COORDINATED DENOISING STABILIZES COOPERATION IN DIFFUSION MULTI-AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion policies are promising for cooperative multi-agent reinforcement learning (MARL) because they can represent multimodal joint action distributions. However, existing methods supervise only the final executed action, leaving intermediate denoising steps unguided. This causes poor coordination when agents execute their denoising processes independently. We propose DiMAC, a novel diffusion-based offline MARL framework that injects centralized coordination supervision directly into the denoising process. The core component is a Denoising Process Critic (DPC) that evaluates intermediate denoising states according to their expected contribution to the eventual coordinated joint action. DiMAC further employs coalition value decomposition to translate centralized process values into decentralized agent-wise and pairwise guidance signals, along with reliability weighting to emphasize more stable denoising steps. This design preserves the Centralized Training with Decentralized Execution (CTDE) paradigm while optimizing coordination throughout the entire action generation process. Experiments on MPE show clear improvements over prior offline MARL and diffusion-based methods, while results on SMAC remain competitive with strong baselines. Code is available at: `https://anonymous.4open.science/r/DiMAC-123321/`

1 INTRODUCTION

Cooperative multi-agent reinforcement learning (MARL) has demonstrated remarkable success across a wide range of complex coordination tasks. Nevertheless, a persistent challenge lies in learning effective policies for environments where the optimal joint action distribution is inherently multimodal. For instance, when two agents must reach the same target location, moving together to point A or point B may both achieve the maximum team return, representing two separate coordination strategies. While conventional MARL methods typically employ direct policy parameterizations that generate actions in a single forward pass—rendering them inadequate for capturing such multimodality—diffusion policies offer a more expressive framework. By generating actions through an iterative denoising process, they are naturally suited to model diverse yet coordinated team behaviors (Ho et al., 2020; Song et al., 2020; Song & Ermon, 2019; Lu et al., 2025).

Despite their representational advantages, diffusion policies introduce significant complexity in learning optimal joint policies. Specifically, developing an effective diffusion-based MARL algorithm faces the following challenges. First, the requirement for multiple denoising steps per decision increases the computational cost of online interaction compared to conventional policies such as MLPs. Second, while feedback is only available for the final executed joint action, the intermediate denoising steps lack explicit supervision (Foerster et al., 2018; Rashid et al., 2020). Third, at inference time, agents must execute their denoising processes independently based solely on local observations, making it difficult to maintain global coordination throughout the generation process.

Existing approaches fall short of addressing these specific challenges. Traditional offline MARL methods, such as MA-CQL (Shao et al., 2023), MAICQ (Yang et al., 2021), OMAR (Pan et al., 2022) and MADT (Meng et al., 2023), optimize executed actions or sequences but do not model the internal coordination of an iterative generative policy. While single-agent diffusion RL methods enhance expressiveness or stability, they fail to address coordination among parallel denoising agents (Lu et al.,

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.

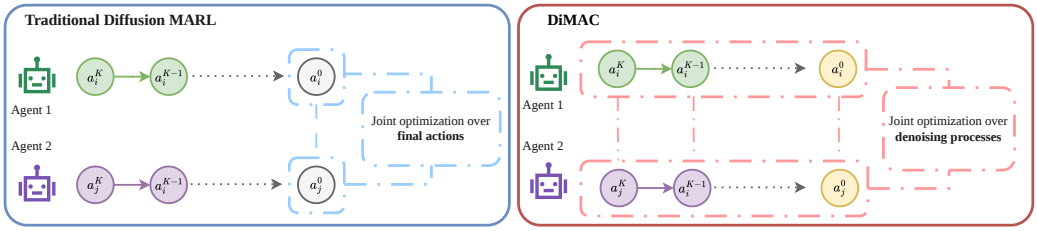


Figure 1: Comparison between traditional diffusion MARL methods and DiMAC. Traditional methods focus on optimizing joint actions, leaving the denoising processes uncontrolled. DiMAC optimizes the joint denoising processes, thus achieving more stable cooperation during execution.

2025; Gao et al., 2025). Recent diffusion-based MARL methods, including MADiff (Zhu et al., 2024) and DoF (Li et al., 2025), improve joint modeling or decentralized factorization but primarily focus on the quality of the final action, neglecting explicit coordination supervision during the denoising process. Consequently, even with the Centralized Training and Decentralized Execution (CTDE) framework, decentralized denoising processes can easily lose control and result in suboptimal joint policies.

To bridge this gap, we propose **DiMAC**, a **di**ffusion-based framework for offline **m**ulti-**a**gent **c**ooperation, which injects centralized coordination supervision directly into the denoising process. Our core insight, as illustrated in Figure 1, is to evaluate intermediate denoising states based on their expected contribution to the eventual coordinated joint action, rather than supervising only the final output. DiMAC consists of three key innovations. First, it learns a Denoising Process Critic (DPC) to provide process-level evaluation for intermediate denoising states, transforming terminal team-level feedback into supervision for the denoising process itself. Second, it translates this centralized evaluation into decentralized coordination guidance, ensuring that locally executed denoising updates remain aligned with team-level objectives. Third, DiMAC further accounts for the varying importance of denoising steps, emphasizing more stable guidance. In this manner, DiMAC preserves decentralized execution while optimizing coordination throughout the entire action generation process.

In summary, the main contributions of this paper are as follows:

- **A Process-Aware Diffusion MARL Framework:** We propose DiMAC, a novel offline MARL framework with a dedicated Denoising Process Critic (DPC) that evaluates intermediate denoising states. By this way, terminal team-level feedback is transformed into step-wise supervision signals to guide the internal generation process, thereby optimizing coordination throughout the entire action generation process.
- **Stabilized Decentralized Coordination**: We unify denoising-process-level evaluation with step importance weighting to provide robust decentralized guidance. This ensures that locally executed denoising updates remain aligned with global team objectives and maintains stable, reliable coordination among agents during the inference phase.
- **Comprehensive Evaluation of DiMAC on Diverse Benchmarks:** We evaluate DiMAC on MPE and SMAC against strong offline MARL and diffusion-based baselines. DiMAC achieves substantial and consistent improvements across diverse tasks, and further analysis shows that denoising-process supervision is the main source of these gains.

2 Related Works

Multi-Agent Reinforcement Learning Frameworks. In cooperative multi-agent reinforcement learning (MARL), value decomposition methods such as VDN (Sunehag et al., 2017) and QMIX (Rashid et al., 2020) established the foundation for the Centralized Training and Decentralized Execution (CTDE) paradigm. Following this, actor-critic methods such as MAPPO (Yu et al., 2022) and HATRPO (Kuba et al., 2021) improved optimization stability through trust-region style policy updates. In offline MARL, the central challenge becomes controlling extrapolation error under fixed datasets. Representative methods include conservative and implicitly constrained approaches such as MA-CQL (Shao et al., 2023), MAICQ (Yang et al., 2021), and OMAR (Pan et al., 2022), as well

as sequence-modeling approaches such as MADT (Meng et al., 2023). While these methods are effective, they usually optimize Gaussian, categorical, or sequence-level action models directly at the executed action, which makes it difficult to represent highly multimodal cooperative behavior while also assigning terminal coordination information back through a multi-step generative process.

Diffusion-Based Single-Agent Decision Making. Owing to their exceptional capacity to capture complex, multi-modal data distributions, Diffusion Models (DMs) (Ho et al., 2020) are rapidly emerging as a new paradigm for sequential decision-making. In single-agent reinforcement learning, pioneering works such as Diffuser (Janner et al., 2022) and Diffusion-QL (Wang et al., 2022) achieved strong performance by effectively reframing the policy learning process as a generative task, leveraging the ability of diffusion models to iteratively denoise random noise into optimal trajectories. Subsequent methods, including BDPO (Gao et al., 2025), further refined this approach by introducing regularization mechanisms during each denoising step to strictly constrain the policy within the support of the behavioral dataset, thereby improving offline stability and mitigating distributional shift. However, the complexity of diffusion policies makes it hard to extend them to the multi-agent setting, where non-stationarity of other agents can hardly be captured by the diffusion process.

Multi-Agent Diffusion Policies and Coordination. Recently, diffusion and flow-based generative models have been introduced to the multi-agent domain to address complex coordination challenges. MADiff (Zhu et al., 2024) models joint multi-agent trajectories and uses attention mechanisms to capture coordination patterns, while DoF (Li et al., 2025) proposes a diffusion factorization framework to facilitate decentralized execution. OM2P (Li et al., 2026) develops an offline multi-agent mean-flow policy, and MAC-Flow (Lee et al., 2026) explores coordination through flow matching, demonstrating the potential of generative modeling in multi-agent control. These approaches have shown the potential of diffusion models in capturing the multimodal nature of joint policies. However, they primarily focus on achieving coordination at the level of final actions, and often converge to suboptimal policies due to the lack of effective guidance on decentralized denoising processes.

3 PRELIMINARIES

Multi-Agent Reinforcement Learning. We consider a fully cooperative multi-agent task, which can be modeled as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Oliehoek et al., 2016) defined by a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \Omega, \mathcal{O}, \gamma, N \rangle$. Here, N is the number of agents, $s \in \mathcal{S}$ is the global state. At each timestep t, each agent $i \in \{1, \dots, N\}$ receives a local observation $o_i \in \Omega$ through the observation function $\mathcal{O}(s, i)$. Each agent chooses an action $a_i \in \mathcal{A}$, forming a joint action $\boldsymbol{a} = \{a_1, \dots, a_N\}$. The environment transitions to s' according to $\mathcal{P}(s'|s, \boldsymbol{a})$ and provides a team reward $r = r(s, \boldsymbol{a})$. The goal is to find a joint policy $\boldsymbol{\pi} = \prod_i \pi_i(a_i|o_i)$ that maximizes the expected discounted return $J(\boldsymbol{\pi}) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$.

Diffusion Models for Decision Making. Diffusion models (Ho et al., 2020; Sohl-Dickstein et al., 2015; Song & Ermon, 2019) represent a policy $\pi_\theta(a|o)$ by learning a reverse denoising process. Starting from Gaussian noise $a^K \sim \mathcal{N}(0, \mathbf{I})$, the model iteratively produces cleaner samples a^{k-1} following:

$$a^{k-1} = \frac{1}{\sqrt{\alpha_k}} \left(a^k - \frac{1 - \alpha_k}{\sqrt{1 - \bar{\alpha}_k}} \epsilon_\theta(a^k, o, k) \right) + \sigma_k \epsilon \tag{1}$$

where ϵ_θ is a neural network trained via the objective (Ho et al., 2020):

$$\mathcal{L}_{diff}(\theta) = \mathbb{E}_{a,o,\epsilon,k}[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_k} a + \sqrt{1 - \bar{\alpha}_k}\epsilon, o, k)\|^2] \tag{2}$$

In MARL setting, a joint diffusion policy should capture complex multimodal action distributions and inter-agent dependencies. However, achieving this is nontrivial, because coordination must be preserved throughout an iterative denoising process rather than enforced only at the final joint action.

