M^{3}HF: Multi-agent Reinforcement Learning from Multi-phase Human Feedback of Mixed Quality

Ziyan Wang [1] **Zhicheng Zhang** [2] **Fei Fang** [2] **Yali Du** [1]

Abstract

Designing effective reward functions in multi-agent reinforcement learning (MARL) is a significant challenge, often leading to suboptimal or misaligned behaviors in complex, coordinated environments. We introduce Multi-agent Reinforcement Learning from Multi-phase Human Feedback of Mixed Quality (M^{3}HF), a novel framework that integrates multi-phase human feedback of mixed quality into the MARL training process. By involving humans with diverse expertise levels to provide iterative guidance, M^{3}HF leverages both expert and non-expert feedback to continuously refine agents' policies. During training, we strategically pause agent learning for human evaluation, parse feedback using large language models to assign it appropriately and update reward functions through predefined templates and adaptive weight by using weight decay and performance-based adjustments. Our approach enables the integration of nuanced human insights across various levels of quality, enhancing the interpretability and robustness of multi-agent cooperation. Empirical results in challenging environments demonstrate that M^{3}HF significantly outperforms state-of-the-art methods, effectively addressing the complexities of reward design in MARL and enabling broader human participation in the training process.

1. Introduction

Designing effective reward functions for reinforcement learning (RL) agents is a well-known challenge, particularly in complex environments where the desired behaviors are intricate or the rewards are sparse (Singh et al., 2009; Ng et al., 2000). This difficulty is magnified in multi-agent reinforcement learning (MARL) settings, where agents must not only learn optimal individual behaviors but also coordinate with others, leading to an exponential increase in task complexity (Zhang et al., 2021; Oroojlooy & Hajinezhad, 2023; Du et al., 2023). Sparse or hard-to-learn rewards can severely hinder the learning process, causing agents to converge slowly or settle on suboptimal policies (Andrychowicz et al., 2017; Pathak et al., 2017). In such scenarios, relying solely on environmental rewards may be insufficient for effective learning. Incorporating human feedback has thus emerged as a promising approach (Christiano et al., 2017; Knox & Stone, 2009; Ho & Ermon, 2016), since human guidance can provide additional, informative signals that help agents navigate complex tasks more efficiently when intrinsic rewards are inadequate.

To leverage human expertise in accelerating the learning process of MARL agents, we propose the Multi-phase Human Feedback Markov Game (MHF-MG), an extension of the Markov Game that incorporates human feedback across multiple generations of learning. At each generation, agents gather experiences using their current policies but may still struggle under the original reward function. Humans then observe the agents' behaviors, offering feedback that reflects the discrepancy between their own (potentially more expert) policy and the agents' policies. Building on the MHF-MG, we develop Multi-agent Reinforcement Learning from Multi-phase Human Feedback of Mixed Quality (M^{3}HF), which operationalizes the MHF-MG by directly integrating feedback into the agents' reward functions. This framework utilizes large language models (LLMs) to parse human feedback of various quality levels, employs predefined templates for structured reward shaping, and applies adaptive weight adjustments to accommodate mixed-quality signals.

In summary, we make the following contributions: (1) We propose the **MHF-MG** to address reward sparsity and complexity in MARL through iterative human guidance; (2) We develop the **M^{3}HF** framework, which leverages LLMs to parse diverse human feedback and dynamically incorporates it into agents' reward functions; (3) We provide a theoretical analysis justifying the use of rollout-based performance estimates and offering a weight decay mechanism that mitigates

[1]King's College London, London, UK [2]Carnegie Mellon University, Pittsburgh, US. Correspondence to: Ziyan Wang <ziyan.wang@kcl.ac.uk>.

Preprint. Under review.

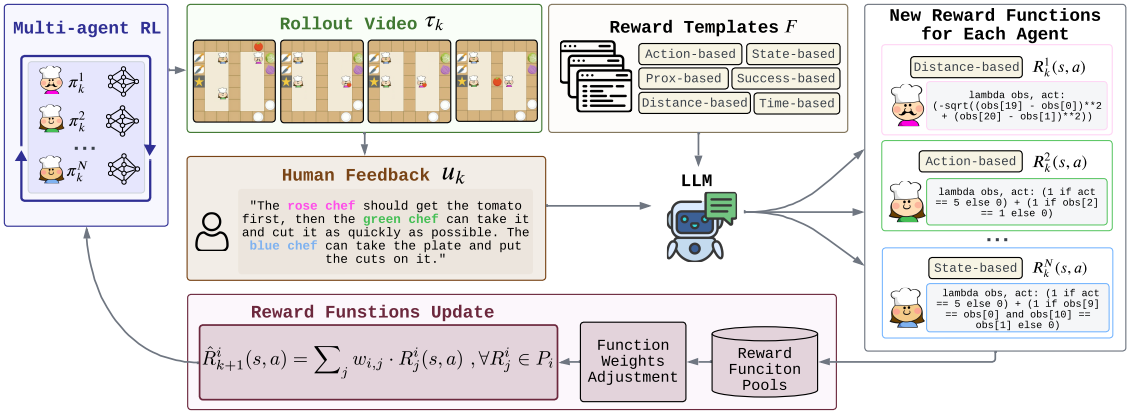


Figure 1. Workflow of the M^{3}HFmethod. Each generation $k \in (0, .., K-1)$ begins with Multi-agent RL training. Agents generate rollout videos τ_k for human evaluation. Human feedback u_k is parsed by a Large Language Model (LLM) into agent-specific instructions. The LLM then selects appropriate reward function templates $f \in F$ and parameterizes them based on the parsed feedback. New reward functions $R^i_k(s,a)$ are added to each agent's reward function pool P_i, with weights $w_{i,m}$ adjusted using performance-based criteria. The updated reward functions $\hat{R}^i_{k+1}(s,a)$ guide the next generation of agents training, creating an iterative loop of multi-agent learning from human feedback.

the impact of low-quality feedback. Extensive experiments in Overcooked demonstrate that M^{3}HF consistently outperforms strong baselines, ultimately providing a robust and flexible method for enhancing multi-agent cooperation under challenging reward structures.

2. Related Work

Multi-Agent Reinforcement Learning (MARL) has been extensively studied to enable agents to learn coordinated behaviours in shared environments (Du et al., 2023; Yu et al., 2022). Traditional MARL approaches often rely on predefined reward functions and suffer from scalability and stability issues arising from the non-stationarity introduced by multiple learning agents (Busoniu et al., 2008; Canese et al., 2021). However, designing appropriate reward functions in MARL remains a significant challenge due to the complexity of agent interactions and the potential for conflicting objectives. To address this, researchers have explored various techniques for reward design. These include credit assignment methods (Nguyen et al., 2018; Zhou et al., 2020), reward shaping (Mannion et al., 2018), and the use of intrinsic rewards (Du et al., 2019). Furthermore, reward decomposition approaches have been proposed to balance individual and team objectives, such as separating rewards into contributions from self and nearby agents (Zhang et al., 2020), or combining dense individual rewards with sparse team rewards (Wang et al., 2022).

Reinforcement Learning from Human Feedback (RLHF). Reinforcement Learning from Human Feedback has emerged as a promising avenue to address the limitations of handcrafted reward functions. Christiano et al. (2017) introduced methods for training agents using human preferences to shape the reward function, demonstrating that human feedback can significantly enhance policy learning. Building on this, Lee et al. (2021) proposed PEBBLE, leveraging unsupervised pre-training and experience relabeling to improve feedback efficiency in interactive RL settings.

While RLHF has been successfully applied to train Large Language Models (LLMs) (Ouyang et al., 2022; Shani et al., 2024), these approaches primarily focus on aligning LLM outputs with human preferences through single-turn interactions and scalar reward signals. Our work differs by incorporating multi-phase, mixed-quality human feedback directly into the reinforcement learning loop of agents in a multi-agent environment, rather than using RLHF to fine-tune LLMs. This allows for richer and more nuanced guidance, enabling agents to adapt their policies based on diverse forms of human input over multiple interaction phases.

Language Models in Reward Design and Policy Learning. Recent advances in LLMs have opened new possibilities for integrating natural language instructions into reinforcement learning. Ma et al. (2023) presented EUREKA, utilizing coding LLMs to achieve human-level reward design, highlighting the potential of LLMs in crafting sophisticated reward functions. Similarly, Liang et al. (2023) proposed Code as Policies, where language model programs are used for embodied control, allowing agents to interpret and execute high-level instructions.

Yu et al. (2023) explored translating language instructions into rewards for robotic skill synthesis, demonstrating that LLMs can bridge the gap between human intentions and machine execution. Kwon et al. (2023) investigated reward design using language models, emphasizing the utility of LLMs in capturing nuanced human preferences. In the

context of human-AI coordination, Hu & Sadigh (2023) introduced methods for language-instructed reinforcement learning, enabling more effective collaboration between humans and agents. Additionally, Liang et al. (2024) proposed learning to learn faster from human feedback using language model predictive control, showcasing the benefits of integrating LLMs to accelerate policy learning through human guidance. While these approaches have made significant strides, they are often limited to single-agent settings or require expert knowledge, restricting their applicability in broader contexts.

