# DPM: Dual Preferences-based Multi-Agent Reinforcement Learning

**Sehyeok Kang** [*1] **Yongsik Lee** [*1] **Se-Young Yun** [1]

## Abstract

Multi-agent reinforcement learning (MARL) has demonstrated strong performance across various domains but still faces challenges in sparse reward environments. Preference-based Reinforcement Learning (PbRL) offers a promising solution by leveraging human preferences to transform sparse rewards into dense ones. However, its application in MARL remains under-explored. We propose Dual Preferences-based Multi-Agent Reinforcement Learning (DPM), which extends PbRL to MARL by introducing preferences comparing not only trajectories but also individual agent contributions. Moreover, the research introduces a novel method taking advantage of Large Language Models (LLMs) to gather preferences, addressing challenges associated with human-based preference collection. Experimental results in the StarCraft Multi-Agent Challenge (SMAC) environment demonstrate significant performance improvements over baselines, indicating the efficacy of DPM in optimizing individual reward functions and enhancing performances in sparse reward settings.

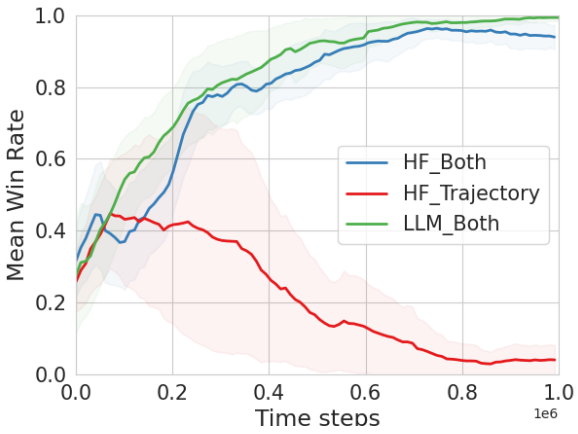


*Figure 1.* Comparison of win rates based on preference types in the SMAC `3m` scenario. `HF_Both` refers to training the reward model using two types of human preferences, `HF_trajectory` indicates training solely with trajectory comparison preferences, and `LLM_Both` represents training with two types of preferences acquired using LLM.

## 1. Introduction

Multi-agent reinforcement learning (MARL) has demonstrated strong performance across various domains (Du & Ding, 2021; Oroojlooy & Hajinezhad, 2023). However, it encounters significant challenges in solving problems within sparse reward environments, where the reward signals are rarely given hence learning the optimal policy is challenging. The situation worsens in MARL since the space is substantially larger than single-agent reinforcement learning, making identification of desirable behaviors more difficult.

Preference-based Reinforcement Learning (PbRL) is one notable approach to addressing sparse reward challenges. By training a reward model based on human preferences, PbRL can transform a sparse reward environment into a dense reward environment, thereby allowing for the facile resolution of issues arising from sparse rewards. Recent works have demonstrated that PbRL effectively solves single-agent reinforcement learning tasks in sparse reward setting or even without rewards from the environments, proving PbRL to be an effective alternative (Lee et al., 2021a; Kim et al., 2023). However, its application in MARL has been explored in only a few studies (Zhu et al., 2024).

Meanwhile, a challenge in applying PbRL to MARL arises from the limitation in optimizing the reward function. Common methods, which depend on a single preference type comparing trajectories, encounter difficulties in accurately assessing the contributions of individual agents, thereby complicating the optimization of individual reward functions. As illustrated by the red line in Figure 1, which represents the win rate in the StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019) `3m` scenario using trajectory comparison preferences, there is no observed performance improvement. This indicates that comparing only trajectories is insufficient for enhancing performance.

[*]Equal contribution [1]KAIST AI. Correspondence to: Se-Young Yun <yunseyoung@kaist.ac.kr>.

*ICML 2024 Workshop on Models of Human Feedback for AI Alignment*, Vienna, Austria. 

In this paper, we propose the **D**ual **P**references-based **M**ulti-Agent Reinforcement Learning (DPM) that applies PbRL to address the sparse reward problem in MARL. The proposed model leverages the characteristics of MARL, allowing for comparisons not only between trajectories but also between agents. By utilizing preferences that compare contributions among agents, it becomes possible to optimize individual reward functions. The blue line in Figure 1 represents the win rate when using two types of preferences. Convergence to a higher win rate is achieved compared to when only one type of preference is used. Therefore, we collect two types of preferences: trajectory comparisons and agent comparisons, and use them to train the reward model.

Additionally, our proposed model introduces a method to overcome the limitations associated with human preferences by utilizing a Large Language Model (LLM) to collect preferences. When acquiring preferences from humans, challenges such as high costs, inconsistency in preferences among individuals, and the possibility of human error are present. To address these challenges, researches have been conducted on gathering preferences through LLMs, which are believed to possess a level of comprehension akin to that of humans (Bai et al., 2022; Lee et al., 2023). The green line in Figure 1 represents the results obtained when preferences were acquired using LLM. It shows no significant difference or even a tendency towards better performance compared to when human preferences were used. In this study, we expand upon this approach to apply it to MARL.

The experiments are conducted in the sparse reward settings of SMAC environment. Our proposed model brings significant performance improvements across various scenarios compared to existing MARL baselines. Furthermore, compared to the cases which rely solely on trajectory comparisons, our method demonstrates more stable convergence and higher win rates, indicating better optimization of individual reward functions through dual preference types.

## 2. Preliminary

### 2.1. A Cooperative Multi-agent Reinforcement Learning

A cooperative MARL task can be formulated as a Dec-PODMP (Oliehoek et al., 2016) which consists of a tuple $\langle S, A, P, R, O, \Omega, n, \gamma \rangle$. $s \in S$ is the global environment state. At each time step, each agent $i \in N \equiv \{1, ..., n\}$ obtains an observation $o^i \in O$ with the observation function $\Omega(s, i) : S \times N \rightarrow O$, and selects an action $a^i \in A$ which forms a joint action $\mathbf{a} = \{a^1, \cdots, a^n\} \in A^n$. Then the environment follows the transition function $P(s'|s, \mathbf{a}) : S \times A^n \times S \rightarrow [0, 1]$ and all the agents share the same reward function $r(s, \mathbf{a}) : S \times A^n \rightarrow \mathbb{R}$. The objective is to learn a joint policy $\pi$ to maximize the expected return $\mathbb{E}_{s_{t+1:\infty}, \mathbf{a}_{t+1:\infty} \sim \pi}[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t, \mathbf{a}_t]$ with $\gamma \in [0, 1)$.

In sparse-reward setting, non-zero rewards $r(s, \mathbf{a})$ are rarely given (e.g., when the given task is completed). To address the sparse-reward challenge, various approaches have been proposed including intrinsic motivation for exploration (Gronauer & Diepold, 2022), subgoal-based methods (Tang et al., 2018; Jeon et al., 2022), and influence-based methods (Jaques et al., 2019; Li et al., 2022).