4 METHODS

Motivation. Under CTDE, the quality of a diffusion policy depends on whether independently executed local denoising chains remain compatible throughout generation, not merely on the final joint action they eventually produce. When supervision is attached only to the endpoint, centralized

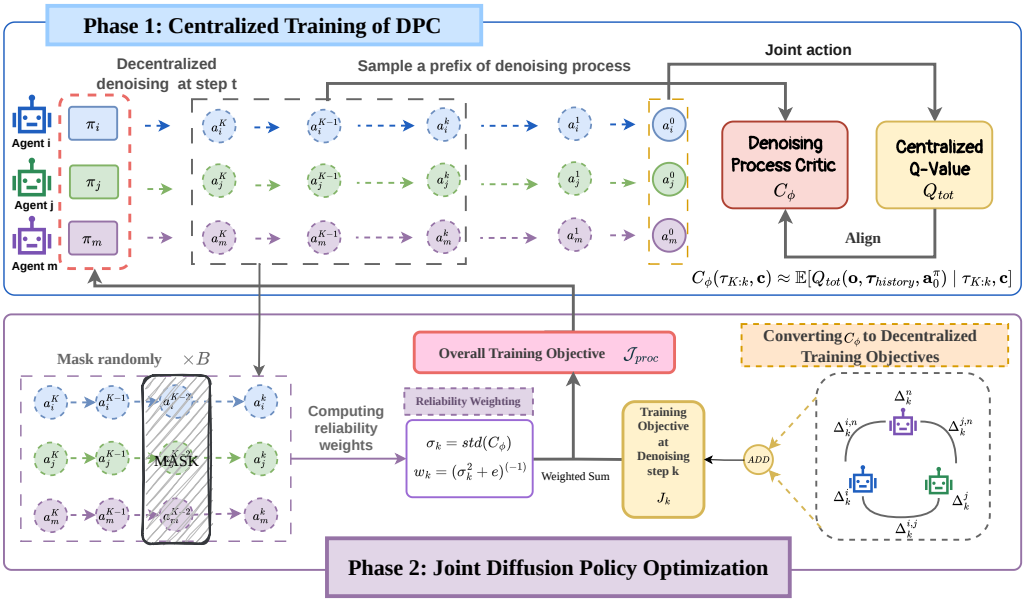


Figure 2: **Overview of DiMAC**. Under CTDE, each agent denoises from local observations, while training uses joint information. DiMAC has two phases. In Phase 1, sampled denoising prefixes are used to train the Denoising Process Critic (DPC), which estimates denoising-process values from centralized final-action values. In Phase 2, another sampled prefix is used for policy optimization, where coalition value decomposition and reliability weighting turn DPC evaluation into decentralized learning signals for the diffusion policy.

coordination information arrives too late to regulate the intermediate denoising decisions that shape decentralized behavior. DiMAC addresses this issue by learning a centralized evaluator over denoising prefixes and using it to turn terminal coordination feedback into process-level supervision for the reverse chain itself.

Overview. Figure 2 shows DiMAC as a two-phase training pipeline. In Phase 1, we sample denoising prefixes from the current decentralized diffusion policy and train a Denoising Process Critic (DPC) to estimate how promising each prefix is for the eventual joint action, using centralized final-action values together with denoising-time bootstrap regularization. In Phase 2, we separately sample prefixes for policy optimization, convert DPC scores into process-level coordination effects through coalition value decomposition, estimate the reliability of different denoising steps through masked perturbations, and aggregate these weighted signals to update the decentralized diffusion policy. In this way, DiMAC extends centralized supervision from the action endpoint to the denoising process itself while preserving decentralized execution.

4.1 Phase 1: Centralized Learning of Denoising-Process Values

We study diffusion-based offline MARL under CTDE, where joint information is available during training while each agent executes reverse denoising from local observations at test time. The offline dataset provides tuples $(\mathbf{o}, \boldsymbol{\tau}_{history}, \mathbf{a}_0^{\mathcal{D}})$, and the decentralized diffusion policy is trained with the standard denoising objective:

$$\mathcal{L}_{diff}(\theta) = \mathbb{E}_{(\mathbf{o}, \boldsymbol{\tau}_{history}, \mathbf{a}_0^{\mathcal{D}}) \sim \mathcal{D},\, \epsilon \sim \mathcal{N}(0,\mathbf{I}),\, k} \left[\left\| \epsilon - \epsilon_\theta(\mathbf{a}_k, \mathbf{o}, \boldsymbol{\tau}_{history}, k) \right\|^2 \right]. \tag{3}$$

We reserve $\mathbf{a}_0^{\mathcal{D}}$ for offline joint actions and $\mathbf{a}_0^{\pi}$ for terminal joint actions generated by sampled reverse denoising chains.

At execution time, the reverse kernel factorizes across agents:

$$p_\theta(\mathbf{a}_{k-1} \mid \mathbf{a}_k, \mathbf{o}, \boldsymbol{\tau}_{history}) = \prod_{i=1}^{N} p_{\theta_i}(a_{k-1}^i \mid a_k^i, o_i, \boldsymbol{\tau}_{history}^i, k). \tag{4}$$

Equation (4) is the CTDE constraint respected by both phases in Figure 2.

Phase 1 in Figure 2 samples a denoising chain from the current decentralized policy and selects a prefix ending at denoising step k to train the Denoising Process Critic. Let $\mathbf{c} := (\boldsymbol{\tau}_{history}, \mathbf{o})$ denote the centralized training context, consisting of the replay history and joint observation. For a sampled denoising chain $\tau = (\mathbf{a}_K, \mathbf{a}_{K-1}, \ldots, \mathbf{a}_0^\pi)$, let $\tau_{K:k} = (\mathbf{a}_K, \mathbf{a}_{K-1}, \ldots, \mathbf{a}_k)$ denote the denoising prefix observed up to step k. We define the denoising-process value as follows:

$$V_k^\pi(\tau_{K:k}, \mathbf{c}) := \mathbb{E}[Q_{tot}(\mathbf{o}, \boldsymbol{\tau}_{history}, \mathbf{a}_0^\pi) \mid \tau_{K:k}, \mathbf{c}]\,. \tag{5}$$

This quantity asks how promising the current denoising prefix is for the eventual coordinated joint action if denoising continues under the current policy. DiMAC approximates it with a **Denoising Process Critic (DPC)**, denoted by C_ϕ:

$$C_\phi(\tau_{K:k}, \mathbf{c}) \approx V_k^\pi(\tau_{K:k}, \mathbf{c}). \tag{6}$$

DPC evaluates whether the current denoising prefix is already moving toward a high-quality coordinated action. In this way, terminal team-level feedback is transformed into supervision for intermediate denoising states.

The denoising-process value also satisfies a natural recursion along denoising time. Let $\mathcal{F}_k = \sigma(\mathbf{c}, \tau_{K:k})$ denote the information contained in the observed denoising prefix. Since $\mathcal{F}_k \subseteq \mathcal{F}_{k-1}$, the tower property gives:

$$V_k^\pi(\tau_{K:k}, \mathbf{c}) = \mathbb{E}\big[V_{k-1}^\pi(\tau_{K:k-1}, \mathbf{c}) \mid \mathcal{F}_k\big]\,. \tag{7}$$

This relation provides the temporal structure used to train DPC on partial denoising trajectories.

During Phase 1, the terminal joint action $\mathbf{a}_0^\pi$ of the sampled chain is evaluated by the centralized critic Q_{tot}, and the resulting target is propagated back to the sampled prefix through DPC. We first anchor the full denoising chain to the centralized value target $\hat{Q}_{tot}(\mathbf{o}, \boldsymbol{\tau}_{history}, \mathbf{a}_0^\pi)$:

$$\mathcal{L}_{term}(\phi) = \mathbb{E}_{(\mathbf{o}, \boldsymbol{\tau}_{history}) \sim \mathcal{D},\, \tau \sim p_\theta(\cdot \mid \mathbf{o}, \boldsymbol{\tau}_{history})} \left[\Big(C_\phi(\tau_{K:0}, \mathbf{c}) - \hat{Q}_{tot}(\mathbf{o}, \boldsymbol{\tau}_{history}, \mathbf{a}_0^\pi) \Big)^2 \right]. \tag{8}$$

Here $\mathbf{a}_0^\pi$ is the terminal joint action of the sampled denoising chain τ.

Terminal supervision alone is not sufficient, because DPC should also evolve consistently across neighboring denoising prefixes. We therefore add a denoising-time bootstrap regularization term:

$$\mathcal{L}_{TB}(\phi) = \mathbb{E}_{\substack{(\mathbf{o}, \boldsymbol{\tau}_{history}) \sim \mathcal{D}, \\ \tau \sim p_\theta(\cdot \mid \mathbf{o}, \boldsymbol{\tau}_{history}),\, k \sim \mathrm{Unif}(\{1, \ldots, K\})}} \left[\Big(C_\phi(\tau_{K:k}, \mathbf{c}) - \mathrm{sg}\big[C_{\bar{\phi}}(\tau_{K:k-1}, \mathbf{c}) \big] \Big)^2 \right], \tag{9}$$

where $\bar{\phi}$ denotes target-network parameters and $\mathrm{sg}[\cdot]$ is the stop-gradient operator. The overall DPC objective is then:

$$\mathcal{L}_{DPC}(\phi) = \mathcal{L}_{term}(\phi) + \beta \mathcal{L}_{TB}(\phi). \tag{10}$$

Together, the terminal anchor ties DPC to final-action quality, while the bootstrap term regularizes how denoising-process values evolve along the chain. Phase 1 therefore outputs a centralized evaluator that can score sampled denoising prefixes before denoising terminates.

4.2 Phase 2: DPC-Guided Policy Optimization

Phase 2 in Figure 2 samples a denoising prefix for policy optimization and uses the trained DPC to construct process-level supervision for the decentralized policy. This phase identifies which agents and interactions make a sampled prefix valuable, and also determines which denoising steps provide stable guidance for policy optimization.