Multi-phase Human Feedback. Prior works have considered the role of iterative and multi-phase human feedback in reinforcement learning. Yuan et al. (2022) and Sumers et al. (2022) explored multi-phase bidirectional interactions between humans and agents through predefined communication protocols, which, while structured, limit the flexibility of feedback. Zhi-Xuan et al. (2024) examined the use of human demonstrations via trajectories to convey intentions, requiring humans to perform the task themselves, which can be resource-intensive. Early attempts by Chen et al. (2020) and Zhang et al. (2023) delved into language for task generalization and policy explanation but were constrained to single-agent domains.

3. Preliminaries

We consider a **Markov Game** (Littman, 1994), defined by the tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, in multi-agent reinforcement learning (MARL). Here, $\mathcal{N} = \{1, 2, \ldots, N\}$ represents the set of agents. The state space $\mathcal{S}$ encompasses all possible configurations of the environment, while the action space $\mathcal{A}$ denotes the set of actions available to each agent. At each time step t, the environment is in a state $s_t \in \mathcal{S}$. Each agent $i \in \mathcal{N}$ selects an action $a_t^i \in \mathcal{A}$ according to its policy $\pi^i(a_t^i|s_t)$. The joint action $\mathbf{a}_t = (a_t^1, a_t^2, \ldots, a_t^N)$ leads to a state transition to s_{t+1} according to the transition function $P(s_{t+1}|s_t, \mathbf{a}_t)$. The agents receive a shared reward $r_t = R(s_t, \mathbf{a}_t)$, where $R : \mathcal{S} \times \mathcal{A}^N \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. The objective for each agent is to learn a policy π^i that maximizes the expected cumulative discounted reward:

$$J^i(\pi^i) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \middle| \pi^i, \pi^{-i}\right], \tag{1}$$

where π^{-i} denotes the policies of all agents other than agent i, and the expectation is over the trajectories induced by the policies and the environment dynamics.

In our setting, although the reward function R is known (denoted as original reward function R_{ori}), it is challenging for agents to learn optimal policies due to its sparsity or complexity. This difficulty can lead to slow convergence or suboptimal performance for traditional reinforcement learning algorithms.

4. Method

To address the challenges posed by sparse or complex reward functions in multi-agent environments, we introduce the **Multi-phase Human Feedback Markov Game (MHF-MG)** as a tuple, $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathcal{U}, \pi^h \rangle$. Compared to a standard Markov Game, the added $\mathcal{U}$ denotes the set of possible human utterances or feedback messages. π^h represents the human's policy. In this framework, the agents interact with both the environment and a human over discrete generations indexed by $k = 0, 1, \ldots, K-1$. At each generation k, agents collect experiences by interacting with the environment using their current policies π_k^i. Each generation k consists of multiple iterations, and each iteration involves millions of environment time steps t. This setup allows agents to gain substantial experience within a generation before receiving human feedback. The human possesses a policy π^h, which may be sub-optimal but is assumed to be initially superior to the agents' policies. This human policy provides valuable guidance that can accelerate the agents' learning. The human observes the agents' behaviors and generates utterances $u_k \in \mathcal{U}$ at each generation k, offering feedback based on the discrepancy between their own policy and the agents' current policies.

We model the human's utterances as a mapping f from the human's policy and the agents' policies to the set of possible utterances:

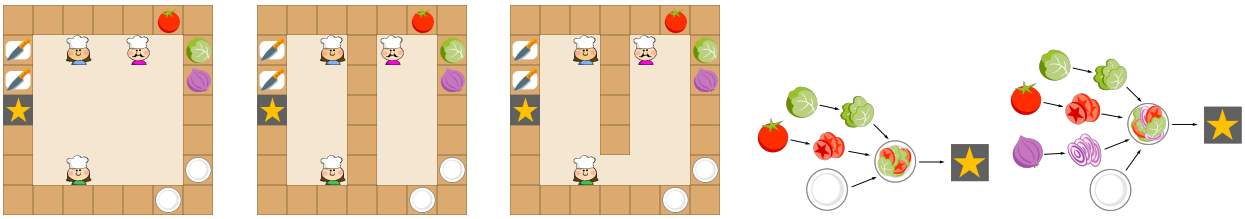
$$u_k = f\left(\pi^h, \pi_k^1, \pi_k^2, \ldots, \pi_k^N\right), \tag{2}$$

where f captures how the human generates feedback by comparing their policy with those of the agents. The utterances may include specific action recommendations, strategic advice, or corrections aimed at guiding the agents toward better performance. The agents parse the human's utterance u_k to extract actionable information. This process may involve natural language understanding techniques, potentially leveraging large language models (LLMs) to interpret the feedback accurately. Based on the parsed feedback, each agent adjusts its reward function to incorporate the human's guidance. For agent i, the updated reward function at generation $k+1$ becomes:

$$R_{k+1}^i(s, \mathbf{a}) = R(s, \mathbf{a}) + R_{\text{hf}}{}_k^i(s, \mathbf{a}, u_k), \tag{3}$$

where $R_{\text{hf}}{}_k^i$ represents the reward adjustment derived from the human's feedback u_k at generation k. This adjustment modifies the reward signal to encourage behaviors aligned with the human's guidance, effectively reshaping the agents' learning objectives.

The agents then update their policies for the next generation by optimizing the expected cumulative reward under the



(a) Overcooked-A (b) Overcooked-B (c) Overcooked-C (d) Lettuce-Tomato salad recipe (e) Lettuce-Onion-Tomato salad recipe

Figure 2. The Overcooked Environment. (a)-(c) The three different kitchen layouts with increasing difficulty: (a) Overcooked-A offers ample movable space; (b) Overcooked-B has less movable space compared to A; (c) Overcooked-C forces agents divided on both sides to cooperate due to the partitioned kitchen. (d)-(e) The two salad recipes: In both recipes, the corresponding chopped foods must be combined on a single plate and delivered. To facilitate training, we use macro-actions based on (Xiao et al., 2022), where the agents' actions are simplified. More details refer to Sec .5.

new reward function $R^{i,k+1}$. Formally, the policy update for agent i is given by:

$$\pi^i_{k+1} = \arg\max_{\pi^i} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R^i_{k+1}(s_t, \mathbf{a}_t) \middle| \pi^i, \pi^{-i}_k\right], \quad (4)$$

where π^{-i}_k denotes the policies of all other agents at generation k, and the expectation is taken over the distribution of trajectories induced by the policies and the environment dynamics. This iterative process continues across generations, with agents repeatedly interacting with the environment, receiving human feedback, and updating their reward functions and policies accordingly. The inclusion of human feedback helps agents navigate the challenges of sparse or complex reward functions by providing additional signals that highlight desirable behaviors and strategies. In this paper, to minimize human involvement, we limit agent-human interactions to at most five times throughout the entire training process, enabling efficient learning with minimal guidance.

4.1. Agent to Human Interaction

In the MHF-MG, agents periodically interact with the human to receive feedback that guides their learning process. This interaction is initiated by the agents, who decide when to seek human input based on specific criteria. The primary mechanism for this interaction is through the generation of rollout trajectories, which approximate the agents' current policy performance.

Rollout Generation and Communication. During each generation k, agents interact with the environment using their current policies π^i_k over multiple iterations, accumulating substantial experience. Despite this extensive exploration, the complexity or sparsity of the original reward function R may hinder the agents' ability to learn optimal policies efficiently.

To address this, agents decide to seek human feedback when certain conditions are met. Specifically, they generate rollout trajectories $\tau_k = \{(s_t, \mathbf{a}_t, r_t)\}_{t=1}^{H}$ over a horizon H, where s_t is the state at time t, $\mathbf{a}_t$ is the joint action of all agents, and r_t is the reward received.

The criteria for generating rollouts are twofold. First, **Periodic Evaluation**, mandates that after every N_{ep} training episodes, agents pause and generate rollouts to keep the human informed of their progress and to receive regular feedback. Second, **Long-term Exploration Failure** refers to a situation in which agents' performance shows minimal improvement over a specified number of episodes, suggesting potential convergence to a suboptimal policy. Formally, we define long-term exploration failure as the condition under which the change in the average cumulative reward over a window of K episodes is below a threshold ϵ:

$$\left| \frac{1}{K} \sum_{e=t-K}^{t-1} J_e - \frac{1}{K} \sum_{e=t-2K}^{t-K-1} J_e \right| < \epsilon, \quad (5)$$

where J_e is the cumulative reward obtained in episode e.

Approximation of Policy Performance via Rollouts.

Consider a stochastic game with stationary policies. Let $\pi^k = (\pi^1_k, \pi^2_k, \ldots, \pi^N_k)$ denote the joint policy of all agents at generation k. The environment dynamics, together with the joint policy π^k, induce a probability distribution over trajectories.

We aim to show that the empirical distribution of states and actions observed in the rollout trajectory τ_k approximates the true distribution under the policy π^k. This allows the human to make informed assessments based on the rollout.

