fixed point of the improvement map, hence optimal for Eq. 2. □

D Details of Experiment Environment

D.1 Experiment Settings

Seed Selection for Experiments In complex MARL tasks, performance often exhibits substantial variability across random seeds. Consequently, evaluations with arbitrarily selected seeds may lead to either overly optimistic or pessimistic conclusions. To mitigate the effect of seed selection, we use consecutive random seeds throughout our analysis.

The SMAC, SMACv2, and GRF comparison uses four consecutive seeds for each reproduced method and 32 evaluation episodes per checkpoint, with matched environment-step budgets, reward definitions, and evaluation schedules. Separate SMAX evaluations in Appendix G.4 use six seeds for

both 6h_vs_8z and 2s3z, and report sample standard deviations. Overcooked and MABrax results in Appendix I.3 use three seeds and are reported separately. The threshold and continuous-gating ablations retain the four-seed standard-deviation conventions specified in their captions.

In addition to final win rate, we report area under the win-rate learning curve normalized by the evaluation horizon (AULC). Tables explicitly distinguish fractional AULC from AULC expressed as a percentage; a value of one, or 100%, respectively, corresponds to perfect win rate throughout that horizon. Consecutive seeds prevent selecting favorable seed identities, but the limited computing resources constrain us to a relatively small seed budget, which may reduce statistical precision in high-variance tasks. Nevertheless, the number of seeds used in our experiments is consistent with the range commonly adopted in prior MARL studies.

Mixer Structure BMARL adopts the FOP-style mixer implemented in MACPF (Wang et al., 2022), following FOP. For DMAC and MACPF, we use their designed mixer structures. For RODE, we use QMIX.

D.2 Hyperparameter Settings

For baseline methods, we use the default settings provided by the original papers. For omitted hyperparameters, we use the same hyperparameters that work for BMARL in the cases of FOP and DMAC, as they share the same or similar backbones.

For BMARL, as mentioned in Section 4, we use the scheduled α and β defined as:

$$\alpha(t) = \max\left(\alpha_f, \alpha_s - \frac{t_{env}}{T_{\alpha_e}}\right), \qquad \beta(t) = \min\left(\beta_f, \beta_s + \kappa_\beta \beta_f \frac{t_{env} - t_\beta^{\text{start}}}{T_{\beta_e}}\right). \tag{32}$$

Here, t_{env} is the training timestep and κ_β denotes whether π_{PE} is ready. If it is ready then $\kappa_\beta = 1$ and $\kappa_\beta = 0$ otherwise. For some hyperparameters, we use the same values for all tasks: α_f=$\beta_f = 0.05$, $\beta_s = 0.01$, $T_{\alpha_s} = T_{\beta_s} = 200K$, except for 2s3z_sparse which uses $T_{\beta_s} = 800K$. Table 3 summarizes task-dependent hyperparameters.

Table 3: Hyperparameter settings for experiments.

	task	α_s	T_ϵ	λ_{mix}	$\lvert\mathcal{D}\rvert^*$	rb/sb
SMAC (dense)	1c3s5z	0.5	50K	0.125	1000	rb
	5m_vs_6m	0.5	50K	0	1000	rb
	MMM2	0.5	200K	0.125	1000	rb
	6h_vs_8z	0.5	500K	0	500	rb
SMAC (sparse)	3m	0.5	50K	0.125	500	rb
	2s3z	1.5	500K	0	1000	sb
GRF (sparse)	3_vs_1WK	0.6	50K	0.125	1000	sb
	CA_easy	0.4	200K	0.125	1000	sb

In Table 3, λ_{mix} is the mixing ratio of samples from the desirable buffer $\mathcal{D}^*$ to those from the replay buffer $\mathcal{D}$, and rb/sb indicates whether the desirable buffer insertion condition, where rb represents *return*-based update, while sb represents *desirability*-based update. Note that since there is no reward except for scoring a goal in GRF, we use sb for GRF tasks.

For pseudo-expert training explained in Section 3.2, we use the same hyperparameters for all evaluation cases as: Polyak parameter $\tau_{PE} = 0.005$; $\beta_{BC} = 1.0$ and $w_{max} = 20$ in Eq.8; update frequency $i_{BC} = 100$; mini-batch samples $\mathcal{B}_{BC} = 32$; iterations per update with the same mini-batch samples $j_{BC} = 10$.

Shared settings and staged hyperparameter selection. The dense-reward SMAC tasks share $\alpha_s = 0.5$, while selected sparse-reward tasks use different initial entropy coefficients to accommodate exploration requirements and reward scales. For example, SMAX (Rutherford et al., 2024) uses

a different default scale for the normalized episodic return, with a maximum of approximately 2, compared with 20 in the original SMAC task. This difference can affect the relative influence of the regularization term in Eq. 17. By rescaling the SMAX rewards to match the SMAC return scale, BMARL can be applied effectively to SMAX without retuning the hyperparameters α_s and β_s. Please see Appendix G.4 for the corresponding results.

In addition, the initial and final values of β are shared across tasks, and no independent task-wise sweep of these endpoints was performed. Pseudo-expert optimization settings are also shared. Activation follows the fixed readiness rule in Appendix F.3. The return-priority construction (rb) is the default, and the success-based FIFO construction (sb) is used for `2s3z_sparse` and GRF, as specified in Table 3.

E Experiment Details

In this section, we describe the experimental environments and reward settings used in our experiments. We evaluate our method on the StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019) and Google Research Football (GRF) (Kurach et al., 2020). Table 4 summarizes the state dimension, action dimension, and maximum episodic length of each task. Table 5 reports the sparse reward configuration used for SMAC.

Table 4: Environment specifications for SMAC and GRF tasks.

Benchmark	Task	State dimension	Action dimension	Episodic length
SMAC	3m	48	9	60
SMAC	8m	168	14	120
SMAC	2s3z	120	11	120
SMAC	2m_vs_1z	26	7	150
SMAC	1c3s5z	270	15	180
SMAC	5m_vs_6m	98	12	70
SMAC	MMM2	322	18	180
SMAC	6h_vs_8z	140	14	150
GRF	3_vs_1WK	26	19	150
GRF	CA_easy	30	19	150

Sparse Reward settings in SMAC. For SMAC tasks, we also use a sparse reward setting. The agents receive a large positive reward when all enemies are defeated, a small positive reward whenever an enemy unit dies, and a penalty when an allied unit dies. The detailed reward configuration is summarized in Table 5.

Table 5: Sparse reward setting for SMAC tasks.

Condition	Reward
All enemies die, i.e., win	$+200$
Each enemy dies	$+10$
Each allied unit dies	-5

GRF reward setting. For GRF tasks, we use a goal-based sparse reward setting. In this setting, the agents receive a positive reward only when the controlled team scores a goal, and no intermediate reward is given for ball possession, passing, or movement. The reward at each timestep t is defined as

$$r_t^{\mathrm{GRF}} = \begin{cases} +1, & \text{if the controlled team scores a goal at timestep } t, \\ 0, & \text{otherwise.} \end{cases} \tag{33}$$

When scoring a goal, the episode ends. This sparse reward setting encourages agents to learn policies that directly maximize goal-scoring outcomes rather than relying on dense hand-crafted feedback.