Coalition value decomposition. We first convert the centralized DPC score into decentralized coordination effects. For agent i, let $\tau_{K:k}^{i,\mathrm{act}}$ denote the local denoising prefix produced by the current policy, and let $\tau_{K:k}^{i,\mathrm{ref}}$ denote a randomly sampled local reference prefix. For any coalition $G \subseteq \{1, \ldots, N\}$, we define the mixed denoising prefix as follows:

$$\tau_{K:k}^{[G]} := \big(\bar{\tau}_{K:k}^1(G), \ldots, \bar{\tau}_{K:k}^N(G) \big), \qquad \bar{\tau}_{K:k}^i(G) = \begin{cases} \tau_{K:k}^{i,\mathrm{ref}}, & i \in G, \\ \tau_{K:k}^{i,\mathrm{act}}, & i \notin G. \end{cases} \tag{11}$$

Its associated process value is defined by:

$$\begin{aligned} v_k(G) &= C_\phi(\tau_{K:k}^{[\emptyset]}, \mathbf{c}) - C_\phi(\tau_{K:k}^{[G]}, \mathbf{c}) \\ &\approx \mathbb{E}\Big[Q_{tot}(\mathbf{o}, \boldsymbol{\tau}_{history}, \mathbf{a}^{full}) - Q_{tot}(\mathbf{o}, \boldsymbol{\tau}_{history}, \mathbf{a}^{[G]}) \Big| \tau_{K:k}^{[\emptyset]}, \tau_{K:k}^{[G]}, \mathbf{c}\Big]. \end{aligned} \tag{12}$$

Here $\mathbf{a}^{full}$ denotes the terminal joint action obtained by continuing from the full current prefix $\tau_{K:k}^{[\emptyset]}$, and $\mathbf{a}^{[G]}$ denotes the terminal joint action obtained from the mixed prefix $\tau_{K:k}^{[G]}$. The quantity $v_k(G)$ therefore measures the drop in expected final-action quality caused by replacing the agents in G with reference prefixes while keeping the remaining agents on the current denoising prefix. By construction, $v_k(\emptyset) = 0$.

Modeling all higher-order interactions is exponential in the number of agents and is unnecessary in practice. We therefore use a tractable second-order approximation around the no-replacement coalition $\emptyset$:

$$\hat{v}_k(G) = v_k(\emptyset) + \sum_{i \in G} \Delta_k^i + \sum_{\substack{i<j \\ i,j \in G}} \Delta_k^{i,j}, \tag{13}$$

The singleton and pairwise effects are defined as follows:

$$\Delta_k^i = v_k(\{i\}) - v_k(\emptyset),\ \Delta_k^{i,j} = v_k(\{i,j\}) - v_k(\{i\}) - v_k(\{j\}) + v_k(\emptyset). \tag{14}$$

The singleton term measures the process-value drop caused by replacing agent i with its reference prefix. The pairwise term measures the additional drop caused by jointly replacing agents i and j beyond the sum of their individual effects. In this way, coalition value decomposition refines a centralized denoising-process value into agent-wise and pairwise coordination effects.

At denoising step k, we define the resulting process term as follows:

$$\mathcal{J}_k = \sum_{i=1}^{N} \Delta_k^i + \lambda_{pair} \sum_{(i,j) \in \mathcal{P}_k} \Delta_k^{i,j}, \tag{15}$$

where $\mathcal{P}_k$ is the set of agent pairs considered at step k.

Reliability weighting. Phase 2 also needs to determine how strongly each denoising step should influence policy optimization. Early denoising states are typically noisier and induce longer stochastic continuations, whereas middle and late denoising states are often more stable indicators of final-action quality. DiMAC therefore applies reliability weighting to modulate the process term across denoising time.

For a given denoising prefix $\tau_{K:k}$, we sample B random masks $m^{(1)}, \ldots, m^{(B)} \sim p_m$ and evaluate DPC on the corresponding masked views:

$$s_k^{(b)} = C_\phi(\tau_{K:k} \odot m^{(b)}, \mathbf{c}). \tag{16}$$

We compute the sample mean as follows:

$$\mu_k = \frac{1}{B} \sum_{b=1}^{B} s_k^{(b)}, \tag{17}$$

We also compute the empirical variance as follows:

$$\hat{\sigma}_k^2 = \frac{1}{B-1} \sum_{b=1}^{B} (s_k^{(b)} - \mu_k)^2. \tag{18}$$

A small variance indicates that the estimated denoising-process value is stable under masked perturbations of the same denoising prefix, whereas a large variance indicates lower reliability. We define an unnormalized reliability weight:

$$w_k = (\hat{\sigma}_k^2 + \varepsilon)^{-1}, \tag{19}$$

We then normalize it over the selected denoising steps:

$$\tilde{w}_k = \frac{w_k}{\frac{1}{|\mathcal{I}|} \sum_{k' \in \mathcal{I}} w_{k'}}. \tag{20}$$

The normalized weight has unit average over the supervised denoising steps, so more reliable steps receive larger emphasis without introducing uncontrolled scale drift across batches.

4.3 Overall Training Procedure

The outputs of Phase 2 are assembled into the final policy objective in Figure 2. We use a behavior regularizer to keep terminal actions close to the offline support:

$$\mathcal{L}_{BC}(\theta) = \mathbb{E}_{(\mathbf{o},\boldsymbol{\tau}_{history},\mathbf{a}_0^{\mathcal{D}})\sim\mathcal{D}}\left[\ell_{BC}\big(\hat{\mathbf{a}}_\theta^0(\mathbf{o},\boldsymbol{\tau}_{history}),\mathbf{a}_0^{\mathcal{D}}\big)\right], \tag{21}$$

where ℓ_{BC} is mean-squared error for continuous actions and cross-entropy for discrete actions.

Using the process term $\mathcal{J}_k$ in Eq. (15) and the normalized reliability weight $\tilde{w}_k$ in Eq. (20), we define the process objective as follows:

$$\mathcal{J}_{proc}(\theta) = \mathbb{E}\left[\frac{1}{|\mathcal{I}|}\sum_{k\in\mathcal{I}}\tilde{w}_k\,\mathcal{J}_k\right]. \tag{22}$$

The final policy objective is then:

$$\mathcal{L}_{policy}(\theta) = \mathcal{L}_{diff}(\theta) + \eta\mathcal{L}_{BC}(\theta) - \lambda\mathcal{J}_{proc}(\theta), \tag{23}$$

where the expectation in $\mathcal{J}_{proc}$ is over sampled denoising chains, random reference trajectories, and masks. Because each DPC score depends on the differentiable reverse denoising chain, gradients from $\mathcal{J}_{proc}$ propagate to intermediate denoising states according to how they affect the expected quality of the final joint action.

Each training iteration follows the two-phase pipeline shown in Figure 2. Phase 1 samples prefix trajectories for DPC training, evaluates the terminal joint actions with the centralized critic, and updates DPC under Eq. (10). Phase 2 separately samples prefix trajectories for policy optimization, constructs coalition-based process effects together with reliability weights, and updates the decentralized diffusion policy under Eq. (23). Algorithm 1 summarizes this procedure, and Appendix B provides the stochastic implementation details.

5 Experiments

5.1 Experimental Setup and Baselines

We evaluate DiMAC on two standard cooperative MARL benchmarks: Multi-Agent Particle Environment (MPE) (Lowe et al., 2017) and StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019). On MPE, we report results on Tag, Spread, and World. On SMAC, we report results on 3m, 8m, 5m_vs_6m, and 2s3z. For SMAC, legal-action masking is applied during discrete-action generation. DiMAC is trained with three independent random seeds. Detailed benchmark settings, training and evaluation protocol, and implementation details are deferred to Appendix C, Appendix D, and Appendix B.

We compare against representative baselines from behavior cloning, offline MARL, sequence modeling, and diffusion-based MARL, including BC, MA-CQL (Shao et al., 2023), MA-TD3+BC (Fujimoto & Gu, 2021), MAICQ (Yang et al., 2021), MADT (Meng et al., 2023), OMAR (Pan et al., 2022), MADiff (Zhu et al., 2024), and DoF (Li et al., 2025), under a unified offline evaluation protocol. Detailed baseline descriptions are deferred to Appendix E.

5.2 Main Results

Table 1 summarizes the quantitative comparison of DiMAC against strong offline MARL and diffusion-based baselines on MPE and SMAC. DiMAC performs strongly on both benchmarks and is consistently competitive with the best existing methods across continuous and discrete cooperative tasks, with the clearest gains appearing on tasks where success depends on maintaining coordinated behavior throughout generation rather than only producing a good final action.

On MPE, the gains are clearest on Tag and Spread, especially on the higher-quality splits. These tasks are particularly sensitive to whether coordinated spatial behavior can be preserved as actions are gradually refined, and DiMAC shows a clear advantage over both conventional offline MARL baselines and previous diffusion-based policies in this regime.

On SMAC, DiMAC achieves its strongest absolute returns on 3m and 2s3z, but it does not dominate every split. It leads on 3m-Good and 3m-Medium, 5m_vs_6m-Medium, and 2s3z-Good and

Table 1: Main results on MPE and SMAC. Entries report mean ± standard deviation over three independent seeds. Bold and underlined entries indicate the best and second-best mean, respectively, within each column.

MPE

Method	Tag				Spread				World			
	Expert	Md-Rep	Medium	Random	Expert	Md-Rep	Medium	Random	Expert	Md-Rep	Medium	Random
BC	40.0 ± 9.6	0.9 ± 1.4	22.5 ± 1.8	1.2 ± 0.8	35.0 ± 2.6	7.0 ± 1.7	31.6 ± 4.8	-0.5 ± 3.2	33.0 ± 9.9	2.3 ± 1.5	25.3 ± 2.0	-2.4 ± 0.5
MAICQ	95.2 ± 10.1	29.5 ± 21.8	58.3 ± 18.0	2.2 ± 2.6	101.4 ± 3.4	13.7 ± 5.6	29.3 ± 5.5	5.3 ± 3.4	98.5 ± 21.8	12.0 ± 9.1	69.9 ± 20.1	1.0 ± 2.2
MA-TD3+BC	113.1 ± 11.6	25.7 ± 20.1	63.3 ± 25.6	3.7 ± 3.5	110.3 ± 3.3	14.4 ± 5.8	32.3 ± 3.8	8.8 ± 4.4	95.3 ± 18.3	15.4 ± 8.1	72.4 ± 9.3	2.8 ± 3.5
MA-CQL	84.3 ± 10.2	23.9 ± 16.2	62.3 ± 27.8	2.7 ± 4.4	85.3 ± 4.6	19.2 ± 6.4	35.3 ± 10.3	20.5 ± 5.8	65.4 ± 20.2	21.3 ± 10.3	56.4 ± 6.4	2.4 ± 3.2
OMAR	115.8 ± 13.6	52.4 ± 18.3	55.3 ± 16.7	10.9 ± 3.8	113.9 ± 2.6	35.3 ± 14.0	45.0 ± 18.8	30.4 ± 8.2	113.4 ± 23.1	42.6 ± 28.2	69.2 ± 21.5	9.2 ± 3.6
MADiff	120.8 ± 11.3	57.4 ± 13.4	78.6 ± 12.3	13.3 ± 3.4	120.1 ± 6.3	48.4 ± 3.4	67.5 ± 8.5	20.6 ± 7.6	124.7 ± 20.1	51.6 ± 12.1	80.1 ± 13.4	6.1 ± 2.2
DoF	125.6 ± 8.6	65.4 ± 12.5	86.3 ± 10.6	16.5 ± 6.3	136.4 ± 3.9	57.4 ± 6.8	75.6 ± 8.7	35.9 ± 6.8	135.2 ± 19.1	58.6 ± 10.4	85.2 ± 11.2	**13.1 ± 2.1**
Ours	**216.3 ± 5.5**	**119.1 ± 9.3**	**134.6 ± 8.7**	**29.1 ± 8.3**	**203.8 ± 9.7**	**83.5 ± 12.5**	**108.3 ± 9.9**	**36.1 ± 6.7**	**137.5 ± 9.6**	**64.0 ± 15.2**	**89.8 ± 7.6**	2.3 ± 9.9
Gain vs. Best Prior	+90.7	+53.7	+48.3	+12.6	+67.4	+26.1	+32.7	+0.2	+2.3	+5.4	+4.6	-10.8