To formalize this, we leverage the concept of *discounted occupancy measures*. A discounted occupancy measure $d_{\pi^k}(s, \mathbf{a})$ represents the discounted visitation frequency of state-action pairs under policy π^k:

$$d_{\pi^k}(s, \mathbf{a}) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t P(s_t = s, \mathbf{a}_t = \mathbf{a} | \pi^k), \quad (6)$$

where γ is the discount factor. We can then define the

expected discounted reward under policy π^k as:

$$J(\pi^k) = \mathbb{E}_{(s,\mathbf{a})\sim d_{\pi^k}}\big[R(s,\mathbf{a})\big], \tag{7}$$

which represents the performance of the policy. In practice, we cannot compute d_{π^k} exactly, but we can estimate it using empirical data from rollouts. Let $\tau_k = \big\{(s_t, \mathbf{a}_t, r_t)\big\}_{t=0}^{H-1}$ be a rollout trajectory generated by executing π^k for H steps.

We define the empirical occupancy measure $\hat{d}_{\pi^k}(s,\mathbf{a})$ as:

$$\hat{d}_{\pi^k}(s,\mathbf{a}) = \frac{1}{H}\sum_{t=0}^{H-1}\delta(s_t = s, \mathbf{a}_t = \mathbf{a}), \tag{8}$$

where δ is the indicator function. Under certain conditions, the empirical occupancy measure $\hat{d}_{\pi^k}$ converges to the true occupancy measure d_{π^k} as $H \to \infty$. Specifically, if the Markov chain induced by π^k is ergodic and the policy is stationary, then the Law of Large Numbers ensures convergence. This allows us to approximate the expected reward using the empirical average reward observed in the rollout:

$$\begin{aligned}\hat{J}(\pi^k) &= \mathbb{E}_{\pi^k}\big[\frac{1}{H}\sum_{t=0}^{H-1} r_t\big] = \mathbb{E}_{(s,\mathbf{a})\sim\hat{d}_{\pi^k}(s,\mathbf{a})}\big[R(s,\mathbf{a})\big] \\ &\approx \mathbb{E}_{(s,\mathbf{a})\sim d_{\pi^k}(s,\mathbf{a})}\big[R(s,\mathbf{a})\big] = J(\pi^k).\end{aligned} \tag{9}$$

Therefore, the rollout trajectory provides an unbiased estimator of the policy's performance, assuming sufficient length and ergodicity.

This approximation is critical for the human to assess the agents' policies accurately based on the rollout. By observing the trajectory τ_k, the human can understand how the agents behave under π^k and identify areas for improvement.

Proposition 4.1. *Assume that the Markov chain induced by policy π^k is ergodic, and the reward function $R(s,\mathbf{a})$ is bounded. Then, the empirical average reward $\hat{J}(\pi^k)$ converges to the expected reward $J(\pi^k)$ almost surely as $H \to \infty$:*

$$\lim_{H\to\infty}\hat{J}(\pi^k) = J(\pi^k). \tag{10}$$

This result relies on the assumption that the policy π^k is fixed during the rollout, and the environment is stationary and ergodic. In stochastic games, where the environment dynamics depend on the joint actions of agents, ensuring ergodicity can be complex. However, if the policies induce sufficient exploration and the environment dynamics satisfy mixing conditions, the convergence holds. We provide a detailed proof in Appendix B.

By leveraging this result, we justify using the rollout trajectory τ_k as a reliable approximation of the agents' policy performance. The human can thus make informed assessments and provide feedback $u_k = f\left(\tau_k; \pi^h\right)$ based on the observed behaviors. Similar assumptions are often made when learning from a fixed dataset of transitions (Levine et al., 2020; Kumar et al., 2019) in the context of offline reinforcement learning and out-of-distribution analysis. Our situation is analogous, as the human evaluates the agents based on the fixed rollout data, which represents the distribution induced by the current policies.

4.2. Human to Agents

Feedback Parsing. Our method employs a Large Language Model (LLM), denoted as $\mathcal{M}$, to parse the human feedback u_k received at generation k and assign it either to specific agents or to all agents collectively. This parsing process is mathematically represented as $u_k^i, u_k^{\text{all}} = \mathcal{M}(u_k, N)$, where N is the number of agents, u_k^i is the feedback assigned to agent i, and u_k^{all} is the feedback applicable to all agents. This approach ensures that each agent receives relevant instructions or corrections based on the human input. The detailed prompts used for guiding the LLM in this parsing process are provided in the Appendix F.4.

Generating New Reward Function The new reward function R_k^i for the current generation involves selecting and parameterizing predefined function templates based on human feedback. For each agent i, the new reward function for the current generation k is generated as follows:

$$R_k^i(s,a) = \mathcal{M}(F, u_k^i, u_k^{\text{all}}, \boldsymbol{e}), \tag{11}$$

where F is a set of predefined function templates, u_k^i is the parsed feedback for agent i at generation k, and $\boldsymbol{e}$ are the entities based on the environment states.

Predefined Function Templates. In our framework, we define a set of predefined reward function templates F that can be parameterized based on human feedback and the specific entities within the environment, as shown in Figure 1. These templates enable the system to systematically generate reward functions aligned with human intentions, facilitating efficient policy updates in response to feedback. The templates capture common interaction patterns such as distance-based rewards that encourage agents to minimize their distance to target entities, action-based rewards that incentivize specific actions, and status-based rewards that reward agents for achieving certain states of the environment. For instance, given the human feedback "I think the red chef needs to be responsible for getting the onion" and the LLM will select the distance-based reward template and parameterize it as:

$$R_k^i(s,a) = -\|s[\text{Agent1.pos}] - s[\text{Onion.pos}]\|_2. \tag{12}$$

Here, the s[Agent1.pos] and s[Onion.pos] are the relevant entities of the observation vector, which encourages Agent 1 (The red chef in the rollout video) to minimize its distance

to the onion, thus aligning its behavior with the desired objective. This structured approach allows agents to interpret and act upon multi-fidelity human feedback effectively. Detailed formulations of these reward templates and additional examples are provided in the Appendix F.2.

Reward Function for the next generation. At the end of the processing of the human feedback in generation k, we conclude the final reward function for each agent to a weighted combination of the base reward and the consistency weight:

$$\hat{R}^i_{k+1}(s,a) = \sum\nolimits_j w_{i,j} \cdot R^i_j(s,a)\ , \forall R^i_j \in P_i \tag{13}$$

Here, $\hat{R}^i_{k+1}(s,a)$ denotes the final reward function for agent i after processing the k-th generation of human feedback, and it will be used for the next generation $k+1$ for the policy training and await the subsequent rollout generation and human interaction, as outlined in Algorithm 1.

One main challenge is how to set the weights $w_{i,j}$ which can effectively balance different reward components and adapt to changing human feedback. To address this challenge, we employ weight decay and performance-based adjustment to optimize the weights $w_{i,j}$.

4.3. Weight Decay and Performance-based Adjustment

The straightforward way to adjust weights is based on a simple weight decay mechanism and performance feedback. When generating a new reward function, we add it to the pool P_i, then set an initial weight, $w_{i,m} = \frac{1}{|P_i|+1}$. Then, we apply a decay to existing weights of the formal reward functions:

$$w_{i,m} = w_{i,m} \cdot \alpha^{M-m}, \forall m \in 1,\dots,M-1, \tag{14}$$

where $\alpha \in (0,1)$ is a constant decay factor. We then normalize all weights by using

$$w_{i,m} = \frac{w_{i,m}}{\sum w_{i,m}}, \forall m \in 1,\dots,M. \tag{15}$$

Additionally, we introduce a performance-based adjustment rule that compares the agent's performance under the original reward function R^i_{ori} across consecutive generations. We calculate ${r_{\text{ori}}}^i_{k+1} - {r_{\text{ori}}}^i_k$, where ${r_{\text{ori}}}^i_{k+1}$ is the performance of the policy trained using the new reward function $\hat{R}^i_{k+1}(s,a)$ (after processing human feedback in generation k) when evaluated on R^i_{ori}, and ${r_{\text{ori}}}^i_k$ is the performance of the policy trained using the previous reward function $R^i_k(s,a)$ (before processing human feedback in generation k) when evaluated on R^i_{ori}. If this difference is positive, it indicates that the new reward function leads to improved performance on the original task. Otherwise, it suggests that the new reward function may be detrimental to the agent's performance on the original task. We then adjust the weight of the newest reward function component $w_{i,m}$ as follows:

$$w_{i,m} = \begin{cases} w_{i,m} + \beta, & \text{if } {r_{\text{ori}}}^i_{k+1} - {r_{\text{ori}}}^i_k > 0, \\ \max(0, w_{i,m} - \beta), & \text{otherwise,} \end{cases} \tag{16}$$

where β is a small adjustment factor. This approach allows for the dynamic adjustment of the reward function pool, emphasizing recent human feedback while maintaining a diverse set of reward components and adapting to performance changes.