F Implementation Details

F.1 Infrastructures and Code Implementation

For experiments, we mainly use NVIDIA GeForce RTX 4090 and RTX PRO 6000 GPUs. Our code is built on PyMARL Samvelyan et al. (2019) and the open-sourced code from HACPF (Wang et al., 2022) with modifications for training efficiency and for debugging some issues. Our code will be released as open source upon acceptance of the paper.

F.2 Algorithm Details

Algorithm 1 Base BMARL with desirability-gated divergence regularization

1: Maintain replay buffer $\mathcal{D}$ and desirable buffer $\mathcal{D}^\star$
2: Initialize π_θ, fallback target policy π_{θ^-}, and $\pi_{PE,\theta_{PE}}$
3: **for** each training iteration i **do**
4: Collect data with π_θ; store transitions with desirability d into $\mathcal{D}$
5: Update $\mathcal{D}^\star$ using return-priority (rb) or success-based FIFO (sb) insertion
6: **if** $|\mathcal{D}^\star| \geq 32$ & $i \geq 400$ & $\mathrm{mod}(i, i_{BC}) = 0$ **then**
7: Update θ_{PE}^{new} by weighted BC using Eq. 8
8: Update θ_{PE} using Eq. 10
9: **end if**
10: Sample minibatches $\mathcal{B}_\mathcal{D} \sim \mathcal{D}$ and $\mathcal{B}_{\mathcal{D}^\star} \sim \mathcal{D}^\star$ such that

$$|\mathcal{B}_\mathcal{D}| : |\mathcal{B}_{\mathcal{D}^\star}| = (1 - \lambda_{\mathrm{mix}}) : \lambda_{\mathrm{mix}}.$$

11: Construct the mixed mini-batch $\mathcal{B}_{\mathrm{mix}} = \mathcal{B}_\mathcal{D} \cup \mathcal{B}_{\mathcal{D}^\star}$
12: Construct $\rho(\cdot \mid \cdot; d)$ on $\mathcal{B}_{\mathrm{mix}}$ via Eq. 7
13: Form the mixed critic $Q^{jt}_{\phi,\Phi}$ on $\mathcal{B}_{\mathrm{mix}}$ according to Eq. 13
14: Update (ϕ, Φ) by minimizing the critic loss Eq. 20 over $\mathcal{B}_{\mathrm{mix}}$
15: Update θ by minimizing the actor loss Eq. 18 over $\mathcal{B}_{\mathrm{mix}}$
16: Periodically hard-copy critic/mixer targets via Eq. 34 and update policy targets via Eq. 35
17: **end for**

In Algorithm 1, i_{BC} determines the update frequency for a pseudo-expert. The conditions $|\mathcal{D}^\star| \geq 32$ & $i \geq 400$ in `Line #6` are for the readiness of pseudo-expert training. We hold these conditions across all tasks.

In addition, the BMARL implementation maintains two kinds of target networks. The critic and mixer use hard periodic copies,

$$\phi^- \leftarrow \phi, \qquad \Phi^- \leftarrow \Phi, \tag{34}$$

triggered every fixed number of training episodes. By contrast, the policy prior networks can be updated by Polyak averaging,

$$\theta^- \leftarrow (1 - \tau_\pi)\theta^- + \tau_\pi \theta, \tag{35}$$

This target update supplies the fallback prior π_{θ^-} before the pseudo-expert is ready.

F.3 Pseudo-expert readiness

The pseudo-expert becomes eligible for use only after both $|\mathcal{D}^*| \geq N_{\min} = 32$ stored trajectories and $\texttt{step_train} \geq \texttt{step_warmup} = 400$. The warmup is measured in training updates, not environment steps. These settings are fixed across tasks. The warmup delays use of the teacher but does not directly measure trajectory diversity or certify teacher quality. Before readiness, BMARL uses the uniform prior, as specified in Eq. 7, and thus, an untrained pseudo-expert does not regularize the actor. As a result, if no desirable trajectories have been found, the available undesirable samples retain entropy-based exploration.

For episode-level success or return criteria, the label is determined when the episode ends and is attached to replayed samples. No desirability-dependent update is applied to a trajectory before its

outcome is available. Appendix J examines the evolution of the teacher and actor once informative experience is available.

F.4 Training Time Analysis and Required Memory

Table 6 reports training time averaged over four seeds using the same GPU model, an NVIDIA RTX 4090. Relative to FOP, the reported mean runtimes increase by approximately 6% on average across `1c3s5z`, `5m_vs_6m`, and MMM2.

Table 6: Training time for each model in various SMAC maps (in hours).

Model	`1c3s5z`	`5m_vs_6m`	`MMM2`
FOP	16.3	12.8	17.2
DMAC	17.3	13.9	17.3
HACPF	28.6	20.6	29.0
BMARL	17.9	13.5	17.8

The additional pseudo-expert is one individual-policy network shared across agents. Its reported additional VRAM is 0.12 MiB on `5m_vs_6m` and 0.16 MiB on MMM2, compared with total FOP-backbone usage of 1.27 GiB and 1.85 GiB, respectively. These small network-storage figures should not be interpreted as a full accounting of optimizer states, temporary activations, or peak training memory.

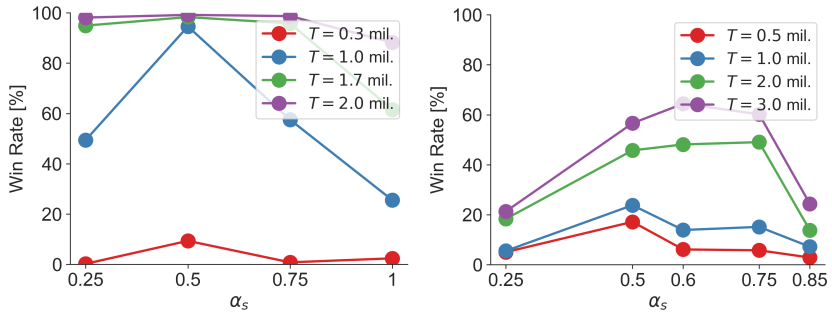
The desirable buffer resides in system RAM. Its storage grows linearly with capacity and trajectory size, approximately $O(|\mathcal{D}^*|\,\bar{T}\,d_{\mathrm{tr}})$, where $\bar{T}$ is the stored episode length and d_{tr} is the number of stored values per timestep, including agent-dependent fields. Larger teams, longer episodes, and higher-dimensional observations therefore increase its cost. Table 3 lists desirable buffer capacities of 500 or 1000 trajectories. The common setting of 1000 trajectories is one-fifth of the 5000-trajectory main replay buffer. A compact layout retaining only the fields needed for pseudo-expert training and desirability computation, such as local observations, actions, and returns or labels, reduced buffer storage by approximately 35% relative to copying the full replay structure. The periodic behavior-cloning workload is controlled by its update frequency, mini-batch size, and iterations per update; each update samples a fixed mini-batch rather than traversing the full buffer.

G ADDITIONAL EXPERIMENTAL RESULTS

In this section, we present the omitted experiment results.