SMAC

Method	3m			8m			5m_vs_6m			2s3z		
	Good	Medium	Poor	Good	Medium	Poor	Good	Medium	Poor	Good	Medium	Poor
BC	3.7 ± 1.1	4.0 ± 1.0	3.4 ± 1.0	4.8 ± 0.6	5.6 ± 0.6	3.6 ± 0.8	2.4 ± 0.4	3.8 ± 0.5	3.3 ± 0.5	7.7 ± 0.9	7.6 ± 0.7	6.6 ± 0.2
MA-CQL	19.1 ± 0.1	13.7 ± 0.3	4.2 ± 0.1	5.4 ± 0.9	4.5 ± 1.5	3.5 ± 1.0	7.4 ± 0.6	8.1 ± 0.2	6.8 ± 0.1	17.4 ± 0.3	15.6 ± 0.4	8.4 ± 0.8
MAICQ	18.7 ± 0.7	13.9 ± 0.8	8.4 ± 2.6	18.3 ± 0.2	13.8 ± 3.5	3.2 ± 1.3	11.0 ± 0.6	10.6 ± 0.6	6.6 ± 0.2	18.3 ± 0.2	17.0 ± 0.1	9.9 ± 0.6
MADT	19.0 ± 0.3	15.8 ± 0.5	4.2 ± 0.1	18.5 ± 0.4	18.2 ± 0.1	4.8 ± 0.1	16.8 ± 0.1	16.1 ± 0.2	7.6 ± 0.3	18.1 ± 0.1	15.1 ± 0.2	8.9 ± 0.3
MADiff	19.3 ± 0.5	16.4 ± 2.6	10.3 ± 6.1	18.9 ± 1.1	16.8 ± 1.6	9.8 ± 0.9	16.5 ± 2.8	15.2 ± 2.6	8.9 ± 1.3	15.9 ± 1.2	15.6 ± 0.3	8.5 ± 1.3
DoF	19.8 ± 0.2	18.6 ± 1.2	**10.9 ± 1.1**	**19.6 ± 0.3**	**18.6 ± 0.8**	**12.0 ± 1.2**	**17.7 ± 1.1**	16.2 ± 0.9	**10.8 ± 0.3**	18.5 ± 0.8	**18.1 ± 0.9**	10.0 ± 1.1
Ours	**20.0 ± 0.0**	**18.7 ± 0.5**	9.8 ± 0.7	19.2 ± 0.4	17.9 ± 0.7	5.8 ± 2.3	16.8 ± 1.1	**16.3 ± 0.7**	7.4 ± 1.2	**19.3 ± 0.3**	17.6 ± 0.2	**10.3 ± 2.4**
Gain vs. Best Prior	+0.2	+0.1	-1.1	-0.4	-0.7	-6.2	-0.9	+0.1	-3.4	+0.8	-0.7	+0.3

2s3z-Poor, while DoF remains stronger on 3m-Poor, all 8m splits, 5m_vs_6m-Good and Poor, and 2s3z-Medium. The 2s3z results remain informative because this heterogeneous map requires complementary positioning, coordinated pressure, and synchronized target switching across unit roles. These results show that process supervision improves coordination broadly, while robustness on some low-quality or asymmetric settings remains a limitation.

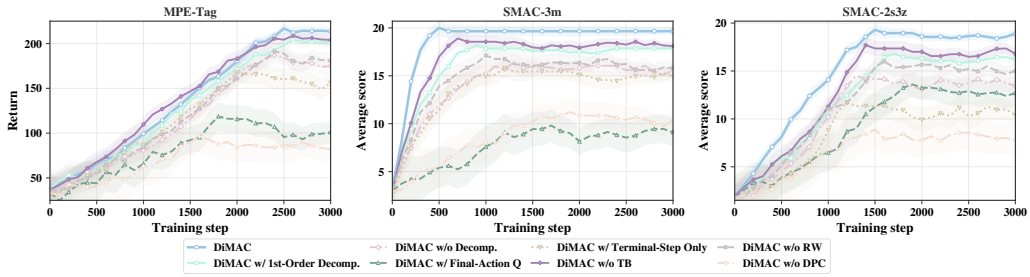


Figure 3: Training curves for representative ablations on MPE-Tag, SMAC-3m, and SMAC-2s3z. Removing DPC or replacing DPC-based process supervision with direct final-Q causes the largest degradation. Weakening the coalition value decomposition also hurts performance, while removing temporal bootstrap regularization ('w/o TD' in the plot) or reliable weighting ('w/o RW') mainly affects convergence speed and stability.

5.3 Ablation Studies

We conduct ablation experiments to validate the roles of the Denoising Process Critic (DPC) and its temporal bootstrap regularization, coalition value decomposition, and reliability weighting. Figure 3 compares DiMAC with representative variants on MPE-Tag, SMAC-3m, and SMAC-2s3z, spanning continuous control, homogeneous discrete coordination, and heterogeneous discrete coordination. In the figure, '1st-order only' keeps only the singleton terms in Eq. (14), 'No decomp.' removes coalition value decomposition and directly uses a shared process objective, 'Direct final-Q' replaces DPC-based process supervision with final-action Q guidance, 'Terminal-only' keeps only the terminal supervision in Eq. (8), and 'w/o TB', 'w/o RW', and 'w/o DPC' remove the bootstrap term in Eq. (9), the weighting in Eq. (20), and DPC altogether, respectively.

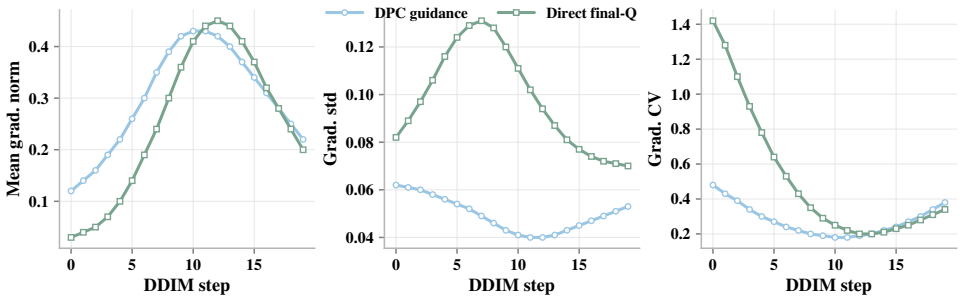


Figure 4: How the gradient signal is distributed across denoising steps. Compared with direct final-Q guidance, DPC guidance is spread more smoothly along the denoising chain and exhibits lower variability and lower relative noise. Detailed metric definitions are deferred to Appendix H.

Three conclusions are clear. First, weakening DPC-based process supervision is the most damaging change: 'w/o DPC' and 'Direct final-Q' suffer the largest drops on all three tasks, while 'Terminal-only' is also consistently weaker than the full model. This pattern indicates that DiMAC benefits from supervising denoising prefixes themselves rather than relying on terminal-action evaluation alone. Second, DiMAC consistently outperforms both '1st-order only' and 'No decomp.', showing that coalition value decomposition provides more useful decentralized updates than either a shared global signal or singleton terms alone. Third, 'w/o TB' and 'w/o RW' mainly exhibit slower improvement and larger oscillation, with smaller final degradation than the variants that discard process supervision. This is consistent with TB and RW serving primarily to stabilize process-level training.

5.4 Mechanism Analysis

We analyze DiMAC from three complementary perspectives and keep only the main findings here. Detailed metric definitions, exact protocols, the representative 2s3z rollout, the additional MPE-Tag gradient profile, and the full reliable-weight analysis are deferred to Appendix H.

Figure 4 shows that this behavioral advantage is accompanied by a different optimization signal on the denoising chain. On SMAC-3m, DPC guidance yields a smoother step-wise gradient profile together with lower cross-batch fluctuation and lower relative noise than direct final-Q guidance. This result suggests that DiMAC improves not only the endpoint being optimized, but also how supervision is distributed through denoising. Appendix H provides the exact metric definitions, the 50-batch evaluation protocol, and an additional MPE-Tag gradient profile.

Appendix H further examines the reliable-weight mechanism. On both MPE-Tag and SMAC-2s3z, the masked-prefix DPC variance is concentrated in the first few denoising steps and drops sharply afterward, while the learned weights follow the opposite trend. This supports the role of reliability weighting as a stabilization mechanism: it suppresses fragile early-process supervision and emphasizes the denoising stages where process evaluation is already more trustworthy.

6 Conclusion and Limitation

In this work, we introduced DiMAC, a multi-agent reinforcement learning framework for diffusion-based agents that addresses coordination and convergence challenges through denoising-process supervision. The central idea is the Denoising Process Critic, a process evaluator for the complete denoising trajectory, trained with the centralized Q-value of the final joint action. DPC is stabilized by temporal bootstrap regularization, and its estimated process value is translated into policy optimization through single-agent and pairwise coalition effects. This design allows DiMAC to shape the denoising process during centralized training while remaining fully compatible with CTDE at test time. Our empirical results indicate that this combination yields strong and robust cooperative behavior across diverse task environments. Nevertheless, DiMAC has not yet been extended to real-world multi-agent scenarios, and validating its effectiveness in practical systems with richer dynamics, noisy observations, and deployment constraints remains an important direction for future work.

AI USE STATEMENT

Generative AI assistants were used for literature search, language polishing, and formatting checks. The authors take full responsibility for the research content, experimental results, citations, and final text of the paper. The proposed method itself uses diffusion models for trajectory generation and process-reward evaluation, as described in the main text.

ETHICS STATEMENT

This work uses only the public MPE and SMAC multi-agent reinforcement learning benchmark environments. It does not involve human subjects, personal data, or real-world intervention.

REPRODUCIBILITY STATEMENT

The appendices provide the training and inference algorithms, network architecture, hyperparameters, training settings, and evaluation protocols. The source code will be released upon acceptance.