4.4. Analysis of the Low Quality Feedback

In this section, we analyze the robustness of the proposed M^{3}HF framework when dealing with human feedback of varying quality, including potentially noisy or erroneous feedback. To formalize this, we model human feedback as a stochastic process with noise and analyze its impact on the learning dynamics. We leverage concepts from robust reinforcement learning and stochastic approximation theory to support our analysis. We consider that at each generation k, the human provides feedback u_k, which is used to generate a new reward function ${R^{\text{hf}}}^i_k$ for each agent i. However, the feedback may contain noise due to misunderstanding, lack of expertise, or other factors. We model the human feedback as:

$${R_{\text{hf}}}^i_k = {R_{\text{true}}}^i_k + \epsilon_{i,k}, \tag{17}$$

where ${R_{\text{true}}}^i_k$ is the ideal reward function that perfectly captures the intended guidance, and $\epsilon_{i,k}$ is a noise term representing the deviation from the ideal feedback. We assume that $\epsilon_{i,k}$ is a zero-mean random variable with bounded variance:

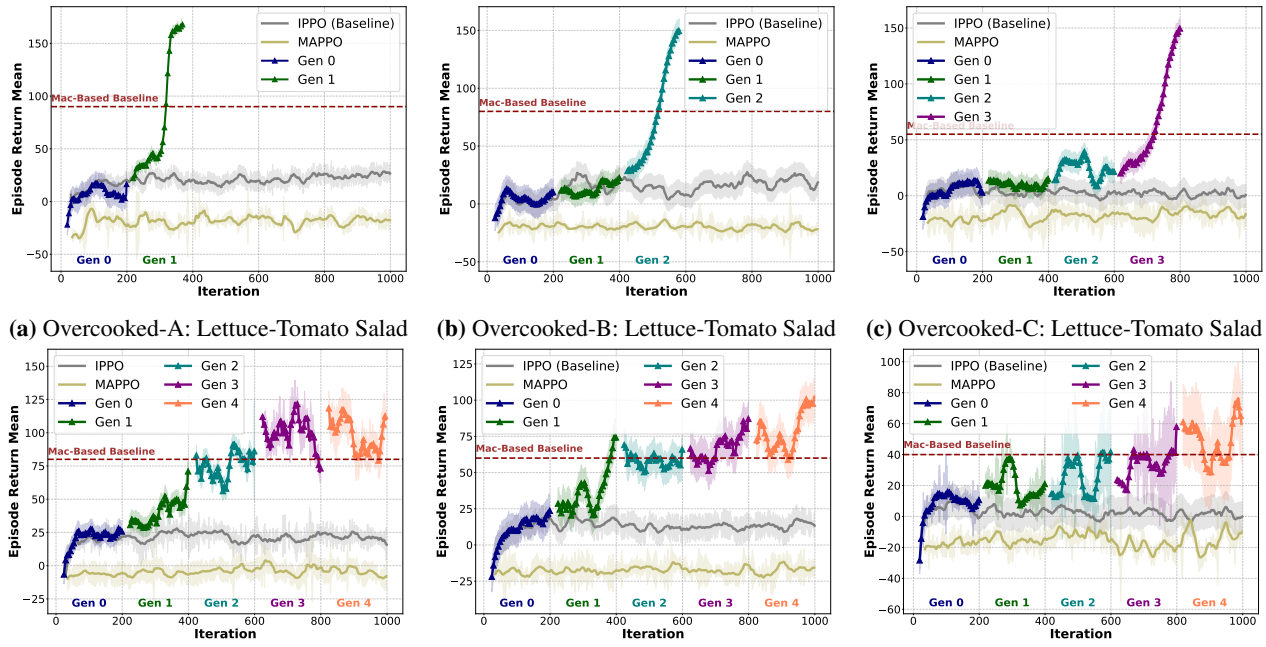
$$\mathbb{E}[\epsilon_{i,k}(s,a)] = 0, \quad \text{Var}[\epsilon_{i,k}(s,a)] \le \sigma^2, \quad \forall s,a. \tag{18}$$

Proposition 4.2 (Robustness to Noisy Human Feedback)**.** *Under Assumptions A.1–A.3 in Appendix A, and assuming that the noise in human feedback is zero-mean and bounded, the expected performance of agent i under the original reward function R^i_{ori} does not degrade over time due to noisy human feedback. Specifically, for all $k > 0$:*

$$\mathbb{E}\left[J^i_{ori}(\pi^i_k)\right] \ge \mathbb{E}\left[J^i_{ori}(\pi^i_0)\right] - \epsilon_k, \tag{19}$$

where ϵ is a small constant from Assumption A.2.

What Proposition 4.2 demonstrates is that our weight adjustment mechanism filters out noisy or harmful human feedback over time, limiting any negative impact on the agent's performance. This is because zero-mean noise does not bias learning on average, the weight decay and performance-based adjustments reduce the influence of unhelpful rewards, and bounded reward functions limit negative effects. Therefore, incorporating multi-quality human feedback is harmless to policy learning in expectation and can be beneficial



(a) Overcooked-A: Lettuce-Tomato Salad **(b)** Overcooked-B: Lettuce-Tomato Salad **(c)** Overcooked-C: Lettuce-Tomato Salad

(d) Overcooked-A : Lettuce-Onion-Tomato **(e)** Overcooked-B : Lettuce-Onion-Tomato **(f)** Overcooked-C : Lettuce-Onion-Tomato

Figure 3. Performance comparison of M^{3}HF against baseline methods across different Overcooked environments and recipes. The plots show the mean episode return over 1000 training iterations (approximately 25k episodes) for (a-c) Lettuce-Tomato salad recipe and (d-f) Lettuce-Onion-Tomato salad recipe in Overcooked layouts A, B, and C, respectively. M^{3}HF consistently outperforms the baseline methods (Mac-based Baseline, IPPO, MAPPO) across all scenarios, with performance improvements becoming more pronounced in more complex environments and recipes. Vertical lines indicate the start of each generation where human feedback is incorporated. All experiments are run with three random seeds, and the shaded areas represent the standard deviation.

when the feedback is accurate. We provide a detailed proof in Appendix C.

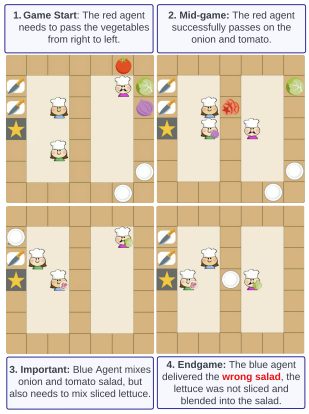
5. Experiment

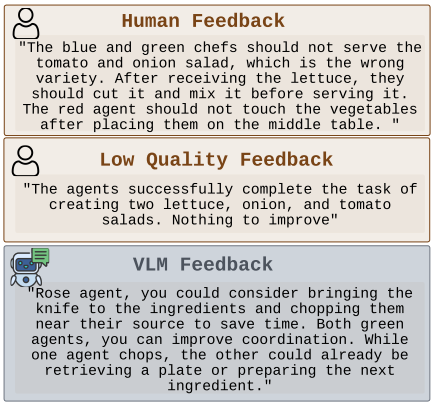
In our experiment, we aim to address three key questions: **Q1.** What is the overall performance of M^{3}HF compared to current state-of-the-art methods? **Q2.** To what extent does multi-quality human feedback impact the performance of M^{3}HF within the same environment? **Q3.** Can Vision-Language Models (VLMs) serve as a scalable and effective alternative to human feedback in M^{3}HF?

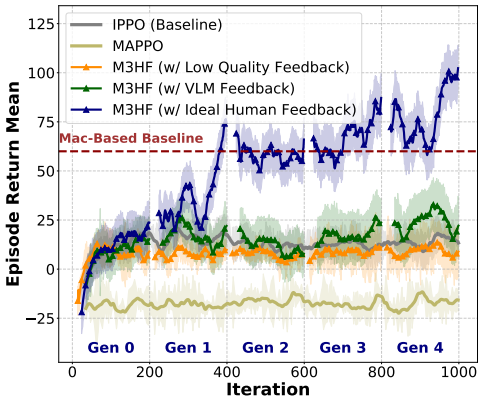
Environment - Macro-Action-Based Overcooked, as shown in Figure 2. In our experiments, we utilize a challenging multi-agent environment based on the Overcooked game (Wu et al., 2021; Xiao et al., 2022), where three agents must learn to cooperatively prepare a correct salad and deliver it to a designated delivery cell. We followed the work of Xiao et al. (2022), where agents operate using macro-actions derived from primitive actions. These macro-actions facilitate effective navigation and interaction within the environment but also introduce complexities in learning optimal policies due to the increased action space. Each agent observes only the *positions* and *statuses* of entities within a 5×5 square centered on itself, introducing partial observability and heightening the coordination challenge. The agents will only receive a significant reward for delivering the correct salad (+200) and punishment if they deliver the wrong salad or food (-50).For more details about the environment setting, please refer to the Appendix E.

Baselines. We evaluate against three strong multi-agent reinforcement learning approaches: The **MAPPO** (Yu et al., 2022), **IPPO** (De Witt et al., 2020), and a **Macro-Action-Based Baseline** from Xiao et al. (2022). Our own framework adopts IPPO as the backbone algorithm, while the macro-action baseline is the average performance of the two best-performing methods in Xiao et al. (2022), namely Mac-IAICC and Mac-CAC, over 25,000 training episodes. Further details on these baselines can be found in Appendix F.