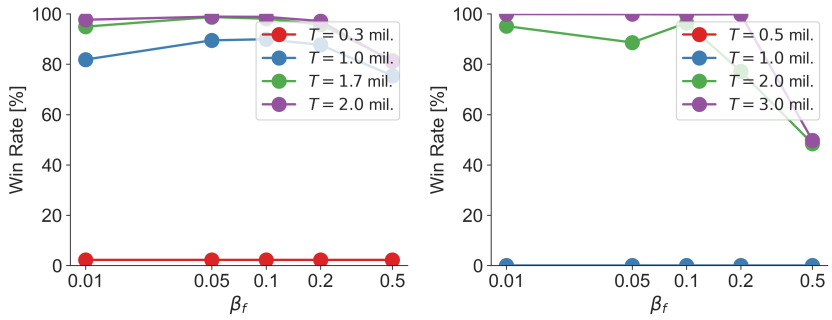
G.1 ADDITIONAL RESULTS FOR PARAMETRIC STUDY

Figures 14 and 15 illustrate the win rate according to training time and control parameters, α_s and β_f. In addition, Figures 16 and 17 describe the corresponding learning curves, including the $\beta_f = 0$ case.



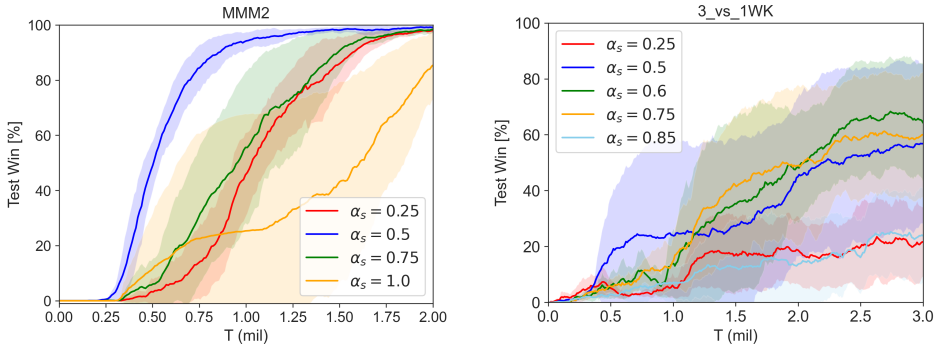
(a) MMM2 (b) 3_vs_1WK

Figure 14: Win rate according to α_s.



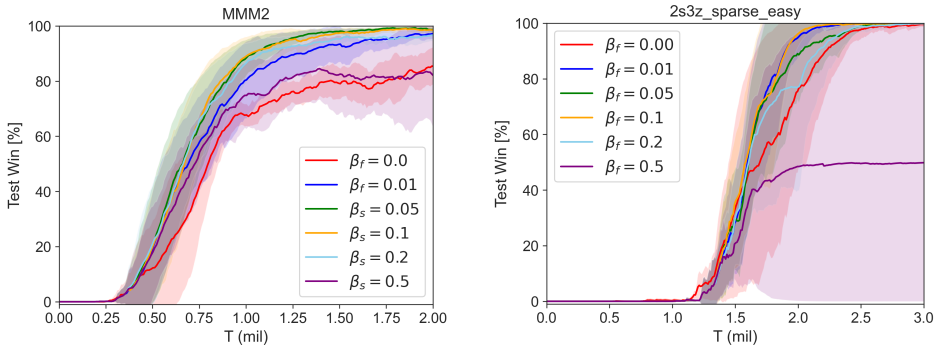
(a) MMM2 (b) 2s3z

Figure 15: Win rate according to β_f.



(a) $\bar{\mu}_w$ over α_s and training time T on `MMM2`. (b) $\bar{\mu}_w$ over α_s and training time T on `3_vs_1WK`.

Figure 16: Learning curves for the parametric analysis of α_s across different tasks.



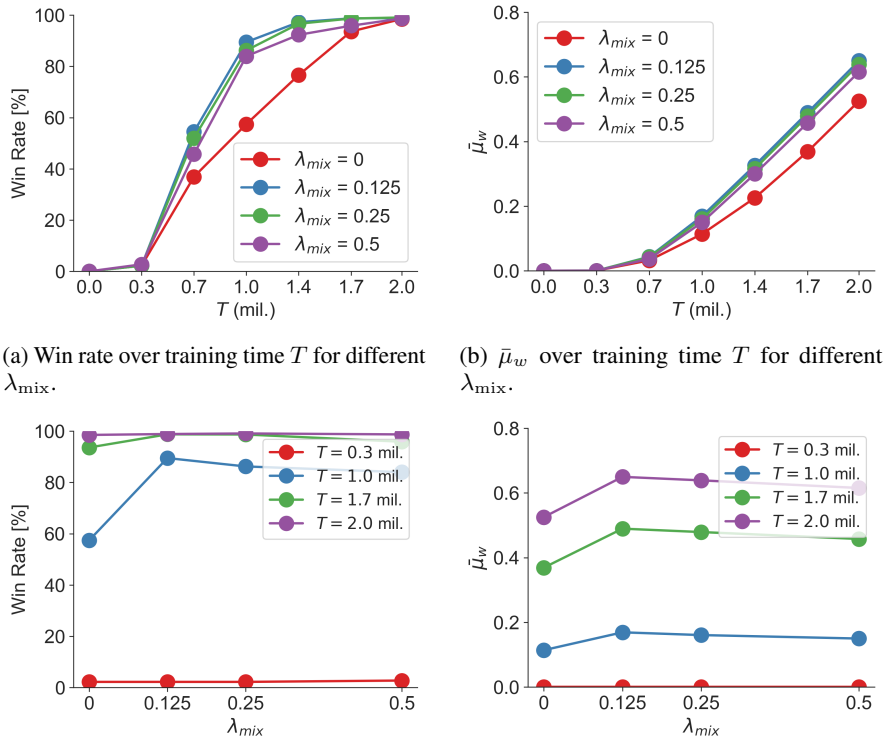
(a) $\bar{\mu}_w$ over β_f and training time T on `MMM2`.

(b) $\bar{\mu}_w$ over β_f and training time T on `2s3z_sparse_easy`.

Figure 17: Learning curves for the parametric analysis of β_f across different tasks.

G.2 Additional Parametric Study for the Mixing Ratio

In this Section, we conducted additional parametric studies for mixing ratio, i.e., λ_{mix}. Figure 18 summarizes its effect on `MMM2`. Overall, using samples from the success buffer improves both the win rate and the overall win rate $\bar{\mu}_w$, compared with the case without success-buffer mixing. The results suggest that a moderate value of λ_{mix} provides useful pseudo-expert guidance while avoiding excessive reliance on successful trajectories.



(a) Win rate over training time T for different λ_{mix}.

(b) $\bar{\mu}_w$ over training time T for different λ_{mix}.

(c) Win rate over λ_{mix} at different training times.

(d) $\bar{\mu}_w$ over λ_{mix} at different training times.

Figure 18: Parametric analysis of the success-buffer mixing ratio λ_{mix} on `MMM2`.