### 2.2. Preference-based Reinforcement Learning

Preference-based Reinforcement Learning (PbRL) is an alternative approach for complex tasks where designing a suitable reward function is difficult. In PbRL, the agent's learning is also guided by a preference between difference behaviors rather than just a single scalar feedback from the environment. The source of preferences could be human feedback (Christiano et al., 2017; Casper et al., 2023), a scripted teacher which assign preferences according to true task rewards, human feedback (Lee et al., 2021b;a), or AI feedback (Bai et al., 2022; Lee et al., 2023).

A common approach for PbRL is to assign preferences over two trajectory segments (Christiano et al., 2017). A segment $\sigma_s$ is a sequence of observations and actions during $k$ timesteps $\{s_0, \mathbf{a}_0, \cdots, s_k, \mathbf{a}_k\}$ in single-RL, and we generate preference labels $y \in \{0, 0.5, 1\}$ for each segment pair $(\sigma_s^1, \sigma_s^2)$ where $y = 0$ and $y = 1$ mean $\sigma_s^1$ and $\sigma_s^2$ is preferred, respectively, and $y = 0.5$ implies both segments are equally preferable. Following the Bradley-Terry model (Bradley & Terry, 1952), the probability of the preference can be defined as:

$$P_\psi[\sigma_s^1 \succ \sigma_s^2] = \frac{\exp(\sum_t \hat{r}_\psi(\mathbf{s}_t^1, \mathbf{a}_t^1))}{\sum_{i \in \{1,2\}} \exp(\sum_t \hat{r}_\psi(\mathbf{s}_t^i, \mathbf{a}_t^i))} \quad (1)$$

where $\sigma_s^1 \succ \sigma_s^2$ indicates $\sigma_s^1$ is preferred to $\sigma_s^2$, and $\hat{r}_\psi$ is a learnable reward function from preferences. Given the preference dataset $\mathcal{D}_s = \{(\sigma_s^1, \sigma_s^2, y)\}$, the loss for $\hat{r}_\psi$ is:

$$\mathcal{L}(\hat{r}_\psi) = -\mathbb{E}_{(\sigma_s^1, \sigma_s^2, y) \sim \mathcal{D}} \Big[ (1-y) \log P_\psi[\sigma_s^1 \succ \sigma_s^2] + y \log P_\psi[\sigma_s^2 \succ \sigma_s^1] \Big] \quad (2)$$

## 3. Method: DPM

In this section, we present Dual Preferences-based Multi-Agent Reinforcement Learning (DPM), which applies preference-based learning to multi-agent systems based on dual preferences. DPM not only offers a solution to the sparse reward problem but also replaces traditional human preferences with large language model-based preferences, thereby addressing the issues associated with human preferences. DPM is based on an off-policy and online learning MARL algorithm such as QMIX (Rashid et al., 2018).

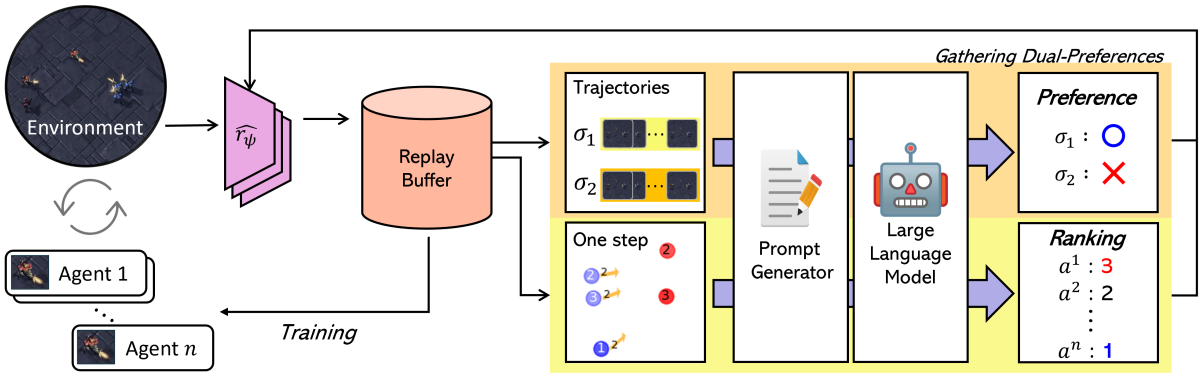


*Figure 2.* The overall framework of DPM

### 3.1. Overview

The overall structure of DPM is illustrated in Figure 2. DPM comprises reward models learned from preferences, which generate intrinsic rewards. This process effectively transforms sparse rewards into dense rewards. Transition data from the environment and intrinsic rewards are stored in the replay buffer and utilized in the policies training.

DPM trains the reward models based on two types of preferences. One involves comparing trajectory pairs, while the other entails ranking the actions of the agents in one scene. Preferences are obtained using LLM. To utilize LLM, vector-based transition information must be transformed into text-based prompts. Therefore, a prompt generator is utilized to convert transition information into text format for input into the LLM. The LLM utilizes the provided information to generate preferences or rankings. Then the generated preferences (or rankings) are used to train the reward model.

### 3.2. Dual-Preferences

DPM utilizes two types of preferences to train the reward models. One is trajectory comparison preference, which selects the better trajectory through comparison, and the other is agent comparison preference, which ranks the actions of agents in a single step. Trajectory comparison is similar to a common approach in PbRL and consistent with Section 2.2.

**Trajectory Comparison**: In a multi-agent concept, a trajectory segment includes additional observations compared to a single RL segment $\sigma = \{(s_0, \mathbf{o}_0, \mathbf{a}_0), \cdots, (s_k, \mathbf{o}_k, \mathbf{a}_k)\}$ where $\mathbf{o}$ denotes observations of all the agents. Two trajectory segments are sampled from the replay buffer to generate a preference label ($y$) and we save the pair and the label into the dataset $\mathcal{D}_T = \{(\sigma^1, \sigma^2, y)\}$.

**Agent Comparison**: Agent comparison involves preferences in the form of rankings. In a given step, the actions of agents $(a_t^1, a_t^2, \ldots, a_t^n)$ are ranked according to their contributions. Therefore, when the state $s_t$ and the actions $\mathbf{o}_t$ are provided, LLM generates ranking labels $z = \{z_1, z_2, \ldots, z_n\}$ based on the contributions and we save the dataset $\mathcal{D}_A = \{(s_t, \mathbf{a}_t, \mathbf{o}_t, z)\}$ to the buffer.