REFERENCES

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, T. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *ArXiv*, abs/2211.15657, 2022. URL https://api.semanticscholar.org/CorpusID:254044710.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Claude Formanek, Asad Jeewa, Jonathan Shock, and Arnu Pretorius. Off-the-grid marl: Datasets with baselines for offline multi-agent reinforcement learning. 2023. URL https://api.semanticscholar.org/CorpusID:262460140.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Chen-Xiao Gao, Chenyang Wu, Mingjun Cao, Chenjun Xiao, Yang Yu, and Zongzhang Zhang. Behavior-regularized diffusion policy optimization for offline reinforcement learning. *arXiv preprint arXiv:2502.04778*, 2025.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Siyi Hu, Fengda Zhu, Xiaojun Chang, and Xiaodan Liang. Updet: Universal multi-agent reinforcement learning via policy decoupling with transformers. *arXiv preprint arXiv:2101.08001*, 2021.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

Chiyu Jiang, Andre Cornman, Cheolho Park, Benjamin Sapp, Yin Zhou, Dragomir Anguelov, et al. Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9644–9653, 2023.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the association for computational linguistics*, 8:64–77, 2020.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. *arXiv preprint arXiv:2109.11251*, 2021.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1179–1191. Curran Associates, Inc., 2020a. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/0d2b2061826a5df3221116a5085a6052-Paper.pdf.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020b.

Dongsu Lee, Daehee Lee, and Amy Zhang. Multi-agent coordination via flow matching, 2026. URL https://arxiv.org/abs/2511.05005.

Chao Li, Ziwei Deng, Chenxing Lin, Wenqi Chen, Yongquan Fu, Weiquan Liu, Chenglu Wen, Cheng Wang, and Siqi Shen. Dof: A diffusion factorization framework for offline multi-agent reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Zhuoran Li, Xun Wang, Hai Zhong, Qingxin Xia, Lihua Zhang, and Longbo Huang. Om2p: Offline multi-agent mean-flow policy, 2026. URL https://arxiv.org/abs/2508.06269.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

Haofei Lu, Dongqi Han, Yifei Shen, and Dongsheng Li. What makes a good diffusion planner for decision making? *arXiv preprint arXiv:2503.00535*, 2025.

Linghui Meng, Muning Wen, Chenyang Le, Xiyun Li, Dengpeng Xing, Weinan Zhang, Ying Wen, Haifeng Zhang, Jun Wang, Yaodong Yang, et al. Offline pre-trained multi-agent decision transformer. *Machine Intelligence Research*, 20(2):233–248, 2023.

Mang Ning, Mingxiao Li, Jianlin Su, Albert Ali Salah, and Itir Onal Ertugrul. Elucidating the exposure bias in diffusion models. *arXiv preprint arXiv:2308.15321*, 2023.

Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

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.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

Allen Z. Ren, Justin Lidard, Lars L. Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization, 2024. URL https://arxiv.org/abs/2409.00588.

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.

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

Yorai Shaoul, Itamar Mishani, Shivam Vats, Jiaoyang Li, and Maxim Likhachev. Multi-robot motion planning with diffusion models. *arXiv preprint arXiv:2410.03072*, 2024.

Jascha Narain Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *ArXiv*, abs/1503.03585, 2015. URL https://api.semanticscholar.org/CorpusID:14888175.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Jianhong Wang, Yuan Zhang, Tae-Kyun Kim, and Yunjie Gu. Shapley q-value: A local reward approach to solve global reward games. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7285–7292, April 2020a. ISSN 2159-5399. doi: 10.1609/aaai.v34i05.6220. URL http://dx.doi.org/10.1609/aaai.v34i05.6220.

Jianhong Wang, Yuan Zhang, Yunjie Gu, and Tae-Kyun Kim. Shaq: Incorporating shapley value theory into multi-agent q-learning, 2023. URL https://arxiv.org/abs/2105.15013.

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

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

Yiqin Yang, Xiaoteng Ma, Chenghao Li, Zewu Zheng, Qiyuan Zhang, Gao Huang, Jun Yang, and Qianchuan Zhao. Believe what you see: Implicit constraint approach for offline multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:10299–10312, 2021.

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.

Yifan Zhong, Jakub Grudzien Kuba, Xidong Feng, Siyi Hu, Jiaming Ji, and Yaodong Yang. Heterogeneous-agent reinforcement learning. *Journal of Machine Learning Research*, 25(32): 1–67, 2024.

Zhengbang Zhu, Minghuan Liu, Liyuan Mao, Bingyi Kang, Minkai Xu, Yong Yu, Stefano Ermon, and Weinan Zhang. Madiff: Offline multi-agent learning with diffusion models. *Advances in Neural Information Processing Systems*, 37:4177–4206, 2024.

A LIST OF NOTATIONS

To facilitate a clear understanding of the mathematical framework of DiMAC, we summarize the main symbols used throughout the paper in Table 2.

Table 2: Notation Summary

Notation	Description
N	Number of agents
$s, \mathbf{a}$	Global state and joint action
o_i	Local observation of agent i
π_θ	Diffusion-based decentralized policy with parameters θ
$\mathcal{D}, \mathcal{B}$	Offline dataset and minibatch sampled from it
K, k	Total denoising steps and current denoising index
$\mathbf{a}_0^{\mathcal{D}}, \mathbf{a}_0^{\pi}$	Offline joint action and terminal joint action generated by the current diffusion policy
$\mathbf{a}_k, a_k^i$	Joint denoising state at step k and its local component for agent i
τ	Denoising trajectory sequence $\{\mathbf{a}_K, \mathbf{a}_{K-1}, \ldots, \mathbf{a}_0^{\pi}\}$
$\tau_{K:k}$	Denoising prefix from step K to step k
$\boldsymbol{\tau}_{history}$	Replay-history context used by the implementation-level centralized critic
$Q_{tot}, \hat{Q}_{tot}$	Centralized critic and its conservative target used to supervise DPC
C_ϕ	DPC score on a denoising prefix
$\bar{\omega}, \bar{\phi}$	Target parameters for the centralized critic and DPC
β_k	Diffusion variance schedule at denoising step k
$\mathcal{I}$	Set of denoising steps selected for process supervision
G	Coalition of agents whose denoising prefixes are replaced by references
$\mathcal{P}_k$	Set of agent pairs considered at denoising step k
$w_k, \tilde{w}_k$	Raw and normalized reliability weights at denoising step k
$\Delta_k^i, \Delta_k^{i,j}$	Single-agent and pairwise process contributions at denoising step k
$\mathcal{J}_{proc}$	Process objective for policy improvement
$\mathcal{L}_{diff}, \mathcal{L}_{BC}$	Diffusion loss and behavior-cloning regularizer
$\mathcal{L}_{term}, \mathcal{L}_{TB}, \mathcal{L}_{DPC}$	Terminal DPC loss, temporal bootstrap loss, and total DPC loss
$\mathcal{L}_Q, \mathcal{L}_{policy}$	Centralized critic loss and final policy objective
$\eta_Q, \eta_\phi, \eta_\theta$	Learning rates for the critic, DPC, and policy updates
$\beta, \lambda, \lambda_{pair}, \eta$	Weights for temporal bootstrap regularization, process objective, pairwise effects, and BC regularization
ρ	Polyak averaging coefficient for target-parameter updates

B Implementation Details of Centralized Value Estimation and Process Supervision

In the main text, we write the terminal supervision for DPC as $\hat{Q}_{tot}(\mathbf{o}, \boldsymbol{\tau}_{history}, \mathbf{a}_0^{\pi})$. Here $\mathbf{a}_0^{\pi}$ is the terminal joint action generated by the sampled diffusion chain, whereas $\mathbf{a}_0^{\mathcal{D}}$ is reserved for offline actions taken directly from the dataset. In implementation, this target is built from a centralized critic that receives the joint observation, the replay history, and the final joint action.

B.1 Input Representation

For the action-generation setting used in this paper, the centralized critic takes the form

$$Q_{tot}(\mathbf{o}, \boldsymbol{\tau}_{history}, \mathbf{a}),$$

where $\boldsymbol{\tau}_{history}$ is the replay history trajectory and $\mathbf{a}$ is the final joint action produced by the diffusion policy. The history input improves stability in partially observable settings by giving the critic access to recent coordination context beyond the current observation alone.

B.2 Training Objective

Since the quality of Q_{tot} directly affects DPC supervision, we train the critic with standard offline Temporal Difference (TD) targets:

$$\mathcal{L}_Q(\omega) = \mathbb{E}_{(\mathbf{o}, \boldsymbol{\tau}_{history}, \mathbf{a}^{\mathcal{D}}, r, \mathbf{o}') \sim \mathcal{D}} \left[\left(Q_{tot}(\mathbf{o}, \boldsymbol{\tau}_{history}, \mathbf{a}^{\mathcal{D}}) - \mathcal{Y} \right)^2 \right], \tag{24}$$

with bootstrap target

$$\mathcal{Y} = r + \gamma Q_{\bar{\omega}}(\mathbf{o}', \boldsymbol{\tau}'_{history}, \mathbf{a}'^{\pi}). \tag{25}$$

where $\mathbf{a}'^{\pi}$ is the next joint action sampled from the target diffusion policy under the next-step context. To reduce overestimation on out-of-distribution actions, the implementation uses an ensemble of five centralized critics together with a conservative lower-confidence estimate. The terminal target used to train DPC is then instantiated as the ensemble mean minus a standard-deviation penalty, evaluated at the sampled policy action $\mathbf{a}_0^{\pi}$. This is the quantity denoted by $\hat{Q}_{tot}(\mathbf{o}, \boldsymbol{\tau}_{history}, \mathbf{a}_0^{\pi})$ in the main text.

B.3 Stochastic Implementation of Process Supervision

For conceptual clarity, Eq. (22) in the main text abstracts the process objective as an expectation over weighted denoising-step contributions. In the actual implementation, we use a three-step stochastic estimator rather than explicitly evaluating every denoising step on every sampled chain.

For each sampled minibatch element, the current decentralized diffusion policy first generates a full denoising chain

$$\tau = (\mathbf{a}_K, \mathbf{a}_{K-1}, \ldots, \mathbf{a}_0^{\pi}).$$

We then sample three denoising steps

$$\{k^{(r)}\}_{r=1}^{3} \sim \mathrm{Unif}(\mathcal{I}),$$

and reuse the corresponding prefixes $\{\tau_{K:k^{(r)}}\}_{r=1}^{3}$ for all process-level computations in that update. The same three sampled steps are also reused in the temporal-bootstrap term in Eq. (9).