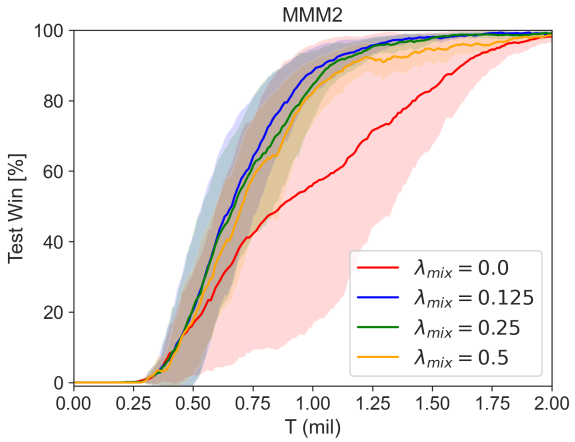
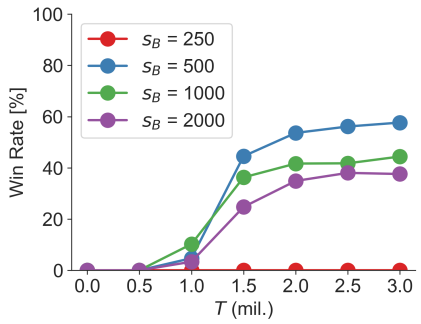


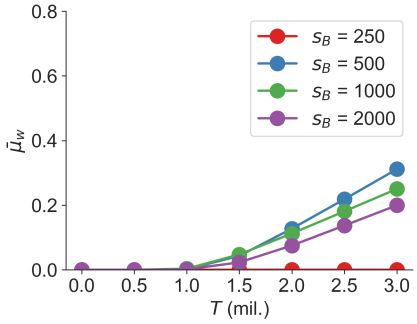
Figure 19: Learning curves for different success-buffer mixing ratios λ_{mix} on `MMM2`.

G.3 Additional Parametric Study for the Size of Desirable Buffer

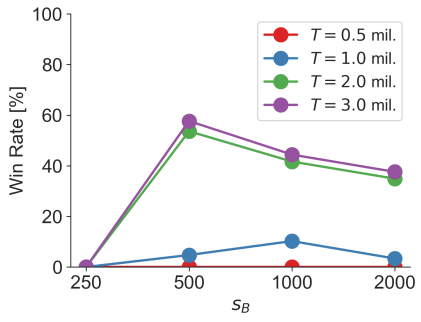
In this section, we conduct an additional parametric study for the size of the desirable buffer $\mathcal{D}^*$. Figures 20 and 21 summarize the effect of the success-buffer size s_B on `6h_vs_8z`. The results show that too small a success buffer provides insufficient pseudo-expert guidance, while excessively large buffers can degrade performance by retaining outdated or less relevant successful trajectories. A moderate buffer size yields better training efficiency and more stable performance.



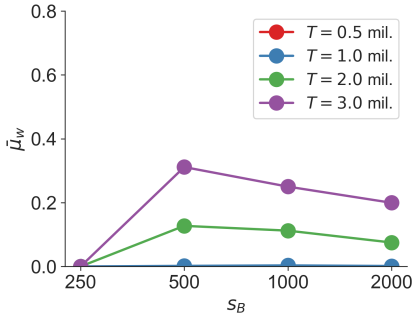
(a) Win rate over training time T for different s_B.



(b) $\bar{\mu}_w$ over training time T for different s_B.



(c) Win rate over s_B at different training times.



(d) $\bar{\mu}_w$ over s_B at different training times.

Figure 20: Parametric analysis of the success-buffer size s_B on `6h_vs_8z`.

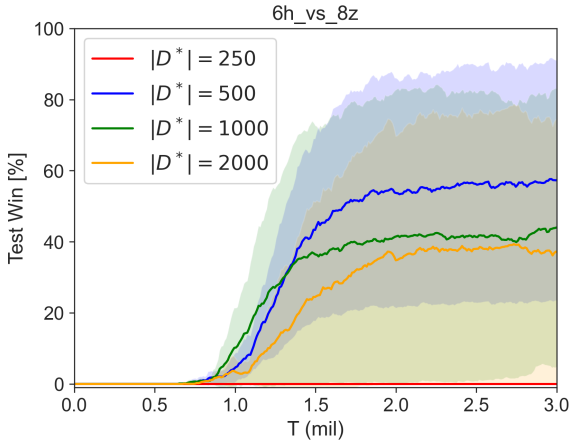


Figure 21: Learning curves for different success-buffer sizes s_B on `6h_vs_8z`.

G.4 Additional Evaluation on SMAX

We additionally evaluate the actor–critic methods BMARL, FOP, and DMAC in SMAX, which preserves a cooperative unit-combat setting without requiring the StarCraft II engine. Runs use six consecutive seeds for both `6h_vs_8z` and `2s3z`. Rewards were rescaled to be comparable to SMAC, to avoid the task-specific hyperparameter tuning. We reused the `6h_vs_8z` settings in Table 3 and the dense-reward `1c3s5z` settings for `2s3z`, without additional hyperparameter tuning. Table 7 summarizes the win-rate at the end of training and normalized AULC, and Figure 22 illustrates their learning curves.

Table 7: SMAX at 2M environment steps. Values are the mean ± sample standard deviation over six seeds. Both cases are dense reward settings. AULC is fractional.

Map	Method	Win rate at 2 M (%)	Normalized AULC ↑
`6h_vs_8z`	BMARL	$\mathbf{100.0 \pm 0.0}$	$\mathbf{0.688 \pm 0.039}$
	FOP	63.3 ± 45.4	0.421 ± 0.348
	DMAC	98.3 ± 1.5	0.602 ± 0.058
`2s3z`	BMARL	$\mathbf{99.7 \pm 0.8}$	0.760 ± 0.041
	FOP	97.8 ± 3.4	0.738 ± 0.040
	DMAC	96.2 ± 3.2	$\mathbf{0.791 \pm 0.042}$

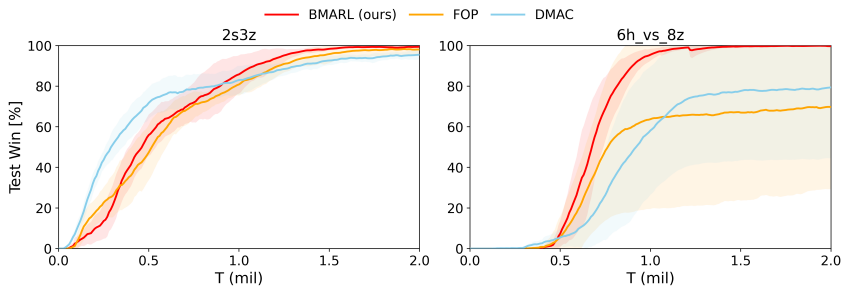


Figure 22: Learning curves on `2s3z` and `6h_vs_8z` in SMAX. Both tasks use dense reward settings, and we evaluate results over six consecutive seeds.

BMARL has the highest mean final win rate on both maps. It also has the highest mean AULC on `6h_vs_8z`, whereas DMAC has the highest AULC on dense-reward `2s3z`, but it converges

suboptimal policy compared to BMARL. The experiment therefore supports strong final performance without showing a universal sample-efficiency advantage.

H LIMITATIONS OF BMARL

Although BMARL provides additional flexibility by introducing a desirability-gated pseudo-expert prior, this flexibility also comes with additional hyperparameters and design choices. In particular, the strength of exploration with α, the strength of pseudo-expert regularization β, and the construction of the desirable buffer can all affect the balance between exploration and exploitation. As a result, finding a near-optimal configuration may require additional hyperparameter search compared with simpler entropy- or target-policy-regularized baselines.