### 3.3. Preference Collection via LLM

**Prompt Generation**: To obtain preferences using a Large Language Model (LLM), prompt generation is essential. However, most environments provide state information in the form of vectors or images rather than text. In this research, we use a prompt generator to convert vector-based states into text-based prompts that an LLM can understand. The prompt generator employs a template-based approach, where the provided vector states and actions are substituted into the corresponding sections of the template. The prompt generator effectively converts vector data into text format. However, it is limited in its ability to include all transitions of trajectories in the prompt. Therefore, for trajectory comparison, the prompt only includes the information of the initial state and the end state. Examples of prompts can be found in the Appendix C.

**LLM Choice**: The LLM generates preferences for the given comparison dataset using the prompts created by the prompt generator. We utilize the GPT-4o (Achiam et al., 2023) as the preference generation model. This model is considered to possess human-level judgment capabilities, enabling it to make decisions at a level comparable to that of humans (Bai et al., 2022; Lee et al., 2023),

### 3.4. Trajectory Selection Strategy

To obtain high-quality preference data, it is crucial to select comparison pairs appropriately. In prior PbRL research (Lee et al., 2021b), ensemble-based sampling techniques are employed. This involves assuming rewards generated by multiple reward models as preferences and selecting pairs of trajectories where the preferences do not align.

On the other hand, in DPM, individual reward functions are utilized, necessitating optimization based on individual rewards rather than global rewards which are the sum of individual rewards. However, if trajectory-based sampling, similar to single-RL, is employed, the global reward becomes the criterion, making it challenging to select appropriate trajectories. To address this issue, DPM employs Kendall's Tau (Kendall, 1938) to calculate the degree of consensus among ranking data generated from individual reward functions. The ranking is determined based on the rewards generated by the reward functions, with higher-ranked agents having higher rewards.

Since Kendall's Tau calculates the concordance between pairs of ranking data, to assess the consensus among multiple reward functions, pairwise combinations is performed, followed by averaging the results. If the value is lower than the threshold, the trajectory is added to the list for comparison. Otherwise, the trajectory is excluded from the comparison. The threshold varies with each iteration, decreasing as the iterations progress.

### 3.5. Reward Models

**Structure**: DPM consists of multiple reward functions, with the mean value of the functions serving as the intrinsic reward. In contrast to common MARL approaches that utilize a global (team) reward function, DPM generates rewards individually for each agent by leveraging preferences based on agent comparisons. These reward functions take as input the transition and state information of the agent $(s_t, s_{t+1}, o_t^i, o_{t+1}^i, a_t^i)$ and produce corresponding an intrinsic reward $(\hat{r}_t^i)$. This reward generation process enables DPM to tailor rewards to the specific contributions of each agent, enhancing its effectiveness in multi-agent environments. For more details of the reward models' structure, please refer to Appendix B. Furthermore, DPM adopts multiple reward models, then the intrinsic reward is defined as the average of rewards generated by the reward models.

**Reward Training**: Since the reward models are trained using dual-preferences, it is necessary to derive the appropriate loss function for each preferences type. First of all, the loss function for trajectory comparison preference is similar to common method. To train the reward functions, Bradley-Terry model is used to calculate the probability :

$$P_\psi[\sigma^1 \succ \sigma^2] = \frac{\exp(\sum_t \hat{R}_t^1)}{\sum_{i\in\{1,2\}} \exp(\sum_t \hat{R}_t^i)} \tag{3}$$

which $\hat{R}_t^i$ is the sum of individual rewards at time $t$ in trajectory segment $i$. The corresponding loss function uses cross-entropy and is defined as follows :

$$\mathcal{L}^T = -\mathbb{E}_{(\sigma^1,\sigma^2,y)\sim\mathcal{D}_T}\Big[(1-y)\log P_\psi[\sigma^1 \succ \sigma^2] + y\log P_\psi[\sigma^2 \succ \sigma^1]\Big] \tag{4}$$

Moreover, we utilize agent comparison data applying the Bradley-Terry model to a single step, similar to the action preferences approach in (Wirth et al., 2017).

$$P_\psi[a_t^i \succ a_t^j] = \frac{\exp(\hat{r}_\psi(s_t, a_t^i, o_t^i))}{\sum_{k\in\{i,j\}} \exp(\hat{r}_\psi(s_t, a_t^k, o_t^k))} \tag{5}$$

The loss function is represented as the sum of the cross-entropy values for each pair of agents. In the equation, $\beta_{i\succ j}$ denotes the preference for action $a_t^i$ over action $a_t^j$, and $M$ means the set of agent pairs $M = \{(i,j)|i,j \in N, i \neq j\}$.

$$\mathcal{L}^A = -\mathbb{E}_{(s_t,\mathbf{a}_t,\mathbf{o}_t,z)\sim\mathcal{D}_A}\Big[\sum_{(i,j)\in M}\beta_{i\succ j}\log P_\psi[a_t^i \succ a_t^j]\Big] \tag{6}$$

$$\beta_{i\succ j} := \begin{cases} 0 & z_i > z_j \\ 1 & z_i < z_j \\ 0.5 & z_i = z_j \end{cases} \tag{7}$$

Finally, the loss function resulting from the use of dual-preferences is as follows :

$$\mathcal{L}(\hat{r}_\psi) = \mathcal{L}^T + \mathcal{L}^A \tag{8}$$

## 4. Experiments

### 4.1. Setup

**Environment and Baselines**: We evaluate DPM on StarCraft Multi-agent Challenge (SMAC) environment (Samvelyan et al., 2019) which consists of diverse microcontrol task and is one of the most widely used benchmarks for MARL. For baselines, We compare DPM with the common MARL algorithms including VDN (Sunehag et al., 2017), QMIX (Rashid et al., 2018) and QPLEX (Wang et al., 2020). Furthermore, we also test DPM against MASER (Jeon et al., 2022) which addresses sparse-reward cooperative tasks. We report the average win rates with the standard deviation from three different random seeds. Further details on the experimental setup can be found in Appendix A.