For each sampled prefix, we first perform reliability weighting. Specifically, we draw three random masks

$$m^{(1)}, m^{(2)}, m^{(3)} \sim p_m,$$

re-evaluate the same sampled prefix under these masked views, and obtain three masked DPC scores

$$s_k^{(b)} = C_\phi(\tau_{K:k} \odot m^{(b)}, \mathbf{c}), \qquad b = 1, 2, 3.$$

Here k denotes any one of the three sampled denoising steps. The resulting sample variance defines the raw reliability score, and the corresponding raw weights are normalized over all sampled prefixes participating in the current minibatch update.

The same three sampled prefixes are then used for coalition value decomposition. For each sampled denoising step, we first evaluate the full current prefix $\tau_{K:k}^{[\emptyset]}$, then evaluate the coalition-mixed prefixes $\tau_{K:k}^{[G]}$ required for singleton and pairwise terms, and finally form the corresponding coalition values $v_k(G)$ and process term $\mathcal{J}_k$.

The policy update finally uses the minibatch average of the weighted three-prefix estimator

$$\hat{\mathcal{J}}_{proc} = \frac{1}{|\mathcal{B}|} \sum_{n=1}^{|\mathcal{B}|} \frac{1}{3} \sum_{r=1}^{3} \tilde{w}_{k_n^{(r)}} \, \mathcal{J}_{k_n^{(r)}}.$$

In other words, the implementation follows the same conceptual objective as Eq. (22), but realizes it with three randomly selected denoising steps per sampled chain for efficiency.

C Benchmark Details

We summarize the task-level evaluation settings used in the main experiments.

MPE. We use the offline datasets of (Pan et al., 2022) and evaluate on three standard tasks. **Spread** requires agents to cover different landmarks without collision and mainly tests continuous-control coordination. **Tag** requires multiple predators to surround and capture a faster prey, making coordinated pursuit and interception important. **World** extends the Tag setting with obstacles and more complex team objectives.

SMAC. We use the Off-the-Grid offline dataset (Formanek et al., 2023) and evaluate on four representative maps. **3m** and **8m** are homogeneous Marine-vs-Marine settings. **5m_vs_6m** is a harder asymmetric map that requires precise kiting and focus-fire behavior. **2s3z** is a heterogeneous setting with two Stalkers and three Zealots, which tests whether the learned policy can maintain coordination across different unit roles.

D Training and Evaluation Protocol

Unless otherwise stated, DiMAC is trained for $20{,}000$ gradient steps with batch size 512. The policy network, the centralized Q ensemble, and DPC use separate Adam optimizers. Training is performed on the chosen offline split, and we additionally create an $80/20$ random train/validation split only for monitoring. Final reported performance is obtained by environment rollout evaluation rather than validation-set prediction error.

For DiMAC, each reported main-table result is averaged over three independent training seeds. We evaluate checkpoints with 30 rollout episodes using 10 parallel environments, and the score for each seed is taken from the best periodically evaluated EMA checkpoint under this rollout protocol. For SMAC, legal-action masking is applied during both training and evaluation, and testing uses deterministic argmax decoding over the masked discrete-action scores.

All experiments were run on a Linux server with 4 NVIDIA Tesla P100 GPUs (16 GB memory each), Intel Xeon E5-2690 v4 CPUs, and 251 GB system memory. Each training run used a single GPU. With an average cost of approximately 3.8 seconds per optimization step and $20{,}000$ training steps per run, a typical single-seed run requires about 7.6×10^4 seconds, or approximately 21.1 hours. Under the three-seed protocol, this corresponds to roughly 63.3 GPU-hours for one task configuration. Exact wall-clock time varies slightly across environments and evaluation frequency.

E Baseline Details

We compare against the following representative baselines in the main experiments.

BC. Behavioral cloning trains the policy only by supervised imitation on the offline dataset and serves as a simple non-RL baseline.

MA-CQL, MA-TD3+BC, and MAICQ. These methods are representative offline MARL baselines based on value regularization, actor-critic learning, or implicit constraint mechanisms.

Table 3: DPC score calibration against completed-trajectory returns.

Task	Spearman$(\mathrm{DPC}, \mathrm{return})$	Top–bottom gap
MPE-Tag	0.58	+78.4
SMAC-2s3z	0.63	+4.7

Table 4: Reliability statistics for perturbed denoising prefixes.

Prefix type	DPC score	Masked variance	Normalized weight
Policy-generated	12.8	0.18	1.27
Masked	10.6	0.41	0.89
Coalition-mixed	8.9	0.29	0.58

MADT. MADT is a sequence-modeling baseline that treats offline multi-agent decision making as autoregressive trajectory modeling.

OMAR. OMAR is an offline MARL baseline designed to improve coordination in continuous-action settings.

MADiff and DoF. These are diffusion-based MARL baselines. MADiff models multi-agent behavior with diffusion generation, whereas DoF emphasizes decentralized factorized diffusion.

F Licenses and Terms of Use for Existing Assets

We use the official Multi-Agent Particle Environment repository and the official SMAC repository as benchmark implementations; both are publicly released under the MIT License. For the SMAC offline benchmark resources, we rely on the official OG-MARL release, which is publicly available under the Apache-2.0 License.

SMAC additionally requires the StarCraft II game installation and is therefore used subject to Blizzard's end-user license agreement and terms of use. For the MPE offline datasets of Pan et al. (Pan et al., 2022), we cite the original research release; however, we did not identify an explicit public license statement on the source page used to access the data at the time of submission, so we use these data only for academic research evaluation.

G Additional Process-Level Validation

G.1 DPC Return Calibration

We evaluate whether DPC scores predict the quality of completed denoising trajectories. For each sampled prefix, we record its DPC score, complete the reverse process, and measure the resulting rollout return. The Spearman correlations are 0.58 on MPE-Tag and 0.63 on SMAC-2s3z. The mean return of prefixes in the top score quintile exceeds that of the bottom quintile by 78.4 and 4.7, respectively. These results show that DPC scores preserve useful information about the quality of the eventual joint action.

G.2 Reliability of Perturbed Prefixes

We compare policy-generated, masked, and coalition-mixed prefixes using their mean DPC score, masked variance, and normalized reliability weight. The results are reported in Table 4.

Coalition-mixed prefixes receive the lowest normalized weight, while masked prefixes have the largest variance. Reliability weighting therefore reduces the contribution of less stable prefix evaluations.

Table 5: Sensitivity to the number of random masks on MPE-Tag.

Number of masks B	Return
2	210.9 ± 8.1
3	216.3 ± 5.5
5	199.6 ± 5.2

G.3 Sensitivity to the Number of Random Masks

We vary the number of random masks used to estimate prefix reliability on MPE-Tag. All other settings remain unchanged.

The best performance is obtained with $B = 3$, which provides a stable reliability estimate without the additional DPC evaluations required by larger mask sets.

H Additional Mechanism Analyses

H.1 SMAC Coordination Statistics

The coordination statistics in Figure 5 are designed to test whether decentralized denoising leads multiple agents to align on the same local combat objective before the final discrete action is sampled. In SMAC, many successful trajectories in the offline dataset exhibit a clear focus-fire pattern, where at least two allied units attack the same enemy unit and create a temporary local numerical advantage. This behavior is especially important in maps such as 2s3z and 5m_vs_6m, where splitting attacks across several enemies often delays the first elimination and leads to unfavorable trades. For this reason, we treat shared attack commitment as a concrete behavioral signature of cooperation rather than as a generic agreement statistic.

Our discrete-action implementation uses a structured output rather than an undifferentiated flat action category. For an agent with M action-type choices and E enemy candidates, the diffusion policy outputs a vector in $\mathbb{R}^{M+E}$. The first M dimensions represent action-selection logits, while the remaining E dimensions represent attack-target logits. The target-related dimensions are used only when the selected action type is *attack*; for non-attack actions they are ignored. Legal-action masking is applied before decoding so that unavailable actions and invalid targets receive zero probability mass. This implementation is equivalent in expressive power to the standard flat SMAC action space, but it makes the target preference explicit and therefore allows a more direct analysis of coordination.

We do not compute the coordination metric from the final executed action alone. Instead, we inspect the last five denoising steps, where the reverse process has already entered a low-noise regime and the action preference becomes behaviorally meaningful. Let $\mathcal{K}_{late}$ denote these five late denoising steps. Both statistics are computed only on denoising steps for which at least one agent is in attack mode. For each agent i at denoising step $k \in \mathcal{K}_{late}$, let $p_i^k(a)$ be the masked distribution over action types and let $p_i^k(u \mid \text{attack})$ be the masked distribution over enemy targets u. We include agent i in the coordination analysis at step k only if its most likely action is attack, namely

$$i \in \mathcal{A}^k \iff \arg\max_a p_i^k(a) = \text{attack}. \tag{26}$$

For every attacking agent, we then define its most likely target as

$$\hat{u}_i^k = \arg\max_u p_i^k(u \mid \text{attack}). \tag{27}$$

The key event of interest is a 2-vs-1 focus-fire configuration. For each enemy unit u, we count how many allied agents select it as their most likely target:

$$n_u^k = \sum_{i \in \mathcal{A}^k} \mathbf{1}[\hat{u}_i^k = u]. \tag{28}$$

Whenever $n_u^k \geq 2$, we regard the corresponding attack assignments as a cooperative event at denoising step k, because at least two agents are committing to the same enemy. The *same-target ratio* reported

in Figure 5 is the fraction of attacking-agent decisions in the late denoising window that participate in such shared-target events:

$$R_{\text{same}} = \frac{\sum_{k\in\mathcal{K}_{late}} \sum_u n_u^k \cdot \mathbf{1}[n_u^k \geq 2]}{\sum_{k\in\mathcal{K}_{late}} |\mathcal{A}^k|}. \tag{29}$$

This statistic is aggregated over evaluation episodes and seeds. A higher value means that independently executed agents are more often converging toward the same enemy during late denoising, which is exactly the kind of local focus-fire behavior frequently observed in successful offline trajectories. Figure 5 summarizes target-selection behavior on SMAC. Across 3m, 8m, and 2s3z, DiMAC consistently attains the highest same-target ratio and the lowest target entropy, indicating that independently denoising agents converge more consistently to shared attack targets. The effect is especially clear on 2s3z, where effective combat requires sustained coupling between Zealots and Stalkers. The detailed metric definition and its connection to cooperative focus-fire behavior are given in Appendix H.1.

The *target entropy* complements this discrete focus-fire count with a soft distributional measure. For each late denoising step, we aggregate the attack-target distributions of the attacking agents:

$$m_u^k = \sum_{i\in\mathcal{A}^k} p_i^k(u \mid \text{attack}), \qquad \tilde{m}_u^k = \frac{m_u^k}{\sum_v m_v^k}. \tag{30}$$

We then compute

$$H_{\text{target}}^k = -\sum_u \tilde{m}_u^k \log(\tilde{m}_u^k + \epsilon), \tag{31}$$

and average it over $k \in \mathcal{K}_{late}$, evaluation episodes, and random seeds. Lower entropy means that the attack preference is concentrated on a smaller set of common targets, whereas higher entropy indicates that the agents' target preferences are spread across different enemies.