To address this limitation, we used the same values for $\alpha_f = \beta_f$ across all tasks and fixed the hyperparameters related to training π_{PE}. Nevertheless, we observed consistent performance gains across a variety of tasks. In some sparse-reward settings, such as `2s3z_sparse`, where successful trajectories are initially rare, we conducted an additional hyperparameter search to achieve stable learning performance. In most cases, however, BMARL achieves strong performance without task-specific hyperparameter tuning, even across different types of tasks, including those without an explicit common goal, such as Overcooked and MABrax in JaxMARL (Rutherford et al., 2024), as presented in Appendix I.

I APPLICABILITY OF BMARL TO TASKS WITHOUT EXPLICIT SUCCESS SIGNALS

I.1 RETURN-BASED CRITERIA AND CONSERVATIVE USE OF PARTIAL PROGRESS

In BMARL, an environment-provided success signal is not mandatory to construct a binary desirability gate, by setting R_{thr} as defined in Definition 2.1. For an episode $\mathcal{T}$, we can define its return $R(\mathcal{T}) = \sum_{t=0}^{T-1} r_t$ and select samples using a threshold R_{thr}:

$$d(\mathcal{T}) = \mathbf{1}[R(\mathcal{T}) \geq R_{\text{thr}}]\,. \tag{36}$$

This generalizes the maximum-return criterion to tasks with graded returns. An explicit win or goal-scoring signal can still be used when available. Success and threshold exceedance select useful experience. The binary gating mechanism adopted by BMARL controls direct use of the prior, not all learning information in a trajectory. For example, returns $0.9R_{\text{thr}}$ and $0.1R_{\text{thr}}$ can receive the same undesirable label while inducing different rewards, TD targets, critic estimates, and value-aware actor updates. In the return-priority buffer, high-return but unsuccessful trajectories can also influence pseudo-expert training, as they are in $\mathcal{D}^*$, and thereby contribute useful coordination prefixes indirectly. The resulting π_{PE} regularizes the current policy π_θ only when a sampled trajectory activates the desirability gate. In contrast, a success-only FIFO buffer retains only trajectories that satisfy the success criterion. In Appendix K, we compare the binary desirability gating mechanism used in BMARL with a continuous gating mechanism, denoted as soft-desirability gating, as a possible alternative. The results show that soft-desirability gating is more prone to convergence to local optima.

I.2 ADAPTIVE DESIRABILITY GATING

To show how BMARL applies to tasks with no explicit desirability signal, we conducted additional experiments to evaluate how this definition works in BMARL on both binary (SMAC and SMACv2 tasks) and non-binary success tasks (Overcooked and MABrax tasks). Specifically, we developed two variants for incorporating a return-based desirability signal:

1) **Adaptive Desirability Gating Type I (aDG-I):** We continuously update R_{thr} using the return cutoff corresponding to the top $k_{\text{elite}}\%$ of trajectories in $\mathcal{D}^*$. We set $k_{\text{elite}} = 20$.

2) **Adaptive Desirability Gating Type II (aDG-II):** Starting from an initial desirability threshold R^0_{thr}, we update R_{thr} only when the return cutoff corresponding to the top $k_{\text{elite}}\%$ of trajectories in $\mathcal{D}^*$ exceeds R^0_{thr}. Thus, R^0_{thr} serves as the minimum desirability threshold. We set $k_{\text{elite}} = 20$.

For evaluation, let $q_{0.8}(k)$ denote the return cutoff for the top 20% of trajectories in the current return-priority buffer $\mathcal{D}^*$. We compare two variants: aDG-I continuously follows this upper quartile, whereas aDG-II imposes an initial minimum threshold R^0_{thr}:

$$\begin{aligned} R^{(2)}_{\text{thr}}(k) &= q_{0.8}(k), \\ R^{(3)}_{\text{thr}}(k) &= \max\{R^0_{\text{thr}}, q_{0.8}(k)\}. \end{aligned} \tag{37}$$

The configuration names aDG-I and aDG-II are retained in the tables. Please note that both variants use the binary threshold gate in Eq. 36, and they are distinct from the continuously valued Soft-DG gate in Appendix K. The minimum threshold in aDG-II limits premature trust in low-quality elite data.

Table 8: MMM2: success gating (DG) and adaptive return-threshold gating. Mean ± sample standard deviation over four seeds.

Metric	Config	0.25M	0.5M	0.75M	1.0M	1.5M	2.0M
Winrate	DG	1.55 ± 1.60	65.81 ± 23.19	$\mathbf{88.82 \pm 13.05}$	$\mathbf{97.15 \pm 2.81}$	$\mathbf{97.80 \pm 3.61}$	$\mathbf{99.14 \pm 1.00}$
	aDG-II	0.21 ± 0.17	$\mathbf{70.74 \pm 12.39}$	86.22 ± 14.89	91.64 ± 9.71	96.21 ± 4.52	98.26 ± 1.29
	aDG-I	$\mathbf{5.21 \pm 7.02}$	45.57 ± 29.22	78.70 ± 15.30	92.08 ± 4.81	95.44 ± 5.98	97.51 ± 2.21
AULC	DG	0.00 ± 0.00	$\mathbf{0.19 \pm 0.07}$	$\mathbf{0.40 \pm 0.07}$	$\mathbf{0.54 \pm 0.06}$	$\mathbf{0.68 \pm 0.05}$	$\mathbf{0.76 \pm 0.04}$
	aDG-II	0.00 ± 0.00	0.13 ± 0.06	0.33 ± 0.08	0.47 ± 0.07	0.63 ± 0.06	0.72 ± 0.05
	aDG-I	0.00 ± 0.00	0.13 ± 0.10	0.29 ± 0.11	0.44 ± 0.10	0.60 ± 0.08	0.70 ± 0.07
R_{thr}	aDG-II	10.86 ± 0.66	17.74 ± 0.25	20.04 ± 0.33	20.30 ± 0.11	20.42 ± 0.06	20.49 ± 0.07
	aDG-I	11.44 ± 0.71	17.56 ± 2.17	19.89 ± 0.80	20.38 ± 0.07	20.45 ± 0.01	20.50 ± 0.05

Table 9: `3m_sparse`: success gating and adaptive return-threshold gating. Mean ± sample standard deviation over four seeds.

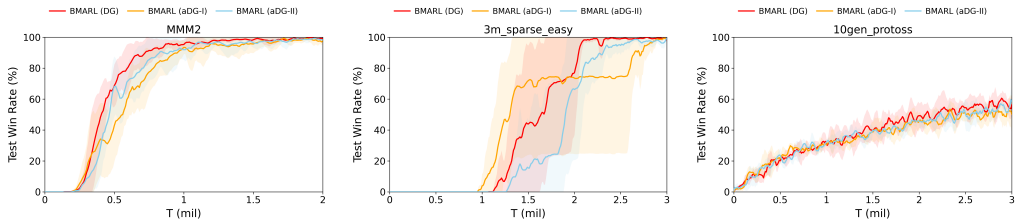
Metric	Config	0.5M	1.0M	1.5M	2.0M	2.5M	3.0M
Winrate	DG	0.00 ± 0.00	0.00 ± 0.00	45.35 ± 52.66	$\mathbf{80.96 \pm 31.12}$	$\mathbf{99.03 \pm 1.12}$	98.20 ± 2.11
	aDG-II	0.00 ± 0.00	0.00 ± 0.00	15.09 ± 30.17	67.35 ± 46.86	96.24 ± 5.20	$\mathbf{99.43 \pm 1.14}$
	aDG-I	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{73.93 \pm 49.30}$	74.59 ± 49.73	75.00 ± 50.00	98.86 ± 1.34
AULC	DG	0.00 ± 0.00	0.00 ± 0.00	0.05 ± 0.07	0.19 ± 0.15	0.35 ± 0.13	0.46 ± 0.11
	aDG-II	0.00 ± 0.00	0.00 ± 0.00	0.02 ± 0.04	0.09 ± 0.12	0.24 ± 0.12	0.37 ± 0.11
	aDG-I	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{0.14 \pm 0.11}$	$\mathbf{0.29 \pm 0.20}$	$\mathbf{0.38 \pm 0.25}$	$\mathbf{0.47 \pm 0.24}$
R_{thr}	aDG-II	5.00 ± 0.00	5.00 ± 0.00	6.90 ± 3.80	10.70 ± 3.80	12.60 ± 0.00	12.60 ± 0.00
	aDG-I	0.00 ± 0.00	3.42 ± 5.94	9.59 ± 6.03	9.59 ± 6.03	9.66 ± 5.89	12.60 ± 0.00