**MARL Algorithms for DPM and Training**: We adopt the Finetuned-QMIX algorithm (Hu et al., 2021) as our baseline for training agents. This algorithm builds upon QMIX (Rashid et al., 2018) incorporating hyper-parameter

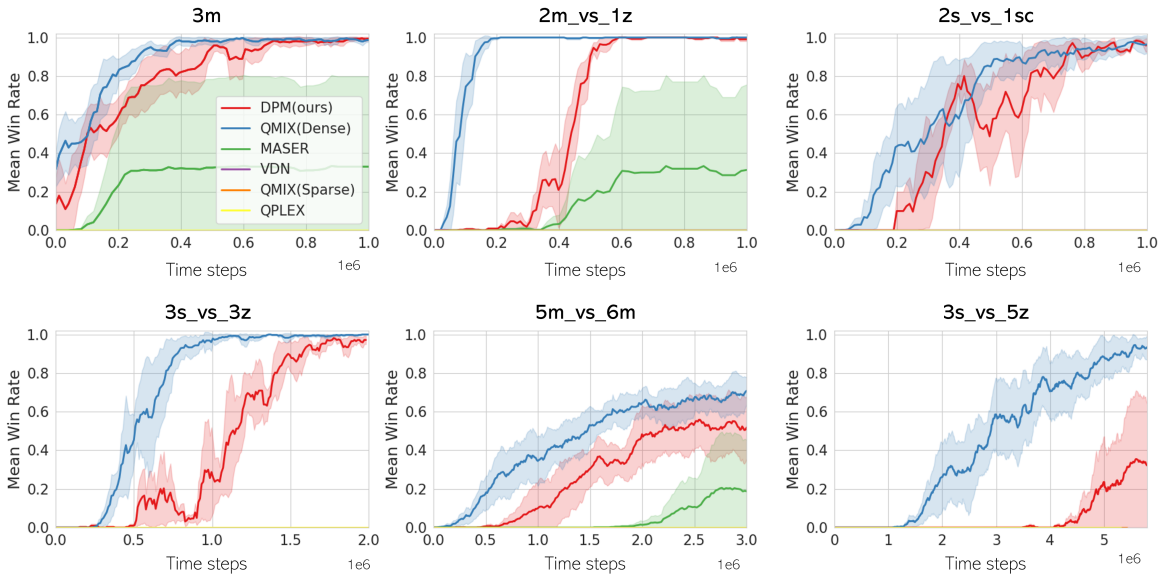


*Figure 3.* Comparison of performance between DPM and baselines in the sparse reward setting of SMAC. In the sparse reward setting of SMAC, a comparison of performance between DPM and baselines reveals that DPM effectively addresses the sparse reward problem when compared to baseline methods

optimization and other enhancements to achieve state-of-the-art performance in dense reward environments. However, in sparse reward environments, its performance remains sub-optimal. To demonstrate that the intrinsic rewards generated by DPM can sufficiently substitute sparse rewards with dense rewards, we train agents using the algorithm. Furthermore, we utilize intrinsic rewards($\hat{r}$) alongside an extrinsic global reward($r^{ext}$) provided by the environment. Then, the reward($r$) used for agent training is as follows :

$$r_t = \sum_{i \in N} \hat{r}_t^i + r_t^{ext} \tag{9}$$

### 4.2. Main Results

In this subsection, we conduct experiments in the sparse reward setting of SMAC to evaluate whether DPM can overcome the sparse reward environment. The experimental results are presented in Figure 3. Across six scenarios, DPM outperformed the baseline algorithms. In the `EASY` scenarios(`3m`, `2m_vs_1z`, `3s_vs_3z`, and `2s_vs_1sc`), DPM achieves a 100% win rate. This is a significant performance improvement compared to dense reward-based algorithms such as QMIX, VDN, and QPLEX, which record a 0% win rate in these scenarios. Additionally, even when compared to MASER, which operates in sparse reward environments, DPM demonstrated superior performance.

The blue line represents the results of the QMIX algorithm operating under dense reward settings. Compared to these results, DPM showed almost similar performance in `3m` and `2s_vs_1sc` scenarios. Although DPM exhibited relatively lower sample efficiency in `2m_vs_1z` and `3s_vs_3z` `scenarios`, it ultimately converged to a 100% win rate.

The `5m_vs_6m` and `3s_vs_5z` scenarios are categorized as `HARD` scenarios in SMAC, posing significant challenges even for algorithms designed to address sparse rewards. However, DPM not only outperformed baseline algorithms (QMIX, VDN, QPLEX) in these scenarios but also achieved victories, demonstrating its robustness.

Based on the results presented in Figure 3, it is evident that applying DPM to QMIX, a dense reward-based algorithm, enables it to solve problems in sparse reward environments. This indicates that DPM effectively transforms a sparse reward environment into a dense reward setting, thereby validating its efficacy in handling sparse reward settings.

### 4.3. Performance analysis of Dual Preferences

DPM optimizes the reward model using two types of preferences. In this subsection, we compare the performance differences between using dual preferences and a single preference type, to highlight the advantages of dual preferences. Additionally, we compare the results of using only ranking preferences. Figure 4 illustrates the performances of models using only one type preference versus those incorporating both trajectory and agent comparison preferences.

The experiments are conducted on `EASY` scenarios in SMAC: `3m`, `2m_vs_1z`, and `2s_vs_1sc`. For both the

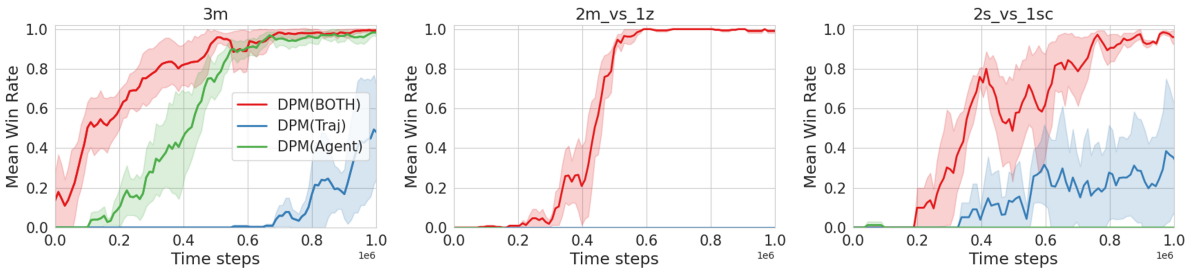


*Figure 4.* Comparing the performance of DPM based on preference types. it is evident that employing dual preferences yields significantly superior performance compared to using a single type of preference.

single and dual preference types, the total number of preferences used is the same. For instance, in this experiment, when using only trajectory comparison preferences, 150 preferences are employed per iteration, leading to a total of 650 preferences over 5 iterations. When utilizing dual preferences, each iteration incorporates 75 trajectory comparison preferences and 75 agent comparison preferences, amounting to a total of 650 preferences over 5 iterations for learning. This setup ensures a balanced utilization of preferences for both types across iterations, maintaining consistency in the learning process.

When using only trajectory comparison, some scenarios did not converge to a high win rate or failed to solve the problem entirely. In contrast, using both types of preferences leads to a convergence to a 100% win rate. Specifically, in the `2m_vs_1z` scenario, the single type approach showed no performance improvement at all. Furthermore, when using only agent comparison preferences, except for the 3m scenario, the win rate converged to 0.

In online learning, unlike offline RL, the agent policy is trained from scratch, making initial policy training crucial. If the reward model is not well-optimized from the beginning, it is difficult to achieve good performance. Using only one type of preference often causes the reward function to fall into a local optimum. Consequently, the policy fails to learn effectively, and the quality of transitions collected subsequently is poor, making it difficult to acquire appropriate preferences in the next iteration. Therefore, using a single type of preference limits policy learning.