Experiment Results for Question 1: Overall Performance of M^{3}HF Figure 3 demonstrates the superior performance of M^{3}HF compared to SOTA baselines across various Overcooked environments and recipes. Our method consistently outperforms Mac-based Baseline, IPPO, and MAPPO in all scenarios, maintaining a substantial performance advantage across different levels of task complexity. The method exhibits accelerated learning, particularly in early training stages, and achieves higher asymptotic performance levels. Notably, in the simpler recipe setting, Figure 3a, 3b and 3c, M^{3}HF converges to the optimal per-







(a) Example rollout in Generation 3 **(b)** Feedback example from different source **(c)** Overcooked-B : Lettuce-Onion-Tomato Salad

Figure 4. Impact of Mixed-Quality Feedback on Agent Performance. (a) An example is the rollout in Generation 3, where agents exhibit suboptimal behavior due to poor coordination and inefficient task execution. (b) Low-quality feedback provided to the agents, inaccurately stating that they successfully completed the task and offering no constructive guidance for improvement. (c) Performance comparison on Overcooked-B with the Lettuce-Onion-Tomato salad recipe under mixed quality feedback conditions.

formance less than five rounds of interaction, showcasing the method's exceptional efficiency in more straightforward settings. The method's robustness is evident as we move to more complex environments. In the challenging Layout C (Figure 3c and 3f), M^{3}HF maintains its effective performance advantage, particularly outperforming its backbone algorithm IPPO. This consistent superiority across varying complexity levels underscores M^{3}HF's effectiveness and adaptability in diverse multi-agent scenarios.

Experiment Results for Question 2: Impact of Mixed-Quality Human Feedback We evaluated our method when facing low-quality feedback by simulating such feedback at each generation. For example, in the rollout shown in Figure 4a, agents exhibited suboptimal behavior due to poor coordination. Despite this, the low-quality feedback inaccurately stated, "The agents successfully complete the task of creating two lettuce, onion, and tomato salads. Nothing to improve," as depicted in Figure 4b. When training with this irrelevant or erroneous feedback, the agents' performance, illustrated in Figure 4c, remained only slightly below that of the baseline IPPO algorithm and did not degrade significantly. This outcome supports Proposition 4.2, demonstrating that M^{3}HF effectively mitigates the impact of unhelpful feedback through its weight adjustment mechanisms. Even with mixed-quality human input, the framework maintains performance close to the backbone algorithm, showcasing its resilience to low-quality guidance.

Experiment Results for Question 3: VLM-based Feedback Generation We explore the potential of VLMs as an alternative to human feedback. The VLM is given the same video rollouts that humans would observe, sampled at a rate of 1 frame per second. Using all sampled frames and a prompt asking for feedback (detailed in Appendix F.4), the VLM generates feedback to the training agents. In our implementation, we leverage `Gemini-1.5-Pro-002` (Reid et al., 2024), which is chosen for its multimodal understanding capability across a long context. We showcase an example of VLM feedback in Figure 4b alongside the human feedback. Here, the feedback provided by the VLM resembles human-like style but lacks specificity on critical issues, which, in this case, are "wrong variety", "cut it and mix it before serving it". Instead it offers vague suggestions like "improve coordination", which is hard to translate into reward design. This limitation is indicative of the VLM's current inability to perform complex reasoning across images. As a result, when plugged into our M^{3}HF framework, the VLM feedback method does not yield much benefit, as shown in Figure 4c. Nonetheless, we expect improved performance with future advancements in VLM.

Additionally, we provide **further ablation experiments** in the Appendix D that explore each module's independent contribution, as well as compare single-phase and multi-phase feedback settings in greater detail.

6. Conclusion

In this paper, we introduced M^{3}HF, a novel framework for MARL that incorporates multi-phase human feedback of mixed quality to address the challenges of sparse or complex reward signals. By extending the Markov Game to include human input and leveraging LLMs to parse and integrate human feedback, our approach enables agents to learn more effectively. Empirically, M^{3}HF outperforms strong baselines, particularly in scenarios with increased complexity. Our findings highlight the potential of integrating diverse human insights to enhance multi-agent policy learning in a more accessible way.

Impact Statement

This research introduces M^{3}HF, a framework that enables multi-agent reinforcement learning systems to learn effectively from human feedback of varying quality. By allowing non-expert humans to provide meaningful feedback to AI systems, M^{3}HF democratizes the development of multi-agent systems while making them more robust to real-world situations where perfect expert guidance may not be available. This could accelerate the deployment of collaborative AI systems in areas such as healthcare, manufacturing, and emergency response, where multiple agents need to coordinate while incorporating human domain knowledge. While this increased accessibility could lead to broader adoption, we acknowledge the importance of appropriate oversight and encourage future work to explore necessary safeguards for such systems.

References

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Busoniu, L., Babuska, R., and De Schutter, B. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

Canese, L., Cardarilli, G. C., Di Nunzio, L., Fazzolari, R., Giardino, D., Re, M., and Spanò, S. Multi-agent reinforcement learning: A review of challenges and applications. *Applied Sciences*, 11(11):4948, 2021.

Chen, V., Gupta, A., and Marino, K. Ask your humans: Using human instructions to improve generalization in reinforcement learning. *arXiv preprint arXiv:2011.00517*, 2020.

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.

De Witt, C. S., Gupta, T., Makoviichuk, D., Makoviychuk, V., Torr, P. H., Sun, M., and Whiteson, S. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

Du, Y., Han, L., Fang, M., Liu, J., Dai, T., and Tao, D. Liir: Learning individual intrinsic reward in multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Du, Y., Leibo, J. Z., Islam, U., Willis, R., and Sunehag, P. A review of cooperation in multi-agent learning. *arXiv preprint arXiv:2312.05162*, 2023.

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

Hu, H. and Sadigh, D. Language instructed reinforcement learning for human-ai coordination. In *International Conference on Machine Learning*, pp. 13584–13598. PMLR, 2023.

Knox, W. B. and Stone, P. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, pp. 9–16, 2009.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in neural information processing systems*, 32, 2019.

Kwon, M., Xie, S. M., Bullard, K., and Sadigh, D. Reward design with language models. *arXiv preprint arXiv:2303.00001*, 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*, 2021.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., and Zeng, A. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

Liang, J., Xia, F., Yu, W., Zeng, A., Arenas, M. G., Attarian, M., Bauza, M., Bennice, M., Bewley, A., Dostmohamed, A., et al. Learning to learn faster from human feedback with language model predictive control. *arXiv preprint arXiv:2402.11450*, 2024.

Littman, M. L. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pp. 157–163. Elsevier, 1994.

Ma, Y. J., Liang, W., Wang, G., Huang, D.-A., Bastani, O., Jayaraman, D., Zhu, Y., Fan, L., and Anandkumar, A. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

Mannion, P., Devlin, S., Duggan, J., and Howley, E. Reward shaping for knowledge-based multi-objective multi-agent reinforcement learning. *The Knowledge Engineering Review*, 33:e23, 2018.

Meyn, S. P. and Tweedie, R. L. *Markov chains and stochastic stability*. Springer Science & Business Media, 2012.

Ng, A. Y., Russell, S., et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Nguyen, D. T., Kumar, A., and Lau, H. C. Credit assignment for collective multiagent rl with global rewards. *Advances in neural information processing systems*, 31, 2018.

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

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

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

Reid, M., Savinov, N., Teplyashin, D., Lepikhin, D., Lillicrap, T., Alayrac, J.-b., Soricut, R., Lazaridou, A., Firat, O., Schrittwieser, J., et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Shani, L., Rosenberg, A., Cassel, A., Lang, O., Calandriello, D., Zipori, A., Noga, H., Keller, O., Piot, B., Szpektor, I., et al. Multi-turn reinforcement learning from preference human feedback. *arXiv preprint arXiv:2405.14655*, 2024.

Singh, S., Lewis, R. L., and Barto, A. G. Where do rewards come from. In *Proceedings of the annual conference of the cognitive science society*, pp. 2601–2606. Cognitive Science Society, 2009.

Sumers, T., Hawkins, R., Ho, M. K., Griffiths, T., and Hadfield-Menell, D. How to talk so ai will learn: Instructions, descriptions, and autonomy. *Advances in neural information processing systems*, 35:34762–34775, 2022.

Sutton, R. S. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

Wang, L., Zhang, Y., Hu, Y., Wang, W., Zhang, C., Gao, Y., Hao, J., Lv, T., and Fan, C. Individual reward assisted multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 23417–23432. PMLR, 2022.

Wu, S. A., Wang, R. E., Evans, J. A., Tenenbaum, J. B., Parkes, D. C., and Kleiman-Weiner, M. Too many cooks: Bayesian inference for coordinating multi-agent collaboration. *Topics in Cognitive Science*, 13(2):414–432, 2021.

Xiao, Y., Tan, W., and Amato, C. Asynchronous actor-critic for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 35:4385–4400, 2022.

Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., and Wu, Y. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

Yu, W., Gileadi, N., Fu, C., Kirmani, S., Lee, K.-H., Arenas, M. G., Chiang, H.-T. L., Erez, T., Hasenclever, L., Humplik, J., et al. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023.

Yuan, L., Gao, X., Zheng, Z., Edmonds, M., Wu, Y. N., Rossano, F., Lu, H., Zhu, Y., and Zhu, S.-C. In situ bidirectional human-robot value alignment. *Science robotics*, 7(68):eabm4183, 2022.