Using the last five denoising steps instead of only the terminal action is important for the claim made in this paper. DiMAC is intended to improve coordination throughout the denoising process, not merely to alter the final discrete decision after denoising has already finished. The late-step logits therefore reveal whether target agreement emerges stably before the final argmax or whether it appears only as an incidental endpoint effect. When DiMAC yields both a higher same-target ratio and a lower target entropy, the result supports the intended interpretation: centralized process supervision is shaping decentralized denoising toward more coherent cooperative target assignment.

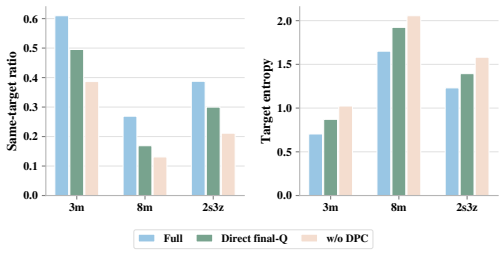


Figure 5: Target selection statistics, computed from the last five denoising steps. DiMAC yields higher same-target ratio and lower target entropy than Direct final-Q and w/o DPC, indicating more cooperative under decentralized execution.

H.2 Representative 2s3z Rollout

Figure 6 visualizes a representative 2s3z battle to complement the aggregate coordination statistics in the main text. The top row shows a DiMAC rollout that attains the maximal score of 20, whereas the bottom row shows a lower-scoring Direct final-Q episode.

The snapshots make the statistical difference more concrete. In the opening stage, both methods begin from a comparable contact configuration. The divergence appears once the engagement unfolds. Under DiMAC, the Zealots stay near the main clash and pin the enemy formation in place, while the Stalkers continue to reinforce the same threatened target from range. This coherent local advantage is preserved through the pressure and focus-fire stages, so the team removes one enemy and immediately rolls that advantage into the next target. Direct final-Q is much less stable after the first exchange. One Zealot begins to peel away from the main fight, the front line no longer anchors the same local target for the back line, and the battle decomposes into partially separated skirmishes. As a result, the eventual target switch is slower and less synchronized, which allows the opponent to trade units more favorably.

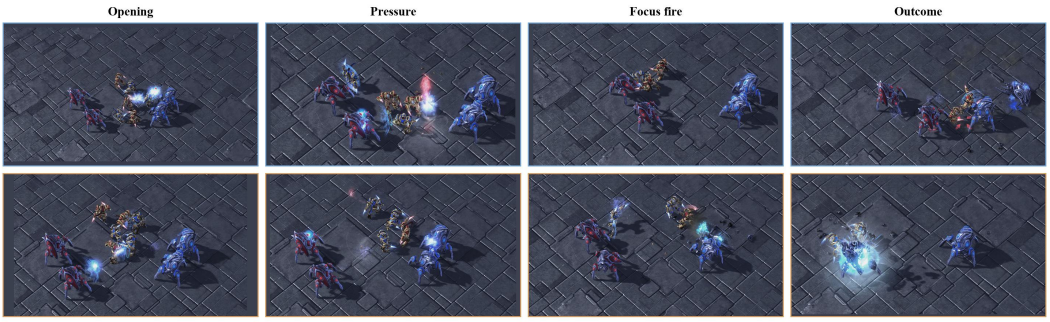


Figure 6: Representative 2s3z rollout snapshots. Top row: DiMAC maintains a compact front-to-back formation, keeps Zealots and Stalkers committed to the same local engagement, and converts the first pickoff into a second elimination. Bottom row: Direct final-Q starts from a similar opening but then drifts into fragmented exchanges, leaving one frontline unit unsupported and weakening the subsequent target switch.

H.3 Step-wise Gradient Profiles

Figure 7 provides an additional gradient-profile analysis on MPE-Tag to complement the SMAC-3m result in the main text. The protocol is the same in both environments. In both figures, the left panel reports the mean gradient norm at each denoising step, the middle panel reports the cross-batch gradient standard deviation, and the right panel reports the coefficient of variation, which we use as a relative-noise measure. For each compared optimization signal, namely the DPC-based process term and the direct final-Q term, we sample 50 training minibatches from the offline dataset and evaluate the resulting step-wise gradients separately. For minibatch b and denoising step k, let

$$g_k^{(b)} = \left\| \nabla_{\mathbf{a}_k} \mathcal{L}_{policy}^{(b)} \right\|_2 \tag{32}$$

denote the gradient magnitude back-propagated to the denoising state at step k. The first curve reports the batch-averaged profile

$$\bar{g}_k = \frac{1}{50} \sum_{b=1}^{50} g_k^{(b)}. \tag{33}$$

The variance curve is computed across minibatches,

$$\mathrm{Var}_k = \frac{1}{49} \sum_{b=1}^{50} \left(g_k^{(b)} - \bar{g}_k \right)^2, \tag{34}$$

and the relative-noise curve is defined as

$$\mathrm{RN}_k = \frac{\sqrt{\mathrm{Var}_k}}{\bar{g}_k + \varepsilon}. \tag{35}$$

The two compared optimization signals are measured separately under this same 50-batch protocol. As in SMAC, the DPC-based process term produces a smoother step-wise gradient profile with lower variance and lower relative noise, especially in the early and middle denoising stages. In other words, DPC not only reallocates where the optimization signal is strongest along denoising time, but also makes that signal more stable from batch to batch. This supports the claim that DiMAC improves not only what signal is optimized, but also how that signal is distributed across denoising time.

H.4 Reliability Weighting

Figure 8 analyzes the reliable-weight mechanism under the same stochastic implementation used in training. Each sampled denoising chain contributes three randomly selected denoising steps.

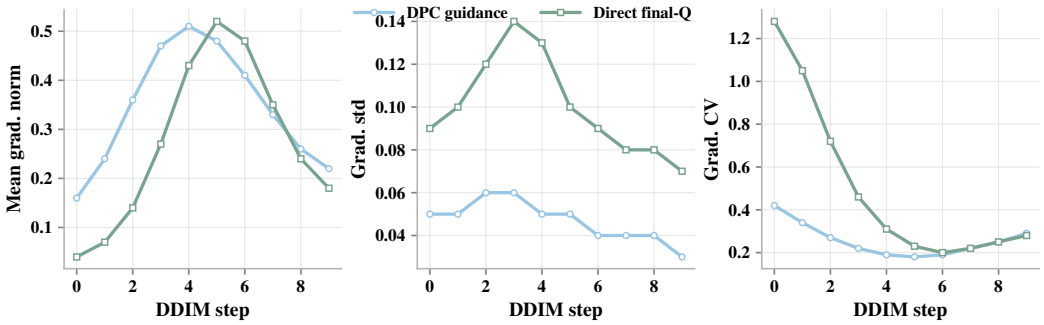


Figure 7: Additional step-wise gradient profile on MPE-Tag. The three panels report the batch-mean gradient norm (left), the cross-batch gradient standard deviation (middle), and the coefficient of variation (right). Compared with the direct final-Q term, the DPC-based process term is distributed more smoothly along the denoising chain and exhibits lower variance and lower relative noise.

For exposition, we describe the masked-variance computation for an arbitrary sampled step k with corresponding prefix $\tau_{K:k}$. We then draw three random masks,

$$m^{(1)}, m^{(2)}, m^{(3)} \sim p_m, \tag{36}$$

and re-evaluate the same prefix under these masked views. This gives three masked DPC scores,

$$s_k^{(b)} = C_\phi(\tau_{K:k} \odot m^{(b)}, \mathbf{c}), \qquad b = 1, 2, 3. \tag{37}$$

For each step k, we then compute the cross-mask sample mean and sample variance,

$$\mu_k = \frac{1}{3}\sum_{b=1}^{3} s_k^{(b)}, \qquad \hat{\sigma}_k^2 = \frac{1}{2}\sum_{b=1}^{3}\left(s_k^{(b)} - \mu_k\right)^2. \tag{38}$$

The figure reports the batch-averaged variance profile together with the corresponding normalized reliability weights obtained from Eq. (20). We compute the two curves separately on sampled prefixes from MPE-Tag and SMAC-2s3z. The resulting profiles show a pronounced two-phase pattern on both tasks. In the first three to four DDIM steps, the cross-mask DPC variance is substantially higher than in the rest of the trajectory, which means that small perturbations of an early high-noise prefix can still induce large changes in the predicted process value. After this short unstable regime, the variance drops sharply and then remains in a much lower range for the middle and late denoising steps. The normalized weights follow the opposite trend almost immediately: they stay small during the unstable early regime, increase rapidly once the variance collapses, and then saturate into a relatively stable high-weight region. This shape matters because it shows that RW is not merely applying a generic late-step preference. Instead, it reacts to a concrete reliability transition in the reverse process and suppresses precisely the part of denoising where DPC evaluation is most fragile.

H.5 Representative Hyperparameters

Tables 6 and 7 summarize the main benchmark-specific hyperparameters used in our experiments.

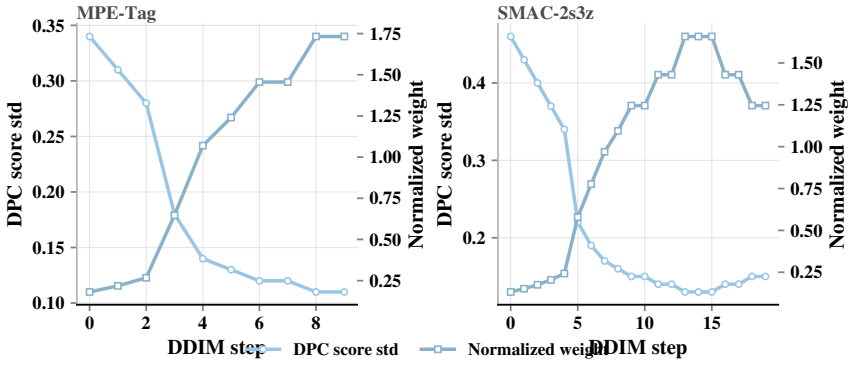


Figure 8: Reliability-weight analysis on MPE-Tag and SMAC-2s3z. Early denoising steps show larger DPC-score variance under masked-prefix perturbations and therefore receive smaller normalized weights, while more stable middle and late steps receive larger weights.

Table 6: Representative hyperparameters for SMAC experiments.