**Case study** : To verify the efficacy of dual-preferences in optimizing the individual reward function, we conduct a case study on a single episode of the `3m` scenario. The top of Figure 5 depicts five selected scenes within the episode, describing the states and actions at these steps. The bar graphs display normalized individual reward values, scaled between 0 and 1, generated by reward models trained using different preference types. From top to bottom, the graph represents dual-preferences, trajectory comparison preference only, and agent comparison preference only.

Firstly, when comparing DPM and the trajectory comparison preference only case, we observe that the reward model, which is trained on trajectory comparison preference only, can sometimes assign high rewards even when allied agents die. For instance, at steps 16 and 19, agents 2 and 3 are assigned high rewards despite being killed. Additionally, the model fails to provide appropriate rewards for the situation; at step 9, agent 3 receives a high reward despite not taking any action while being engaged with an enemy. In contrast, using the reward model trained dual preferences ensures the generation of appropriate individual rewards at all steps.

Similarly, when only agent comparison preference is used, the individual reward model does not optimize well. At step 9, agent 2, with low health, is moving toward the enemy, while agent 1, with higher health, is attacking the enemy agent. Ideally, agent 1 should receive a higher reward, but the model assigns a higher reward to agent 2. At step 13, agent 2 receives a higher reward than agents 1 and 3, despite being low on health and under enemy attack, leading to its imminent death in the next step. Thus, the reward model fails to appropriately assess the situation and assigns a high reward incorrectly. These issues do not arise when dual preference types are utilized.

Overall, the graphs demonstrate that dual-preferences effectively mitigate the drawbacks of using a single preference type by leveraging the advantages of both preference types. Therefore, employing dual-preferences positively impacts the optimization of the reward model.

## 5. Conclusion and Limitation

We propose a novel approach called Dual Preference-based Multi-Agent Reinforcement Learning(DPM) for applying preference-based learning to multi-agent reinforcement learning. DPM leverages a reward model trained on preferences to transform sparse reward environments into ones akin to dense reward settings, thus addressing the sparse

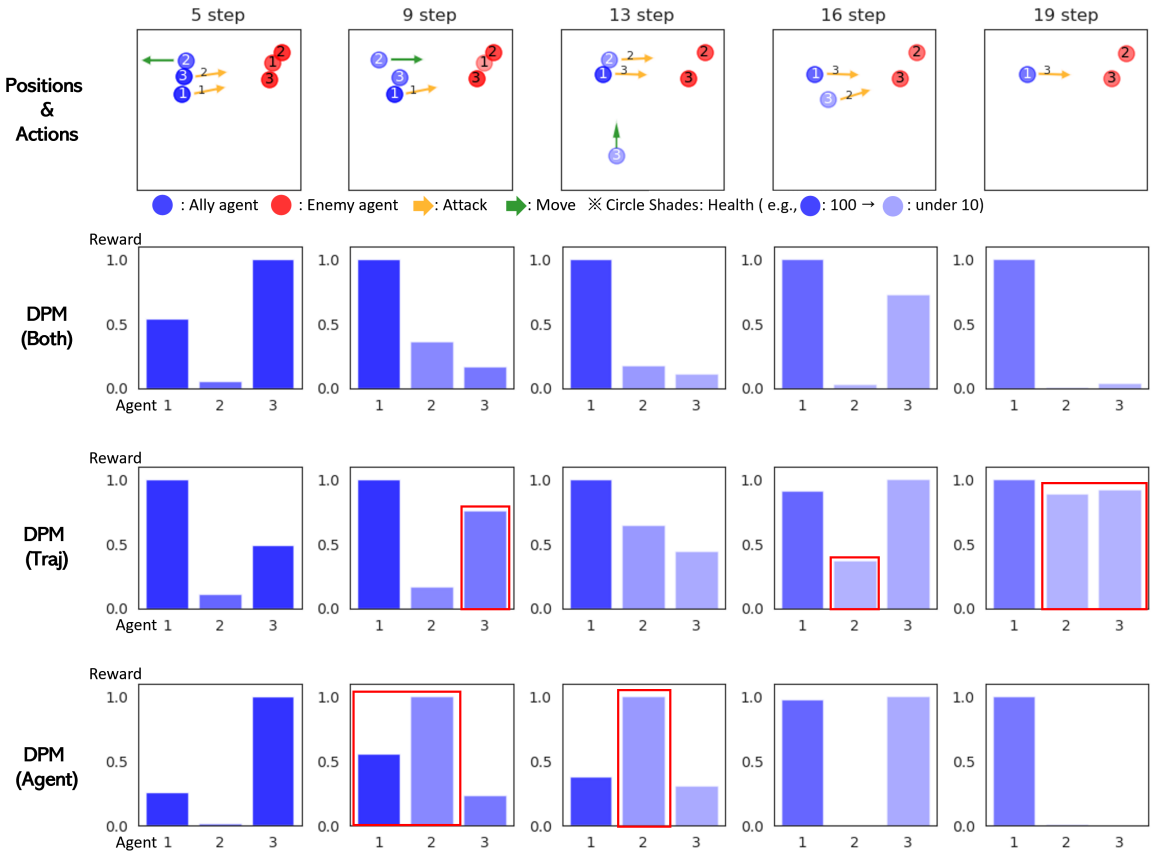


*Figure 5.* A case study for comparing the performance of reward models based on preference types.

reward problem. Moreover, it addresses issues inherent in traditional human-based preference methods by utilizing a large language model to obtain preferences instead of relying solely on human input.

DPM differs from conventional models that solely utilize trajectory comparison preferences by introducing a preference type that compares agents' contributions through ranking. This addition enhances the optimization of the reward model. We evaluate DPM in SMAC, a prominent environment in multi-agent reinforcement learning. DPM demonstrates significant performance improvement compared to baselines in sparse reward settings, and its performance is comparable to that in dense reward settings. This confirms that DPM effectively addresses the sparse reward problem in MARL.

However, there exists a constraint to generate prompts due to the utilization of LLM. It is limited to encapsulate information such as state, observation, actions, etc., within the prompt. To convert vector or image data into text form, additional prepossessing is required. Therefore, we aim for DPM to be more generally applicable across various environments through future research, such as exploring the utilization of Vision-Language Models (VLMs) to effectively substitute non-vector data such as image format data into prompts. This includes investigating methods to seamlessly incorporate data in forms other than vectors into prompts, thereby enhancing the generality of DPM.

We believe that DPM presents a direction for applying preference-based reinforcement learning to multi-agent reinforcement learning effectively. By leveraging Large Language Models (LLMs), we propose an efficient method for addressing problems within this context.