Table 10: `10gen_protoss`: success gating and adaptive return-threshold gating. Mean ± sample standard deviation over four seeds.

Metric	Config	0.5M	1.0M	1.5M	2.0M	2.5M	3.0M
Winrate	DG	24.77 ± 0.88	27.54 ± 2.34	38.89 ± 10.02	$\mathbf{51.49 \pm 8.68}$	55.61 ± 6.34	56.44 ± 6.85
	aDG-II	$\mathbf{25.88 \pm 5.32}$	$\mathbf{34.49 \pm 5.07}$	37.29 ± 2.76	48.20 ± 5.37	$\mathbf{57.71 \pm 12.87}$	$\mathbf{58.34 \pm 3.22}$
	aDG-I	19.02 ± 6.67	30.89 ± 3.55	$\mathbf{40.86 \pm 7.06}$	44.59 ± 7.19	48.94 ± 4.06	50.08 ± 12.60
AULC	DG	0.11 ± 0.03	0.19 ± 0.03	0.24 ± 0.04	$\mathbf{0.30 \pm 0.05}$	$\mathbf{0.34 \pm 0.05}$	$\mathbf{0.38 \pm 0.05}$
	aDG-II	0.12 ± 0.03	0.19 ± 0.03	0.24 ± 0.02	0.29 ± 0.02	0.33 ± 0.02	0.36 ± 0.03
	aDG-I	$\mathbf{0.12 \pm 0.04}$	$\mathbf{0.20 \pm 0.02}$	$\mathbf{0.25 \pm 0.01}$	0.29 ± 0.01	0.33 ± 0.02	0.35 ± 0.02
R_{thr}	aDG-II	23.03 ± 2.03	26.50 ± 2.39	26.59 ± 2.35	26.73 ± 2.32	26.78 ± 2.36	26.78 ± 2.37
	aDG-I	21.95 ± 2.25	25.28 ± 2.71	25.40 ± 2.67	25.57 ± 2.70	25.58 ± 2.69	25.61 ± 2.72

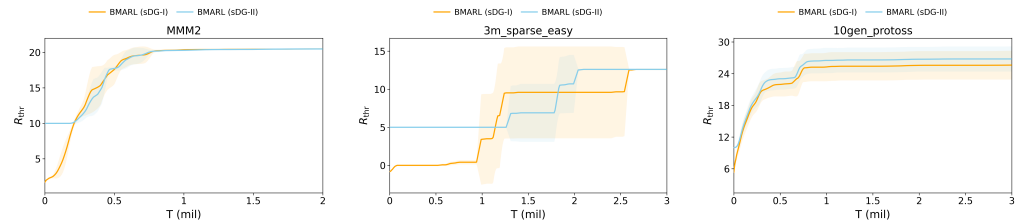
Here, aDG-I exploits high-return partial progress more actively, but its effects depend on the task. On `3m_sparse`, its mean win rate remains near 75% from 1.5M to 2.5M steps before recovering to $98.86 \pm 1.34\%$ at 3M. Its final AULC is 0.47 ± 0.24, compared with 0.46 ± 0.11 for DG and 0.37 ± 0.11 for aDG-II. This is a temporary plateau in the aggregate learning curve, not evidence of permanent convergence to a suboptimal policy. On MMM2, DG has the highest final mean and AULC. On `10gen_protoss`, aDG-II has a higher final mean ($58.34 \pm 3.22\%$) than DG ($56.44 \pm 6.85\%$), but a lower AULC (0.36 ± 0.03 versus 0.38 ± 0.05). The variants thus expose a trade-off between early reuse and conservative selection, rather than one uniformly best threshold rule. Regarding these characteristics, we consider aDG-II as default for tasks without explicit success signals.

Figures 23 illustrate the learning curves with the aforementioned tasks. In addition, we also present the evolving R_{thr} for aDG-II and aD-II in Figures 24.



(a) MMM2 (b) `3m_sparse` (c) `10gen_protoss`

Figure 23: Learning curves for tasks without explicit success signals



(a) MMM2 (b) `3m_sparse` (c) `10gen_protoss`

Figure 24: Adaptive R_{thr} over training time

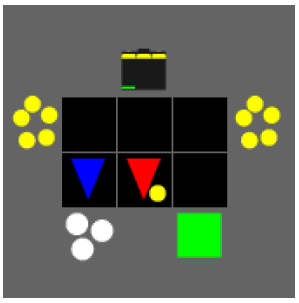
Likewise, BMARL can effectively incorporate adaptive return thresholding to provide a practical definition of desirability.

I.3 Evaluation on Tasks without Explicit Success Signals

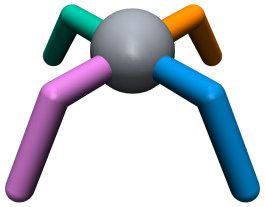
Environment examples. Figure 25 illustrates two cooperative domains supported by JaxMARL (Rutherford et al., 2024). Overcooked is a cooperative cooking environment in which two agents collect ingredients, prepare soup, and deliver completed dishes for a shared reward (Carroll et al., 2019). In the Cramped Room layout, the agents must coordinate their movements and use of shared workstations within a compact kitchen. Although successful deliveries produce rewards, they do not by themselves define a binary success label for an entire episode. Instead, performance is naturally measured by accumulated return.

Multi-Agent MuJoCo (MaMuJoCo), introduced with FACMAC (Peng et al., 2021), constructs cooperative continuous-control tasks by assigning different subsets of a robot's actuated joints to agents with local observations. JaxMARL's Multi-Agent Brax (MABrax) provides a Brax-based analogue of these tasks (Rutherford et al., 2024). Brax is a differentiable rigid-body simulator written in JAX that supports parallel simulation on accelerators (Freeman et al., 2021). In Ant 4×2 (`ant_4x2`), four agents each control the two actuated joints of one leg and cooperate under a shared locomotion reward. MABrax uses this multi-agent factorization with Brax physics.