Hyperparameter	**Value**
Training steps	20,000
Batch size	256
History horizon	2
Forward diffusion steps	100
DDIM reverse steps	20
Q pretraining steps	2,000
Centralized Q learning rate	5×10^{-3}
Policy learning rate	1×10^{-5}
DPC learning rate	1×10^{-3}
Hidden dimension	64
Backbone hidden dimension	64
Time-embedding dimension	128
Backbone layers	2
History encoder	Transformer, 2 layers, 2 heads
Trajectory latent dimension	64
Process encoder	LSTM
Pattern latent dimension	64
Number of target Q networks	5
Evaluation episodes	30
Legal-action masking	Enabled
Structured SMAC action head	Enabled
Non-attack action types	6
Process supervision (DPC / RW / coalition value decomposition)	Enabled

Table 7: Representative hyperparameters for MPE experiments.

Hyperparameter	**Value**
Training steps	20,000
Batch size	512
History horizon	1
Forward diffusion steps	50
DDIM reverse steps	10
Q pretraining steps	3,000
Policy learning rate	1×10^{-5}
Centralized Q learning rate	1×10^{-3}
DPC learning rate	5×10^{-4}
Hidden dimension	64
Backbone hidden dimension	64
Time-embedding dimension	128
Backbone layers	2
History encoder	Transformer, 2 layers, 2 heads
Trajectory latent dimension	64
Process encoder	LSTM
Pattern latent dimension	64
Number of target Q networks	5
Evaluation episodes	30
Process supervision (DPC / RW / coalition value decomposition)	Enabled

I Network Architecture Implementation Details

Figure 9 illustrates the implementation-level architecture used by DiMAC, including the history encoder, the decentralized diffusion backbone, and the DPC scoring network.

I.1 Action Generation Architecture

The decentralized diffusion backbone uses a compact shared design across benchmarks. We use hidden dimension 64, backbone hidden dimension 64, time-embedding dimension 128, and two backbone layers throughout. The backbone is factorized across agents and does not use test-time inter-agent attention or centralized communication. For relatively simple continuous-control tasks such as MPE, the policy uses an MLP backbone that conditions on local observations, diffusion timesteps, and history encodings and predicts denoising noise directly in the action space. For SMAC, the same backbone is combined with the structured discrete-action parameterization described below. We use 50 forward diffusion steps with 10 DDIM reverse steps on MPE, and 100 forward diffusion steps with 20 DDIM reverse steps on SMAC.

I.2 History and Process Encoders

DiMAC uses two distinct encoders. The first is a history encoder that summarizes replay history for policy conditioning and centralized value estimation. The second is a process encoder that summarizes the denoising prefix of each agent before DPC scoring. In the implementation used for this paper, the history encoder is a transformer-style sequence encoder with two layers and two attention heads, and it produces a trajectory latent of dimension 64. The process encoder is an LSTM-style prefix encoder that produces a denoising-pattern latent of dimension 64 before the joint DPC scoring MLP. Their roles are different: the history encoder summarizes past interaction context, whereas the process encoder summarizes how the current denoising chain is unfolding.

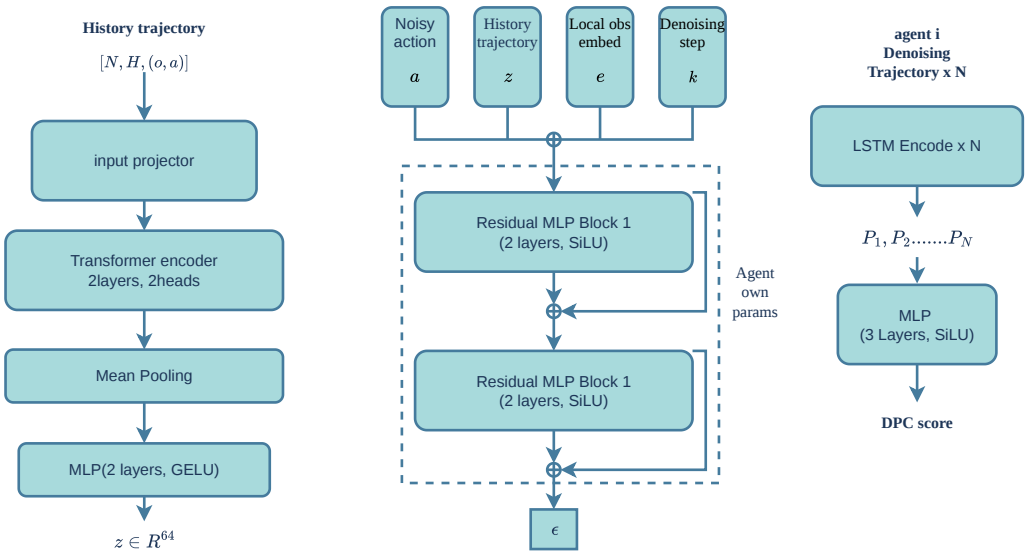


Figure 9: Implementation-level network architecture of DiMAC. Left: a transformer-based history encoder maps each agent's local interaction history to a trajectory latent. Middle: the decentralized diffusion backbone combines the current noisy action, the history latent, the local observation embedding, and the denoising-step embedding to predict the denoising update for each agent. Right: DPC encodes per-agent denoising prefixes with an LSTM-style process encoder and maps the joint representation to a centralized process score.

I.3 Discrete-Action Implementation

For discrete-action tasks such as SMAC, we use a structured parameterization instead of treating the environment action as a single undifferentiated category. If an agent has M action-type choices and E enemy candidates, then the diffusion backbone outputs $M + E$ logits. The first M logits determine the action type, while the remaining E logits represent the target-object preference used only when the chosen action is attack. Legal-action masking is applied before decoding so that unavailable moves, unavailable attacks, and invalid targets are excluded from the executable action set. This design is specific to the SMAC experiments and does not change the main DPC formulation. Appendix H.1 further explains how these late denoising logits are used to build the coordination statistics in the main text.

J Encoder Choice

The implementation separates history encoding from process encoding. Replay history is encoded with a transformer-style model because it summarizes a fixed-length interaction window shared by both the policy and the centralized critic. Denoising prefixes are encoded with an LSTM-style process encoder because DPC repeatedly scores prefixes of different lengths and benefits from an order-sensitive summary of the denoising chain. This separation keeps the implementation simple while matching the different roles of past interaction context and current denoising dynamics.

K Algorithm Details

Algorithms 1 and 2 summarize the centralized training and decentralized execution procedures used in DiMAC. The training algorithm follows the stochastic process-supervision implementation described in Appendix B: each sampled denoising chain contributes three randomly selected denoising steps to DPC bootstrap, reliability weighting, and coalition value decomposition.

Algorithm 1 Centralized Training of DiMAC

Input: $\mathcal{D}, \theta, \phi, \omega, T$
$\bar{\omega} \leftarrow \omega, \quad \bar{\phi} \leftarrow \phi$
Pretrain critic: $\omega \leftarrow \arg\min_\omega \mathcal{L}_Q(\omega)$ Eq. (24)
Optional warm start for DPC: $\phi \leftarrow \arg\min_\phi \mathcal{L}_{term}(\phi)$ Eq. (8)
for $t = 1, \ldots, T$ **do**
  Sample data and denoising chains: $\mathcal{B} \sim \mathcal{D}, \quad \tau \sim p_\theta(\cdot \mid \mathcal{B})$
  Sample three denoising steps per chain: $\{k^{(r)}\}_{r=1}^3 \sim \mathrm{Unif}(\mathcal{I})$
  Update centralized critic: $\omega \leftarrow \omega - \eta_Q \nabla_\omega \mathcal{L}_Q(\omega; \mathcal{B})$
  Update DPC using the sampled steps: $\phi \leftarrow \phi - \eta_\phi \nabla_\phi \mathcal{L}_{DPC}(\phi; \tau)$ Eq. (10)
  Draw three masks per sampled prefix: $m^{(b)} \sim p_m, \quad b = 1, 2, 3$
  Evaluate masked sampled prefixes: $s^{(b)}_{k^{(r)}} \leftarrow C_\phi(\tau_{K:k^{(r)}} \odot m^{(b)}, \mathbf{c}),\ \forall r, b$
  Normalize reliability weights: $\tilde{w}_{k^{(r)}} \propto (\hat{\sigma}^2_{k^{(r)}} + \varepsilon)^{-1}$ within the sampled minibatch Eq. (20)
  Evaluate coalition values: $v_{k^{(r)}}(G) \leftarrow C_\phi(\tau^{[\emptyset]}_{K:k^{(r)}}, \mathbf{c}) - C_\phi(\tau^{[G]}_{K:k^{(r)}}, \mathbf{c}),\ \forall r, G$
  Form singleton and pairwise effects: $\Delta^i_{k^{(r)}} \leftarrow v_{k^{(r)}}(\{i\}) - v_{k^{(r)}}(\emptyset), \quad \Delta^{i,j}_{k^{(r)}} \leftarrow v_{k^{(r)}}(\{i,j\}) - v_{k^{(r)}}(\{i\}) - v_{k^{(r)}}(\{j\}) + v_{k^{(r)}}(\emptyset)$
  Construct stochastic process estimator: $\hat{\mathcal{J}}_{proc} \leftarrow \frac{1}{|\mathcal{B}|} \sum_{n=1}^{|\mathcal{B}|} \frac{1}{3} \sum_{r=1}^3 \tilde{w}_{k_n^{(r)}} \mathcal{J}_{k_n^{(r)}}$
  Update decentralized policy: $\theta \leftarrow \theta - \eta_\theta \nabla_\theta \Big[\mathcal{L}_{diff} + \eta \mathcal{L}_{BC} - \lambda \hat{\mathcal{J}}_{proc}\Big]$
  Update target networks: $\bar{\omega} \leftarrow \rho\omega + (1-\rho)\bar{\omega}, \quad \bar{\phi} \leftarrow \rho\phi + (1-\rho)\bar{\phi}$
end for
Return: decentralized policies $\{\pi_{\theta^i}\}_{i=1}^N$

Algorithm 2 Decentralized Execution of DiMAC

Input: $\mathbf{o} = \{o_i\}_{i=1}^N$ and $\{\theta^i\}_{i=1}^N$
Initialize noisy actions: $a^i_K \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad i = 1, \ldots, N$
for $k = K, K-1, \ldots, 1$ **do**
  Sample reverse noise: $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $k > 1$, otherwise $\epsilon = 0$
  Reverse denoise locally: $a^i_{k-1} \leftarrow \frac{1}{\sqrt{\alpha_k}} \Big(a^i_k - \frac{1-\alpha_k}{\sqrt{1-\bar{\alpha}_k}} \epsilon_{\theta^i}(a^i_k, o_i, k)\Big) + \sigma_k \epsilon,\ \forall i$ Eq. (1)
end for
Apply legal decoding if needed: $\mathbf{a}^\pi_0 \leftarrow \mathrm{Mask}_{legal}(\mathbf{a}^\pi_0)$
Return: final decentralized actions $\{a^i_0\}_{i=1}^N$