## Broader Impact

Our study introduces Dual Preferences-based Multi-Agent Reinforcement Learning (DPM) to enhance decision-making in spare reward multi-agent system. By addressing the sparse reward problem through preference-based approach which trains reward models by using preferences, our approach has broad applications in real-world scenarios. As the sparse reward cases commonly arise in the real world, DPM would bring improved decision-making across diverse multi-agent environments.

## Acknowledgements

This work was conducted by Center for Applied Research in Artificial Intelligence (CARAI) grant funded by DAPA and ADD (UD230017TD), and supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT)(No.RS-2019-II190075 Artificial Intelligence Graduate School Program(KAIST)).

## References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

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

Casper, S., Davies, X., Shi, C., Gilbert, T. K., Scheurer, J., Rando, J., Freedman, R., Korbak, T., Lindner, D., Freire, P., et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*, 2023.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Du, W. and Ding, S. A survey on multi-agent deep reinforcement learning: from the perspective of challenges and applications. *Artificial Intelligence Review*, 54(5): 3215–3238, 2021.

Gronauer, S. and Diepold, K. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 55(2):895–943, 2022.

Hu, J., Wu, H., Harding, S. A., Jiang, S., and Liao, S. RIIT: rethinking the importance of implementation tricks in multi-agent reinforcement learning. *CoRR*, abs/2102.03479, 2021. URL https://arxiv.org/abs/2102.03479.

Jaques, N., Lazaridou, A., Hughes, E., Gulcehre, C., Ortega, P., Strouse, D., Leibo, J. Z., and De Freitas, N. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International conference on machine learning*, pp. 3040–3049. PMLR, 2019.

Jeon, J., Kim, W., Jung, W., and Sung, Y. Maser: Multi-agent reinforcement learning with subgoals generated from experience replay buffer. In *International Conference on Machine Learning*, pp. 10041–10052. PMLR, 2022.

Kendall, M. G. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

Kim, C., Park, J., Shin, J., Lee, H., Abbeel, P., and Lee, K. Preference transformer: Modeling human preferences using transformers for rl. *arXiv preprint arXiv:2303.00957*, 2023.

Lee, H., Phatale, S., Mansoor, H., Lu, K., Mesnard, T., Bishop, C., Carbune, V., and Rastogi, A. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.

Lee, K., Smith, L., and Abbeel, P. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *arXiv preprint arXiv:2106.05091*, 2021a.

Lee, K., Smith, L., Dragan, A., and Abbeel, P. B-pref: Benchmarking preference-based reinforcement learning. *arXiv preprint arXiv:2111.03026*, 2021b.

Li, P., Tang, H., Yang, T., Hao, X., Sang, T., Zheng, Y., Hao, J., Taylor, M. E., Tao, W., Wang, Z., et al. Pmic: Improving multi-agent reinforcement learning with progressive mutual information collaboration. *arXiv preprint arXiv:2203.08553*, 2022.

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

Oroojlooy, A. and Hajinezhad, D. A review of cooperative multi-agent deep reinforcement learning. *Applied Intelligence*, 53(11):13677–13722, 2023.

Rashid, T., Samvelyan, M., de Witt, C. S., Farquhar, G., Foerster, J. N., and Whiteson, S. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. *CoRR*, abs/1803.11485, 2018. URL http://arxiv.org/abs/1803.11485.

Samvelyan, M., Rashid, T., De Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G., Hung, C.-M., Torr, P. H., Foerster, J., and Whiteson, S. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., and Graepel, T. Value-decomposition networks for cooperative multi-agent learning, 2017.

Tang, H., Hao, J., Lv, T., Chen, Y., Zhang, Z., Jia, H., Ren, C., Zheng, Y., Meng, Z., Fan, C., et al. Hierarchical deep multiagent reinforcement learning with temporal abstraction. *arXiv preprint arXiv:1809.09332*, 2018.

Wang, J., Ren, Z., Liu, T., Yu, Y., and Zhang, C. QPLEX: duplex dueling multi-agent q-learning. *CoRR*, abs/2008.01062, 2020. URL https://arxiv.org/abs/2008.01062.

Wirth, C., Akrour, R., Neumann, G., and Fürnkranz, J. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136): 1–46, 2017.

Zhu, T., Qiu, Y., Zhou, H., and Li, J. Decoding global preferences: Temporal and cooperative dependency modeling in multi-agent preference-based reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17202–17210, 2024.

## A. Experimental Details

In this section, we introduce the environments used in the experiments, the baseline algorithms, as well as the hyperparameters and computational resources. Experiments are carried out on NVIDA A6000 and GTX3090 GPUs and AMD EPYC 7313 CPU.

We conduct experiments in the following environment:

- StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019) from https://github.com/oxwhirl/smac which is licensed under MIT license.

All algorithms are implemented based on the open-source framework *pymarl2* (Hu et al., 2021) from https://github.com/hijkzzz/pymarl2 which is an augmented version of *pymarl* from https://github.com/oxwhirl/pymarl. Both are licensed under Apache License 2.0.

The StarCraft Multi-Agent Challenge (SMAC) is one of the benchmarks widely utilized in research to evaluate MARL algorithms. Units from the strategy video game StarCraft II engage in confrontations with each other in diver scenarios. The objective is for multiple agents to collaborate in defeating the enemies. There are multiple scenarios, each categorized into difficulty levels such as EASY, HARD, and SuperHARD. We primarily conduct experiments in EASY, and HARD scenarios. Table 1 provides a detailed description of the scenarios we used in our experiments.

| Scenario | Difficulty | Ally Units | Enemy Units | Type |
|---|---|---|---|---|
| 2s_vs_1sc | EASY | 2 Stalkers | 1 Spine Crawler | micro-trick: alternating fire |
| 3s_vs_3z | EASY | 3 Stalkers | 3 Zealots | micro-trick: kiting |
| 3m | EASY | 3 Marines | 3 Marines | homogeneous & symmetric |
| 2m_vs_1z | EASY | 2 Marines | 1 Zealot | micro-trick: alternating fire |
| 5m_vs_6m | Super HARD | 5 Marines | 6 Marines | homogeneous & symmetric |
| 3s_vs_5z | HARD | 3 Stalkers | 5 Zealots | micro-trick: kiting |

*Table 1.* A detailed description of the SMAC scenario used in the experiment

## B. Structure of Reward Function

The reward functions adopt a structure based on linear layers, with specific architecture detailed in Table 2. In the experiments, the size of the hidden layer used is 16.

| Name | Type | In features | Out features |
|---|---|---|---|
| input_state | Linear | state size | hidden size |
| input_next_state | Linear | state size | hidden size |
| input_obs | Linear | observation size | hidden size |
| input_actions | Linear | action size | hidden size |
| hidden layer | Linear | hidden size $\times$ 4 | hidden size |
| output | Linear | hidden size | 1 |

*Table 2.* Structure of DPM's reward functions

## C. Prompt Examples

We list and discuss the prompts we employ in conducting the experiments. The prompt consists of four stages : LLM system configuration, environment description, providing information about comparisons, and task instructions.