Figures 26 illustrates the comparative evaluation on Overcooked and MABrax. For MABrax, the only modification is to apply the policy for a continuous action space. Surprisingly, the same



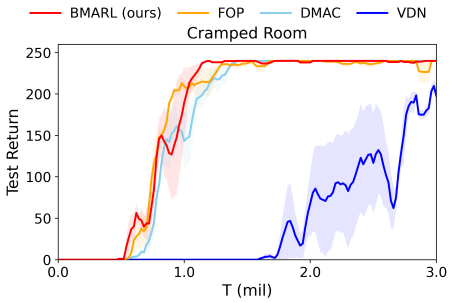
(a) Overcooked: Cramped Room



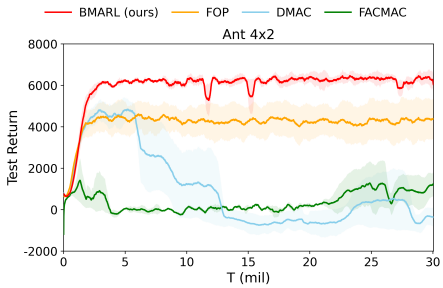
(b) MABrax: Ant 4×2

Figure 25: Examples of cooperative tasks in JaxMARL (Rutherford et al., 2024). Left: a frame from the official Cramped Room visualization where the red and blue triangles represent the two agents. Right: an illustration of the official Brax Ant model (Freeman et al., 2021) with the JaxMARL 4×2 partition. Each colored leg is controlled by one agent through two actuators.

hyperparameter settings from MMM2 in Table 3 were adopted and showed the strongest performance compared to other baseline methods. As demonstrated by these examples, BMARL can be applied to tasks without explicit success signals, with little effort required for hyperparameter tuning.



(a) Overcooked: Cramped Room



(b) MABrax: Ant 4×2

Figure 26: Learning curves for tasks without explicit success signals

J AN EVOLVING PSEUDO-EXPERT AND CHECKPOINT DIAGNOSTICS

BMARL does not freeze the first successful policy as a permanent prior for guidance. Instead, the return-priority buffer replaces lower-return trajectories as better experiences are discovered during training. Undesirable samples continue to contribute to learning through the entropy branch. These mechanisms can reduce dependence on early behaviors, although improvement of the pseudo-expert is not guaranteed solely by higher buffer returns. Before sufficient data are available, the fallback rule in Appendix F.3 prevents the use of an inadequately trained pseudo-expert.

To qualitatively analyze this mechanism, we compare the KL-divergence of the current pseudo-expert and learning policy from the best policy, e.g., $D_{\mathrm{KL}}(\pi^*\|\pi_{PE}^t)$ and $D_{\mathrm{KL}}(\pi^*\|\pi_\theta^t)$. For the best policy π^*, we consider both $\pi_\theta^{T_f}$ and $\pi_{PE}^{T_f}$ in this evaluation. For this analysis, we used replay buffers collected at different training steps, $T_{\mathrm{env}} \in \{500\mathrm{K}, 2000\mathrm{K}\}$, because the trajectory distributions may differ between the intermediate and final stages of training. † indicates self-KL divergence. We use the same task (MMM2 task on SMAC) presented in Section 4.5 for this evaluation.

For intermediate checkpoints $t < T_f$, the current policy $\pi_\theta^{(t)}$ is generally closer to the final pseudo-expert $\pi_{\mathrm{PE}}^{(T_f)}$ than to the final actor $\pi_\theta^{(T_f)}$ under the reported forward KL. More importantly, $D_{\mathrm{KL}}(\pi_{\mathrm{PE}}^{(T_f)}\|\pi_\theta^{(t)})$ consistently decreases throughout BMARL training, whereas it remains

Table 11: 500K replay buffer, BMARL (No-PE + No-DG) candidates on desirable trajectories. Values are forward KL in nats.

$D_{\mathrm{KL}}(\pi_{\mathrm{reference}}\|\pi_{\mathrm{candidate}})$	250K	500K	750K	1000K	1500K	2000K
$D_{\mathrm{KL}}(\pi_\theta^{(T_f)}\|\pi_\theta^{(t)})$	4.123	4.430	4.632	4.518	4.166	3.996
$D_{\mathrm{KL}}(\pi_{\mathrm{PE}}^{(T_f)}\|\pi_\theta^{(t)})$	1.528	1.700	1.801	1.800	1.712	1.654

Table 12: 500K replay buffer, BMARL (ours) candidates on desirable trajectories. Values are forward KL in nats.

$D_{\mathrm{KL}}(\pi_{\mathrm{reference}}\|\pi_{\mathrm{candidate}})$	250K	500K	750K	1000K	1500K	2000K
$D_{\mathrm{KL}}(\pi_\theta^{(T_f)}\|\pi_\theta^{(t)})$	3.637	1.223	0.754	0.505	0.351	$0.000^\dagger$
$D_{\mathrm{KL}}(\pi_\theta^{(T_f)}\|\pi_{\mathrm{PE}}^{(t)})$	3.666	2.123	1.134	1.019	0.898	0.809
$D_{\mathrm{KL}}(\pi_{\mathrm{PE}}^{(T_f)}\|\pi_\theta^{(t)})$	1.598	0.700	0.522	0.327	0.371	0.354
$D_{\mathrm{KL}}(\pi_{\mathrm{PE}}^{(T_f)}\|\pi_{\mathrm{PE}}^{(t)})$	1.278	0.654	0.205	0.111	0.062	$0.000^\dagger$

Table 13: 2000K replay buffer, BMARL (No-PE + No-DG) candidates on desirable trajectories. Values are forward KL in nats.

$D_{\mathrm{KL}}(\pi_{\mathrm{reference}}\|\pi_{\mathrm{candidate}})$	250K	500K	750K	1000K	1500K	2000K
$D_{\mathrm{KL}}(\pi_\theta^{(T_f)}\|\pi_\theta^{(t)})$	4.907	5.083	6.387	6.608	5.946	5.116
$D_{\mathrm{KL}}(\pi_{\mathrm{PE}}^{(T_f)}\|\pi_\theta^{(t)})$	2.110	2.398	3.022	3.134	2.802	2.393

Table 14: 2000K replay buffer, BMARL (ours) candidates on desirable trajectories. Values are forward KL in nats.

$D_{\mathrm{KL}}(\pi_{\mathrm{reference}}\|\pi_{\mathrm{candidate}})$	250K	500K	750K	1000K	1500K	2000K
$D_{\mathrm{KL}}(\pi_\theta^{(T_f)}\|\pi_\theta^{(t)})$	3.603	1.576	0.388	0.190	0.114	$0.000^\dagger$
$D_{\mathrm{KL}}(\pi_\theta^{(T_f)}\|\pi_{\mathrm{PE}}^{(t)})$	3.190	1.593	0.445	0.293	0.227	0.175
$D_{\mathrm{KL}}(\pi_{\mathrm{PE}}^{(T_f)}\|\pi_\theta^{(t)})$	1.750	0.874	0.184	0.099	0.092	0.073
$D_{\mathrm{KL}}(\pi_{\mathrm{PE}}^{(T_f)}\|\pi_{\mathrm{PE}}^{(t)})$	1.342	0.806	0.122	0.052	0.034	$0.000^\dagger$

high or even increases in the No-PE + No-DG ablation. This contrast indicates that the pseudo-expert progressively aligns the learning policy with a stable desirable-policy distribution, thereby facilitating its subsequent convergence to $\pi_\theta^{(T_f)}$. In other words, since $\mathcal{D}^*$ is continually updated with higher-return trajectories, π_{PE} does not converge to an arbitrary fixed policy. Instead, it tracks the progressively improving elite-policy distribution and approaches the optimal policy as better trajectories are discovered.