Zhang, K., Yang, Z., and Başar, T. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, pp. 321–384, 2021.

Zhang, T., Xu, H., Wang, X., Wu, Y., Keutzer, K., Gonzalez, J. E., and Tian, Y. Multi-agent collaboration via reward attribution decomposition. *arXiv preprint arXiv:2010.08531*, 2020.

Zhang, X., Guo, Y., Stepputtis, S., Sycara, K. P., and Campbell, J. Understanding your agent: Leveraging large language models for behavior explanation. 2023.

Zhi-Xuan, T., Ying, L., Mansinghka, V., and Tenenbaum, J. B. Pragmatic instruction following and goal assistance via cooperative language-guided inverse planning. *arXiv preprint arXiv:2402.17930*, 2024.

Zhou, M., Liu, Z., Sui, P., Li, Y., and Chung, Y. Y. Learning implicit credit assignment for cooperative multi-agent reinforcement learning. *Advances in neural information processing systems*, 33:11853–11864, 2020.

A. Assumptions

We make the following assumptions to facilitate the analysis:

Assumption A.1 (Bounded Rewards)**.** All reward functions, including the original reward R^i_{ori}, the true human feedback reward $R_{\text{true}}{}^i_k$, and the noisy human feedback reward $R_{\text{hf}}{}^i_k$, are uniformly bounded:

$$|R(s,a)| \leq R_{\max}, \quad \forall s, a. \tag{20}$$

This assumption is standard in reinforcement learning to ensure stability and convergence (Sutton, 2018).

Assumption A.2 (Learning Algorithm Convergence)**.** Given a fixed reward function, the learning algorithm converges to a policy that is ϵ-optimal with respect to the expected cumulative reward under that reward function:

$$\lim_{t\to\infty} \mathbb{E}[J^i_R(\pi^i_t)] \geq J^R_i(\pi^i_*) - \epsilon, \tag{21}$$

where $J^i_R(\pi)$ is the expected cumulative reward for agent i under policy π and reward function R, and π^i_* is the optimal policy for agent i under R.

Assumption A.3 (Performance Estimation Accuracy)**.** The estimate of the performance difference $\Delta r^i_k = J^i_{\text{ori}}(\pi^i_{k+1}) - J^i_{\text{ori}}(\pi^i_k)$ accurately reflects the true change in expected cumulative reward under the original reward function R^i_{ori} between generations k and $k+1$.

This assumption relies on having sufficient samples to estimate the performance difference accurately, which can be ensured through appropriate exploration and sample size.

B. Proof of Proposition 4.1

Proof. Since the Markov chain is ergodic, it has a unique stationary distribution $\mu(s)$. The state-action pairs observed in the trajectory become samples from the stationary distribution as $H \to \infty$. By the Strong Law of Large Numbers for Markov chains (Meyn & Tweedie, 2012), the sample average converges almost surely to the expected value under the stationary distribution:

$$\lim_{H\to\infty} \mathbb{E}_{\pi^k}\left[\frac{1}{H}\sum_{t=0}^{H-1} R(s_t, \mathbf{a}_t)\right] = \mathbb{E}_{(s,\mathbf{a})\sim d_{\pi^k}(s,\mathbf{a})}\big[R(s,\mathbf{a})\big] = J(\pi^k). \tag{22}$$

□

C. Proof of Proposition 4.2

To prove the proposition, we need to show that the expected decrease in performance due to incorporating noisy human feedback is bounded and that the weight adjustment mechanism reduces the influence of harmful feedback over time.

At each generation k, agent i updates its reward function by combining the existing reward functions in its pool P_i using weights $\{w_{i,m}\}_{m=1}^M$:

$$\hat{R}^i_k(s,a) = \sum_{m=1}^M w^k_{i,m} R^i_m(s,a), \tag{23}$$

where R^i_m are the reward functions in the pool, including the original reward R^i_{ori} and previous human feedback rewards.

The weights are updated using a combination of weight decay and performance-based adjustment:

$$w^{k+1}_{i,m} = \frac{w^k_{i,m}\alpha^{\delta_{m,M}} + \Delta w^k_{i,m}}{\sum_{j=1}^M \left(w^k_{i,j}\alpha^{\delta_{j,M}} + \Delta w^k_{i,j}\right)}, \tag{24}$$

where:

- $\alpha \in (0,1)$ is the decay factor.
- $\delta_{m,M} = 1$ if $m = M$ (i.e., the newest reward function), and 0 otherwise.
- $\Delta w^k_{i,m} = \beta \cdot \text{sign}(\Delta r^k_i) \cdot \mathbb{I}\{m = M\}$.
- $\beta > 0$ is the adjustment factor.

If the performance under the original reward function improves ($\Delta r^i_k > 0$), the weight of the new reward function is increased; otherwise, it is decreased.

Proof. At each generation k, the agent updates its policy π^i_{k+1} to maximize the expected cumulative reward under the combined reward function $\hat{R}^i_k$. The combined reward function can be decomposed as:

$$\hat{R}^i_k = w^k_{i,0} R^i_{\text{ori}} + \sum_{m=1}^M w^k_{i,m}(R_{\text{true}}{}^m_i + \epsilon_{i,m}), \tag{25}$$

where $w^k_{i,0}$ is the weight of the original reward function, and $R_{\text{true}}{}^m_i$ are the true intended reward functions from human feedback.

The expected cumulative reward under $\hat{R}^i_k$ is:

$$J^{\hat{R}^i_k}_i(\pi^i_{k+1}) = w^k_{i,0} J^i_{\text{ori}}(\pi^i_{k+1}) + \sum_{m=1}^M w^k_{i,m}\left(J^{R_{\text{true}}{}^m_i}_i(\pi^i_{k+1}) + \mathbb{E}[\epsilon_{i,m}]\right). \tag{26}$$

Since $\mathbb{E}[\epsilon_{i,m}] = 0$, we have:

$$J^{\hat{R}^i_k}_i(\pi^i_{k+1}) = w^k_{i,0} J^i_{\text{ori}}(\pi^i_{k+1}) + \sum_{m=1}^M w^k_{i,m}\left(J^{R_{\text{true}}{}^m_i}_i(\pi^i_{k+1})\right). \tag{27}$$

The agent aims to maximize $J_i^{\hat{R}_k^i}(\pi_{k+1}^i$. However, due to the noise in the reward functions, the policy may not be optimal with respect to R_{ori}^i.

We analyze the change in performance under R_{ori}^i between generations k and $k+1$:

$$\Delta J_{\text{ori}}^i = J_{\text{ori}}^i(\pi_{k+1}^i) - J_{\text{ori}}^i(\pi_k^i). \tag{28}$$

Our goal is to bound ΔJ_{ori}^i and show that, in expectation, it does not decrease over time.

From Assumption A.2, the learning algorithm seeks to maximize $J_i^{\hat{R}_k^i}(\pi_{k+1}^i)$. The influence of the noisy reward functions diminishes over time due to the weight adjustment mechanism.

Specifically, if incorporating the new reward function leads to a decrease in $J_i^{\text{ori}}(\pi_{k+1}^i)$, the performance-based adjustment reduces $w_{i,M}^{k+1}$. Since the noise $\epsilon_{i,k}$ is zero-mean, the expected impact of the noise on the policy update is zero.

Moreover, since the weights of harmful reward functions are reduced over time, their influence on the combined reward function $\hat{R}_k^i$ decreases. The weight of the original reward function $w_{i,0}^k$ remains significant, ensuring that the agent continues to be guided by R_{ori}^i.

Therefore, the expected decrease in $J_{\text{ori}}^i(\pi_{k+1}^i)$ due to noisy human feedback is bounded and diminishes over time.

Formally, we can write:

$$\mathbb{E}[\Delta J_{\text{ori}}^i] \geq -\epsilon, \tag{29}$$

where ϵ is small due to the boundedness of the reward functions and the learning algorithm's convergence properties.

Accumulating over K generations, we have:

$$\mathbb{E}[J_{\text{ori}}^i(\pi_K^i)] \geq \mathbb{E}[J_{\text{ori}}^i(\pi_0^i)] - \epsilon K. \tag{30}$$

Since ϵ is small and K is finite, the total expected decrease is bounded, and the agent's performance under R_{ori}^i does not degrade significantly due to noisy human feedback.

□

D. Additional Ablation Studies

In this section, we provide two additional ablation experiments conducted in the Overcooked-B: Lettuce-Onion-Tomato Salad scenario. The first experiment (Table 1) compares different approaches to handling mixed-quality human feedback, while the second experiment (Table 2) examines the impact of multiple feedback phases versus a single-phase approach.

Table 1. Comparison of approaches for parsing and utilizing mixed-quality human feedback. "Raw Feedback" directly maps text to reward with minimal structure, whereas "LLM Parsing Only" standardizes feedback but does not apply weight adjustment. "Full M^{3}HF" includes both LLM parsing and performance-based weight adjustment. Results are evaluated under the *original environment reward* in Overcooked-B: Lettuce-Onion-Tomato Salad, averaged over three random seeds.