### C.1. LLM system configuration

In the LLM system configuration, the LLM is endowed with roles and context awareness to enable it to generate high-quality responses.

```
You are a helpful and honest judge of good game playing and progress
in the StarCraft Multi-Agent Challenge game. Always answer as helpfully as possible,
while being truthful.  If you don't know the answer to a question, please don't share
false information.

I'm looking to have you evaluate a scenario in the StarCraft Multi-Agent Challenge.
Your role will be to assess how much the actions taken by multiple agents in a given
situation have contributed to achieving victory.
```

*Figure 6.* Example of a system configuration prompt for the SMAC `3m` scenario.

### C.2. Environment description

The environment description encompasses a comprehensive overview of the SMAC scenario. It includes the scenario name, composition of allies, composition of adversaries, description of the situation, objectives, and other pertinent details.

```
The basic information for the evaluation is as follows.
 - Scenario : 3m
 - Allied Team Agent Configuration : Three marines
 - Enemy Team Agent Configuration : Three marines
 - Situation Description : The situation involves the allied team and the enemy team engaging in combat,
                           where victory is achieved by defeating all the enemies.
 - Objective : Defeat all enemy agents while ensuring as many allied agents as possible
               survive.

I plan to inform you about the status and actions of the agents in a single scene and I will also show
you the subsequent scene based on the agents' actions. Then, you will need to rank the agents in order
of their contribution to victory based on their actions and status.
```

*Figure 7.* Example of an environment description prompt for the SMAC `3m` scenario.

### C.3. Providing information about the comparisons

This part describes the comparison targets for acquiring preferences. In trajectory comparison and agents comparison, separate prompts exist, each allowing for the provision of information to the LLM by altering the details in the square brackets([]), including state, actions, and other relevant information. For the trajectory comparison case, an example prompt is provided in Figure 8, and for the agent comparison case, an example prompt is given in Figure 9.

```
[Trajectory 1]
1. Final State Information
    1) Allied Agents Health : [h_f_a_1_t_1], [h_f_a_2_t_1], [h_f_a_3_t_1]
    2) Enemy Agents Health : [h_f_e_1_t_1], [h_f_e_2_t_1], [h_f_e_3_t_1]
    3) Number of Allied Deaths : [c_f_a_t_1]
    4) Number of Enemy Deaths : [c_f_e_t_1]
    5) Total Remaining Health of Allies : [r_f_a_t_1]
    6) Total Remaining Health of Enemies : [r_f_e_t_1]
2. Total Number of Steps : [step_1]

[Trajectory 2]
1. Final State Information
    1) Allied Agents Health : [h_f_a_1_t_2], [h_f_a_2_t_2], [h_f_a_3_t_2]
    2) Enemy Agents Health : [h_f_e_1_t_2], [h_f_e_2_t_2], [h_f_e_3_t_2]
    3) Number of Allied Deaths : [c_f_a_t_2]
    4) Number of Enemy Deaths : [c_f_e_t_2]
    5) Total Remaining Health of Allies : [r_f_a_t_2]
    6) Total Remaining Health of Enemies : [r_f_e_t_2]
2. Total Number of Steps : [step_2]
```

*Figure 8.* Example of a description of trajectories prompt for the SMAC `3m` scenario.

```
  1. The Scene Information
    1) Allied Agents Information
        - Ally Agent 1's Location : ([a_1_x_1],[a_1_y_1]) / Ally Agent 1's Health : [a_1_h_1]
          * Ally Agent 1's Action : [a_1_a_1]
        - Ally Agent 2's Location : ([a_2_x_1],[a_2_y_1]) / Ally Agent 2's Health : [a_2_h_1]
         * Ally Agent 2's Action : [a_2_a_1]
        - Ally Agent 3's Location : ([a_3_x_1],[a_3_y_1]) / Ally Agent 3's Health : [a_3_h_1]
         * Ally Agent 3's Action : [a_3_a_1]
    2) Enemy Agents Information
        - Enemy Agent 1's Location : ([e_1_x_1],[e_1_y_1]) / Enemy Agent 1's Health [e_1_h_1]
        - Enemy Agent 2's Location : ([e_2_x_1],[e_2_y_1]) / Enemy Agent 2's Health [e_2_h_1]
        - Enemy Agent 3's Location : ([e_3_x_1],[e_3_y_1]) / Enemy Agent 3's Health [e_3_h_1]

  2. The Next Scene Information
    1) Allied Agents Information :
        - Ally Agent 1's Location : ([a_1_x_2],[a_1_y_2]) / Ally Agent 1's Health : [a_1_h_2]
        - Ally Agent 2's Location : ([a_2_x_2],[a_2_y_2]) / Ally Agent 2's Health : [a_2_h_2]
        - Ally Agent 3's Location : ([a_3_x_2],[a_3_y_2]) / Ally Agent 3's Health : [a_3_h_2]
    2) Enemy Agents Information :
        - Enemy Agent 1's Location : ([e_1_x_2],[e_1_y_2]) / Enemy Agent 1's Health [e_1_h_2]
        - Enemy Agent 2's Location : ([e_2_x_2],[e_2_y_2]) / Enemy Agent 2's Health [e_2_h_2]
        - Enemy Agent 3's Location : ([e_3_x_2],[e_3_y_2]) / Enemy Agent 3's Health [e_3_h_2]
```

*Figure 9.* Example of a description of state and agent actions prompt for the SMAC `3m` scenario.

### C.4. Task instructions

The task instruction part provides detailed instructions regarding the output that the LLM should generate. In the trajectory comparison case, the LLM should produce preferences, while in the agent comparison case, it should generate rankings. Therefore, they have different prompt formats to facilitate these distinct tasks.

```
Your task is to inform which one is better between [Trajectory1] and [Trajectory2] based on the information
mentioned above. For example, if [Trajectory 1] seems better, output #1, and if [Trajectory 2] seems better,
output #2. If it's difficult to judge or they seem similar, please output #0.
* Important : Generally, it is considered better when fewer allied agents are killed or injured while
              inflicting more damage on the enemy.
```

*Figure 10.* Example of a task instruction(trajectory comparison) prompt for the SMAC `3m` scenario.

```
Your task is to rank the agents in order of their contribution to victory based on their actions
and inform me of their rankings. Rankings must be displayed for all allied agents, even if a
specific agent has made no contribution. In cases where there is absolutely no contribution,
the lowest ranking should be assigned. For example, if there are three ally agents and their
contributions to victory are greatest in the order of agent 3, 1, 2,
then you should output like below :
Rank #1 : {3}
Rank #2 : {1}
Rank #3 : {2}

Moreover, if the contributions are deemed equal, assign the same rank. For example, if agent 1
and 2 contributed equally and agent 3 contributed the most, output like below :
Rank #1 : {3}
Rank #2 : {1,2}
```

*Figure 11.* Example of a task instruction(agents comparison) prompt for the SMAC `3m` scenario.