K Comparison of Binary and Soft Desirability Gating

We compare the binary gate used in BMARL with an alternative continuous score, $g(\mathcal{T}) \in [0, 1]$. This evaluation aims to validate the use of the binary gating mechanism over possible continuous alternatives. For each agent, Soft-DG geometrically interpolates between the pseudo-expert and a uniform prior over available actions:

$$\begin{aligned} \rho_{g,i}(a_i \mid \tau_i) &= \frac{\pi_{PE,i}(a_i \mid \tau_i)^{g(\mathcal{T})}\mathcal{U}_i(a_i \mid \tau_i)^{1-g(\mathcal{T})}}{Z_{g,i}(\tau_i)}, \\ g(\mathcal{T}) &= \mathrm{clip}\left(\frac{R(\mathcal{T}) - q_{\mathrm{low}}}{q_{\mathrm{high}} - q_{\mathrm{low}}}, 0, 1\right), \end{aligned} \tag{38}$$

where $Z_{g,i}$ normalizes the action probabilities and we set gate parameters as $q_{\text{low}} = 0$ and $q_{\text{high}} = 0.99$. The joint prior factorizes across agents. Once the pseudo-expert is ready, $g = 0$ selects the uniform prior, and $g = 1$ selects the pseudo-expert, recovering the two endpoints of the binary gate. The same fallback rule applies before readiness as described in Appendix F.3. Unlike the threshold variants in Appendix I, behavior patterns of π_{PE} in undesirable trajectories directly affect training of the current policy π_θ via Eq. 18. Tables 15–16 present the win-rate and return comparisons between BMARL with binary desirability gating and its soft-gating variant (denoted as Soft-DG), respectively.

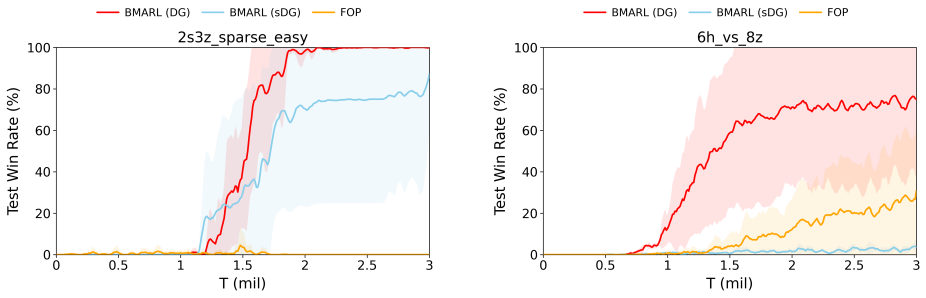
Table 15: Win rate (%) under binary gating, continuous Soft-DG, and FOP. Mean ± sample standard deviation over four seeds.

Task	Method	0.5M	1.0M	1.5M	2.0M	2.5M	3.0M
2s3z_sparse	BMARL (ours)	0.00 ± 0.00	0.00 ± 0.00	$\mathbf{35.94 \pm 32.66}$	$\mathbf{97.66 \pm 4.06}$	$\mathbf{100.00 \pm 0.00}$	$\mathbf{100.00 \pm 0.00}$
	BMARL (Soft-DG)	0.00 ± 0.00	0.00 ± 0.00	31.25 ± 39.59	70.31 ± 41.01	75.00 ± 43.30	85.94 ± 24.36
	FOP	$\mathbf{2.34 \pm 4.06}$	0.00 ± 0.00	3.12 ± 5.41	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
6h_vs_8z	BMARL (ours)	0.00 ± 0.00	$\mathbf{14.84 \pm 8.08}$	$\mathbf{57.03 \pm 26.83}$	$\mathbf{67.97 \pm 29.51}$	$\mathbf{71.09 \pm 32.20}$	$\mathbf{76.56 \pm 35.25}$
	BMARL (Soft-DG)	0.00 ± 0.00	0.78 ± 1.35	0.78 ± 1.35	1.56 ± 1.56	4.69 ± 3.49	2.34 ± 1.35
	FOP	0.00 ± 0.00	2.34 ± 4.06	3.91 ± 6.77	9.38 ± 11.48	21.09 ± 22.47	35.94 ± 36.01

Table 16: Episode return under binary gating, continuous Soft-DG, and FOP. Mean ± sample standard deviation over four seeds.

Task	Method	0.5M	1.0M	1.5M	2.0M	2.5M	3.0M
2s3z_sparse	BMARL (ours)	-0.44 ± 0.04	-0.42 ± 0.11	$\mathbf{1.73 \pm 1.51}$	$\mathbf{4.67 \pm 0.24}$	$\mathbf{4.84 \pm 0.03}$	$\mathbf{4.84 \pm 0.02}$
	BMARL (Soft-DG)	-0.45 ± 0.03	-0.37 ± 0.10	1.49 ± 1.86	3.28 ± 2.09	3.56 ± 2.19	4.17 ± 1.19
	FOP	$\mathbf{0.01 \pm 0.30}$	$\mathbf{-0.07 \pm 0.07}$	-0.08 ± 0.43	-0.18 ± 0.19	-0.24 ± 0.14	-0.26 ± 0.02
6h_vs_8z	BMARL (ours)	$\mathbf{10.88 \pm 0.71}$	$\mathbf{14.97 \pm 0.68}$	$\mathbf{17.98 \pm 1.25}$	$\mathbf{18.80 \pm 0.98}$	$\mathbf{18.82 \pm 1.28}$	$\mathbf{19.27 \pm 1.03}$
	BMARL (Soft-DG)	9.98 ± 0.32	11.46 ± 0.52	12.05 ± 0.25	12.40 ± 0.25	12.51 ± 0.42	12.96 ± 0.46
	FOP	9.92 ± 0.23	12.31 ± 0.98	13.30 ± 1.13	14.15 ± 1.53	15.00 ± 2.18	15.51 ± 3.14

On 2s3z_sparse, Soft-DG reaches a final win rate of $85.94 \pm 24.36\%$, above FOP (0%) but below binary-gated BMARL (100%). Its final return is also lower, 4.17 ± 1.19 versus 4.84 ± 0.02. On 6h_vs_8z, Soft-DG reaches only $2.34 \pm 1.35\%$ win rate and 12.96 ± 0.46 return, compared with $35.94 \pm 36.01\%$ and 15.51 ± 3.14 for FOP, and $76.56 \pm 35.25\%$ and 19.27 ± 1.03 for binary-gated BMARL. These results are consistent with premature reinforcement of incomplete coordination in that setting. They support retaining conservative binary gating when a reliable completion criterion exists, but do not imply that every continuous gating design will fail. Figures 27 illustrate the corresponding learning curves. Interestingly, on the 2s3z_sparse task, Soft-DG exhibits more efficient learning during the early training phase, achieving a higher win rate at around $T_{\text{env}} = 1.25$M. However, it also shows substantially higher variance across seeds, suggesting that some runs may converge to suboptimal solutions. In contrast, binary desirability gating yields more robust performance across seeds.



(a) 2s3z_sparse (b) 6h_vs_8z

Figure 27: Comparison of learning curves with binary and soft desirability gating