Method	**Average Return (Mean ± Std)**
Raw Feedback (direct text-to-reward)	45.3 ± 5.2
LLM Parsing Only (no weight adjustment)	68.7 ± 4.1
Full M^{3}HF (parsing + weight adjustment)	**102.7** ± 10.8

Table 2. Comparison of single-phase feedback (one round of human feedback at the beginning) vs. our multi-phase framework (M^{3}HF). Both methods are evaluated on Overcooked-B: Lettuce-Onion-Tomato Salad. "Single-Phase Feedback" follows the paradigm of applying human feedback only once at the start of training, whereas M^{3}HF iteratively collects and integrates human feedback across multiple generations.

Method	**Average Return (Mean ± Std)**
Single-Phase Feedback	43.1 ± 10.3
M^{3}HF (Ours)	**102.7** ± 10.8

Results Discussion. Table 1 demonstrates that directly converting raw text to rewards yields suboptimal performance due to inconsistent interpretation of mixed-quality feedback. Introducing LLM parsing significantly improves performance (from 45.3 to 68.7 in average return), indicating the importance of structuring feedback. Furthermore, adding our performance-based weight adjustment mechanism ("Full M^{3}HF") further increases the average return to 102.7, showing that dynamically adjusting the influence of each feedback component is crucial for mitigating unhelpful or noisy guidance.

Table 2 shows that a single-phase approach, where all human feedback is given at the start of training, achieves an average return of 43.1. In contrast, M^{3}HF's multi-phase strategy attains a substantially higher score of 102.7, highlighting the benefits of iterative feedback. By periodically refining the policy based on evolving human guidance—and filtering or downweighting low-quality instructions—M^{3}HF leverages human expertise more effectively throughout the entire training process.

E. Environment Details

In this section, we will introduce the details of the environments we are using. We follow the setting from

Goal. Three agents need to learn cooperating with each other to prepare a Tomato-Lettuce-Onion salad and deliver it

to the 'star' counter cell as soon as possible. The challenge is that the recipe of making a tomato-lettuce-onion salad is unknown to agents. Agents have to learn the correct procedure in terms of picking up raw vegetables, chopping, and merging in a plate before delivering.

State Space. The environment is a 7×7 grid world involving three agents, one tomato, one lettuce, one onion, two plates, two cutting boards and one delivery cell. The global state information consists of the positions of each agent and above items, and the status of each vegetable: chopped, unchopped, or the progress under chopping.

Primitive-Action Space. Each agent has five primitive-actions: *up*, *down*, *left*, *right* and *stay*. Agents can move around and achieve picking, placing, chopping and delivering by standing next to the corresponding cell and moving against it (e.g., in Figure 2a, the pink agent can *move right* and then *move up* to pick up the tomato).

Macro-Action Space. Here, we first describe the main function of each macro-action and then list the corresponding termination conditions.

- Five one-step macro-actions that are the same as the primitive ones;
- ***Chop***, cuts a raw vegetable into pieces (taking three time steps) when the agent stands next to a cutting board and an unchopped vegetable is on the board, otherwise it does nothing; and it terminates when:
  - The vegetable on the cutting board has been chopped into pieces;
  - The agent is not next to a cutting board;
  - There is no unchopped vegetable on the cutting board;
  - The agent holds something in hand.
- ***Get-Lettuce***, ***Get-Tomato***, and ***Get-Onion***, navigate the agent to the latest observed position of the vegetable, and pick the vegetable up if it is there; otherwise, the agent moves to check the initial position of the vegetable. The corresponding termination conditions are listed below:
  - The agent successfully picks up a chopped or unchopped vegetable;
  - The agent observes the target vegetable is held by another agent or itself;
  - The agent is holding something else in hand;
  - The agent's path to the vegetable is blocked by another agent;
  - The agent does not find the vegetable either at the latest observed location or the initial location;
  - The agent attempts to enter the same cell with another agent, but has a lower priority than another agent.
- ***Get-Plate-1/2***, navigates the agent to the latest observed position of the plate, and picks the vegetable up if it is there; otherwise, the agent moves to check the initial position of the vegetable. The corresponding termination conditions are listed below:
  - The agent successfully picks up a plate;
  - The agent observes the target plate is held by another agent or itself;
  - The agent is holding something else in hand;
  - The agent's path to the plate is blocked by another agent;
  - The agent does not find the plate either at the latest observed location or at the initial location;
  - The agent attempts to enter the same cell with another agent but has a lower priority than another agent.
- ***Go-Cut-Board-1/2***, navigates the agent to the corresponding cutting board with the following termination conditions:
  - The agent stops in front of the corresponding cutting board, and places an in-hand item on it if the cutting board is not occupied;
  - If any other agent is using the target cutting board, the agent stops next to the teammate;
  - The agent attempts to enter the same cell with another agent but has a lower priority than another agent.
- ***Go-Counter*** (only available in Overcook-B, Figure 2 b), navigates the agent to the center cell in the middle of the map when the cell is not occupied, otherwise it moves to an adjacent cell. If the agent is holding an object the object will be placed. If an object is in the cell, the object will be picked up.
- ***Deliver***, navigates the agent to the 'star' cell for delivering with several possible termination conditions:
  - The agent places the in-hand item on the cell if it is holding any item;
  - If any other agent is standing in front of the 'star' cell, the agent stops next to the teammate;
  - The agent attempts to enter the same cell with another agent, but has a lower priority than another agent.

Observation Space: The macro-observation space for each agent is the same as the primitive observation space. Agents are only allowed to observe the *positions* and *status* of the entities within a 5×5 view centered on the agent. The initial position of all the items are known to agents.

Dynamics: The transition in this task is deterministic. If an agent delivers any wrong item, the item will be reset to its initial position. From the low-level perspective, to chop a vegetable into pieces on a cutting board, the agent needs to stand next to the cutting board and executes *left* three times. Only the chopped vegetable can be put on a plate.

Original Reward Function: $+10$ for chopping a vegetable, $+200$ terminal reward for delivering a correct salad (like tomato-lettuce-onion or tomato-lettuce salad), -5 for delivering any wrong entity, and -0.1 for every timestep.

Episode Termination: Each episode terminates either when agents successfully deliver a tomato-lettuce-onion salad or reaching the maximal time steps, 200.

F. Implementation Details

F.1. Algorithm

s In here, we list the complete algorithm, as shown in Algorithm.1.

F.2. Predefined Reward Function Templates

To effectively incorporate human feedback into the learning process, we define a set of predefined reward function templates F that can be parameterized based on the feedback and entities present in the environment. These templates capture common interaction patterns between agents and their environment, facilitating automatic reward function generation aligned with human intentions.

Firstly, the **distance-based reward** function penalizes the agent proportionally to the Euclidean distance between two entities e_1 and e_2 within the environment:

$$f_{\text{dist}}(s, a, e_1, e_2) = -\|s[e_1.\text{pos}] - s[e_2.\text{pos}]\|_2, \quad (31)$$

where $s[e_i.\text{pos}]$ denotes the position vector of entity e_i in state s, and $\|\cdot\|_2$ represents the Euclidean norm.

Secondly, the **action-based reward** function provides a reward when the agent performs a specific desired action a_{desired}:

$$f_{\text{action}}(s, a, a_{\text{desired}}) = \mathbb{I}(a = a_{\text{desired}}), \quad (32)$$

where a is the action taken by the agent, and $\mathbb{I}(\cdot)$ is the indicator function, returning 1 if the condition is true and 0 otherwise.

Thirdly, the **status-based reward** function rewards the agent when an entity e attains a particular desired status $\text{status}_{\text{desired}}$:

$$f_{\text{status}}(s, a, e, \text{status}_{\text{desired}}) = \mathbb{I}(s[e.\text{status}] = \text{status}_{\text{desired}}), \quad (33)$$

where $s[e.\text{status}]$ represents the current status of entity e in state s.

Additionally, we define a **composite reward** function that allows for more nuanced feedback by combining multiple reward components:

$$f_{\text{comp}}(s, a) = \sum_i \lambda_i f_i(s, a), \quad (34)$$

where $f_i(s, a)$ are individual reward components (e.g., $f_{\text{dist}}, f_{\text{status}}$), and λ_i are weighting coefficients that determine the relative importance of each component.

For instance, given the human feedback "Agent 1 needs to get the onion," we might select the distance-based reward template and parameterize it as:

$$R_i(s, a) = -\|s[\text{Agent1.pos}] - s[\text{Onion.pos}]\|_2. \quad (35)$$

This reward function encourages Agent 1 to minimize its distance to the onion, thus aligning its behavior with the desired objective.

Furthermore, other templates can be incorporated depending on the environmental context and task requirements. For example, a **proximity-based reward** function provides a reward when an agent is within a certain distance d of a target entity:

$$f_{\text{prox}}(s, a, e_1, e_2, d) = \begin{cases} r_{\text{prox}}, & \text{if } \|s[e_1.\text{pos}] - s[e_2.\text{pos}]\|_2 \leq d, \\ 0, & \text{otherwise}, \end{cases} \quad (36)$$

where r_{prox} is the reward assigned for being within distance d.

A **time-based penalty** can be introduced to encourage efficient task completion:

$$f_{\text{time}}(s, a, t) = -\beta \cdot t, \quad (37)$$

where t is the current time step, and β is a penalty coefficient reflecting the cost of time.