## C.5. Full prompt

```
You are a helpful and honest judge of good game playing and progress in the StarCraft Multi-Agent Challenge game.
Always answer as helpfully as possible, while being truthful.
If you don't know the answer to a question, please don't share false information.
I'm looking to have you evaluate a scenario in the StarCraft Multi-Agent Challenge. Your role will be to assess
how much the actions taken by multiple agents in a given situation have contributed to achieving victory.

The basic information for the evaluation is as follows.

- Scenario : 3m
- Allied Team Agent Configuration : Three marines(Marines are long-range attack units in StarCraft 2).
- Enemy Team Agent Configuration : Three marines(Marines are long-range attack units in StarCraft 2).
- Situation Description : The situation involves the allied team and the enemy team engaging in combat,
                          where victory is achieved by defeating all the enemies.
- Objective : Defeat all enemy agents while ensuring as many allied agents as possible survive.

I will provide you with two trajectories, and you should select the better trajectory based on the outcomes of
these trajectories. Regarding the trajectory, it will inform you about the initial and final states,
and you should select the better case based on these two trajectories.

[Trajectory 1]
1. Final State Information
    1) Allied Agents Health : [h_f_a_1_t_1], [h_f_a_2_t_1], [h_f_a_3_t_1]
    2) Enemy Agents Health : [h_f_e_1_t_1], [h_f_e_2_t_1], [h_f_e_3_t_1]
    3) Number of Allied Deaths : [c_f_a_t_1]
    4) Number of Enemy Deaths : [c_f_e_t_1]
    5) Total Remaining Health of Allies : [r_f_a_t_1]
    6) Total Remaining Health of Enemies : [r_f_e_t_1]
2. Total Number of Steps : [step_1]

[Trajectory 2]
1. Final State Information
    1) Allied Agents Health : [h_f_a_1_t_2], [h_f_a_2_t_2], [h_f_a_3_t_2]
    2) Enemy Agents Health : [h_f_e_1_t_2], [h_f_e_2_t_2], [h_f_e_3_t_2]
    3) Number of Allied Deaths : [c_f_a_t_2]
    4) Number of Enemy Deaths : [c_f_e_t_2]
    5) Total Remaining Health of Allies : [r_f_a_t_2]
    6) Total Remaining Health of Enemies : [r_f_e_t_2]
2. Total Number of Steps : [step_2]

Your task is to inform which one is better between [Trajectory1] and [Trajectory2] based on the information
mentioned above. For example, if [Trajectory 1] seems better, output #1, and if [Trajectory 2] seems better,
output #2. If it's difficult to judge or they seem similar, please output #0.
* Important : Generally, it is considered better when fewer allied agents are killed or injured while
              inflicting more damage on the enemy.
```

*Figure 12.* Example of a full trajectory comparison prompt for the SMAC `3m` scenario.

```
You are a helpful and honest judge of good game playing and progress
in the StarCraft Multi-Agent Challenge game. Always answer as helpfully as possible,
while being truthful.  If you don't know the answer to a question, please don't share
false information.

I'm looking to have you evaluate a scenario in the StarCraft Multi-Agent Challenge.
Your role will be to assess how much the actions taken by multiple agents in a given
situation have contributed to achieving victory.

The basic information for the evaluation is as follows.
 - Scenario : 3m
 - Allied Team Agent Configuration : Three marines
 - Enemy Team Agent Configuration : Three marines
 - Situation Description : The situation involves the allied team and the enemy team engaging in combat,
                           where victory is achieved by defeating all the enemies.
 - Objective : Defeat all enemy agents while ensuring as many allied agents as possible
               survive.

I plan to inform you about the status and actions of the agents in a single scene and I will also show
you the subsequent scene based on the agents' actions. Then, you will need to rank the agents in order
of their contribution to victory based on their actions and status.


1. The Scene Information
  1) Allied Agents Information
      - Ally Agent 1's Location : ([a_1_x_1],[a_1_y_1]) / Ally Agent 1's Health : [a_1_h_1]
        * Ally Agent 1's Action : [a_1_a_1]
      - Ally Agent 2's Location : ([a_2_x_1],[a_2_y_1]) / Ally Agent 2's Health : [a_2_h_1]
       * Ally Agent 2's Action : [a_2_a_1]
      - Ally Agent 3's Location : ([a_3_x_1],[a_3_y_1]) / Ally Agent 3's Health : [a_3_h_1]
       * Ally Agent 3's Action : [a_3_a_1]
  2) Enemy Agents Information
      - Enemy Agent 1's Location : ([e_1_x_1],[e_1_y_1]) / Enemy Agent 1's Health [e_1_h_1]
      - Enemy Agent 2's Location : ([e_2_x_1],[e_2_y_1]) / Enemy Agent 2's Health [e_2_h_1]
      - Enemy Agent 3's Location : ([e_3_x_1],[e_3_y_1]) / Enemy Agent 3's Health [e_3_h_1]

2. The Next Scene Information
  1) Allied Agents Information :
      - Ally Agent 1's Location : ([a_1_x_2],[a_1_y_2]) / Ally Agent 1's Health : [a_1_h_2]
      - Ally Agent 2's Location : ([a_2_x_2],[a_2_y_2]) / Ally Agent 2's Health : [a_2_h_2]
      - Ally Agent 3's Location : ([a_3_x_2],[a_3_y_2]) / Ally Agent 3's Health : [a_3_h_2]
  2) Enemy Agents Information :
      - Enemy Agent 1's Location : ([e_1_x_2],[e_1_y_2]) / Enemy Agent 1's Health [e_1_h_2]
      - Enemy Agent 2's Location : ([e_2_x_2],[e_2_y_2]) / Enemy Agent 2's Health [e_2_h_2]
      - Enemy Agent 3's Location : ([e_3_x_2],[e_3_y_2]) / Enemy Agent 3's Health [e_3_h_2]

Your task is to rank the agents in order of their contribution to victory based on their actions
and inform me of their rankings. Rankings must be displayed for all allied agents, even if a
specific agent has made no contribution. In cases where there is absolutely no contribution,
the lowest ranking should be assigned. For example, if there are three ally agents and their
contributions to victory are greatest in the order of agent 3, 1, 2,
then you should output like below :
Rank #1 : {3}
Rank #2 : {1}
Rank #3 : {2}

Moreover, if the contributions are deemed equal, assign the same rank. For example, if agent 1
and 2 contributed equally and agent 3 contributed the most, output like below :
Rank #1 : {3}
Rank #2 : {1,2}
```

*Figure 13.* Example of a full agents comparison prompt for the SMAC `3m` scenario.