A **success-based reward** provides a reward upon achieving a specific goal condition:

$$f_{\text{success}}(s, a) = \mathbb{I}(\text{goal\_condition\_met}) \cdot r_{\text{success}}, \quad (38)$$

where r_{success} is the reward value assigned when the goal condition is met.

An **energy-based penalty** discourages unnecessary expenditure of resources:

$$f_{\text{energy}}(s, a) = -\gamma \cdot \text{energy}(a), \quad (39)$$

where $\text{energy}(a)$ represents the energy cost associated with action a, and γ is a scaling factor.

By leveraging these templates, the system can systematically generate reward functions that align with human feedback, enabling agents to adapt their behavior effectively in response to diverse instructions. This approach allows for the incorporation of mixed-quality human feedback into the learning process, enhancing the agents' ability to perform complex tasks in multi-agent environments.

F.3. Training Details

Our experiments were conducted on a heterogeneous computing cluster running Ubuntu Linux. The hardware configuration included a variety of CPU models, such as Dual Intel Xeon E5-2650, Dual Intel Xeon E5-2680 v2, and Dual Intel Xeon E5-2690 v3. For accelerated computing, we utilized 3 NVIDIA A30 GPUs. The total computational resources comprised 180 CPU cores and 500GB of system memory.

Table 3. Hyperparameters used in Overcooked-A, B, and C.

Hyperparameter	M^{3}HF-IPPO	Baseline-IPPO	Baseline-MAPPO
Training Generation	5	-	-
Training Iterations	1000	1000	1000
Training Episodes	25k	25k	25k
Learning Rate	0.0003	0.0003	0.0003
Training Batch Size	5120	5120	5120
SGD Minibatch Size	1024	1024	1024
Number of SGD Iterations	10	10	10
Discount Factor (γ)	0.99	0.99	0.99
GAE Lambda (λ)	0.95	0.95	0.95
Clip Parameter	0.2	0.2	0.2
Value Function Clip Parameter	10.0	10.0	-
Entropy Coefficient	0.01	0.01	0.01
KL Coefficient	0.2	0.2	-
Gradient Clipping	0.5	0.5	0.5

F.4. Prompts

Prompt 1: FEEDBACK PARSING PROMPT

Given the following feedback for a multi-agent system in an Overcooked environment, assign the feedback to appropriate agents or to all agents. The system has **{num_agents}** agents.

Feedback: **{Human Feedback}**

The agent_1 is the chef in Green, agent_2 is the chef in Rose, agent_3 is the chef in Blue.

Return your response in the following JSON format:

{{

"agent_0": "feedback for agent 0",
"agent_1": "feedback for agent 1",
...
"all": "feedback for all agents"

}}

Only include keys for agents that receive specific feedback and 'all' if there's general feedback.

Prompt 2: REWARD FUNCTION BUILD PROMPT

Given the parsed feedback for an agent in an Overcooked environment, select and parameterize a reward function template.

The observation space is a 32-length vector as described in the task description.

Parsed Feedback: **{feedback for this agent}**

Observation Space (32-length vector for each agent):
- Tomato: position (2), status (1) (obs[0:2])
- Lettuce: position (2), status (1) (obs[3:5])
- Onion: position (2), status (1) (obs[6:8])
- Plate 1: position (2) (obs[9:10])
- Plate 2: position (2) (obs[11:12])
- Knife 1: position (2) (obs[13:14])
- Knife 2: position (2) (obs[15:16])

- Delivery: position (2) (obs[17:18])
- Agent 1: position (2) (obs[19:20])
- Agent 2: position (2) (obs[21:22])
- Agent 3: position (2) (obs[23:24])
- Order: one-hot encoded (7) (obs[25:32])

Available function templates:
1. Distance-based: -sqrt((agent_x - target_x)**2 + (agent_y - target_y)**2)
2. Action-based: reward for specific actions (e.g., chopping, picking up)
3. State-based: reward for achieving specific states (e.g., holding an item)
4. Time-based: penalty for time taken
5. Combination of the above

Select a template and parameterize it based on the feedback. Return your response as a Python lambda function that takes the observation vector (obs) and action (act) as input.

For example, Distance between agent 1 and tomato :

lambda obs, act: -sqrt((obs[19] - obs[0])**2 + (obs[20] - obs[1])**2) # Distance between agent 1 and tomato

Ensure that your function uses the correct indices from the observation vector as described in the task description.

Prompt 3: VLM FEEDBACK PROMPT

You are an AI assistant helping to manage an Overcooked environment with multiple agents. The task is to prepare and deliver a {task_name}.
The environment is a 7x7 grid with various objects and {num_agents} agents.

Observation Space (32-length vector for each agent):
- Tomato: position (2), status (1)
- Lettuce: position (2), status (1)
- Onion: position (2), status (1)
- Plate 1: position (2)
- Plate 2: position (2)
- Knife 1: position (2)
- Knife 2: position (2)
- Delivery: position (2)
- Agent 1: position (2)
- Agent 2: position (2)
- Agent 3: position (2)
- Order: one-hot encoded (7)

MA-V1 Actions (index indicates macro action):
0: No operation
1: Move Up
2: Move Right
3: Move Down
4: Move Left
5: Interact (pick up, put down, chop)

You will be provided with a video of the agents' gameplay, which may be lengthy. Your task is to:
1. Identify and summarize the key actions and strategies employed by the agents throughout the gameplay.
2. Provide constructive feedback based on your observations in a single paragraph. Mark this paragraph with [SUGGESTION].

When generating feedback:
- Address specific agents by their color (e.g., Green agent, Rose agent) or position (e.g., agent on the left, agent near the cutting board).
- Focus on aspects of gameplay that could be significantly improved for any or all agents.
- Offer specific, actionable suggestions that can be immediately applied.
- Relate your feedback to the Overcooked environment, tasks, and overall efficiency.
- Prioritize improvements in teamwork, task allocation, or resource management.
- Consider how the suggestions could impact the agents' performance metrics.

Avoid: - Using overly technical jargon or complex explanations.
- Giving vague or general advice not specific to their gameplay.
- Mentioning anything outside the scope of the Overcooked game.
- Using excessive praise or encouragement.

Provide a brief summary of the agents' actions, followed by a single paragraph of feedback marked with [SUGGESTION], addressing the agents directly about their gameplay in the Overcooked environment. Focus on concrete

improvements for any or all agents rather than motivational language.

Algorithm 1 M^{3}HF: Multi-agent Reinforcement Learning from Multi-phase Human Feedback of Mixed Quality

Require: Number of agents N, Original Reward Functions $\{R_i^{\text{ori}}\}_{i=1}^N$, Predefined Reward Templates F, Environment E, Initial Policies $\{\pi^{i,0}\}_{i=1}^N$, Total Generations K

Ensure: Trained Policies $\{\pi^{i,K}\}_{i=1}^N$

Initialize Reward Function Pools $P_i = \{R_i^{\text{ori}}\}$ for each agent i
for generation $k = 0$ to $K-1$ **do**
Multi-agent Training Phase ▷ **Eq. 4**
  for each agent i **do**
    Train policy $\pi^{i,k}$ using current reward function $\hat{R}_i^k(s,a)$ (Eq. 13)
  end for
Rollout Generation ▷ **Sec. 4.1**
  if Periodic evaluation or performance stagnation detected **then**
    Generate rollout trajectories $\tau_k = \{(s_t, \mathbf{a}_t, r_t)\}_{t=0}^{H-1}$
  end if
Human Feedback Phase ▷ **Sec. 4.2**
  Human observes τ_k and provides feedback u_k
  Feedback Parsing:
  Use LLM $\mathcal{M}$ to parse u_k and assign feedback to agents:
  $u_k^i, u_k^{\text{all}} = \mathcal{M}(u_k, N)$
Reward Function Update ▷ **Sec. 4.3**
  for each agent i **do**
    Generate new reward function from feedback (Eq. 11):
    $R_{i,\text{new}} = \mathcal{M}(F, u_k^i, u_k^{\text{all}}, \boldsymbol{e})$
    Add $R_{i,\text{new}}$ to reward function pool P_i
  end for
  Weight Update:
  for each agent i **do**
    Initialize weight for new reward function:
    $w_{i,M} = \frac{1}{|P_i|}$
    Apply weight decay to existing weights (Eq. 16):
    $w_{i,m} = w_{i,m} \cdot \alpha^{M-m}, \forall m \in \{1, \ldots, M-1\}$
    Normalize weights:
    $w_{i,m} = \frac{w_{i,m}}{\sum_{j=1}^M w_{i,j}}, \forall m \in \{1, \ldots, M\}$
    Compute performance difference:
    $\Delta r_i = r_{i,k+1}^{\text{ori}} - r_{i,k}^{\text{ori}}$
    Adjust weight of newest reward function:
    $$w_{i,M} = \begin{cases} w_{i,M} + \beta, & \text{if } \Delta r_i > 0 \\ \max(0, w_{i,M} - \beta), & \text{otherwise} \end{cases}$$
    Update final reward function (Eq. 13):
    $\hat{R}_i^{k+1}(s,a) = \sum_{m=1}^M w_{i,m} \cdot R_{i,m}(s,a)$
  end for
end for