Towards Efficient Multi-Agent and Temporal Credit Assignment in Reinforcement Learning

Wen-Tse Chen

CMU-RI-TR-25-56

June 30, 2025

School of Computer Science
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Prof. Jeff Schneider, *chair*
Prof. Jean Oh,
Yufei Wang

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To my family, friends, and mentors, whose unwavering support and guidance made this journey possible.

Abstract

This thesis tackles the core challenge of credit assignment in reinforcement learning (RL), where agents must determine which actions or agents deserve credit for outcomes in complex environments. Traditional RL struggles when rewards are sparse, delayed, or shared among multiple agents, as is common in real-world tasks like robotics or game AI. To address this, the thesis introduces two innovative approaches: one for multi-agent collaboration and another for temporal decision-making.

The first contribution, ME-IGM, solves a critical problem in multi-agent RL. While existing methods like QMIX decompose global rewards into individual agent contributions, they often fail when combined with maximum entropy RL, which encourages exploration but can misalign local actions with the team's best strategy. ME-IGM fixes this by introducing an order-preserving transformation that ensures each agent's policy respects the global optimal action sequence. Experiments on the SMAC-v2 benchmark show ME-IGM outperforming prior methods, even matching imitation learning approaches without needing expert demonstrations.

The second contribution, RICOL, rethinks temporal credit assignment by leveraging large language models (LLMs) to analyze past decisions and assign rewards more efficiently. Instead of training a critic network from scratch, RICOL uses an LLM to retrospectively evaluate actions, converting sparse rewards into dense learning signals. This method is much more sample-efficient than traditional Monte Carlo estimation. RICOL builds on this by fine-tuning LLM policies through iterative updates, achieving dramatic improvements over PPO in tasks like BabyAI while remaining robust to noisy feedback.

Together, these advances make RL more practical for real-world applications by improving how agents learn from limited feedback. ME-IGM enables better teamwork in decentralized systems, while RICOL unlocks faster learning in sequential tasks. The results suggest promising directions for future work, such as adapting these methods to continuous control or integrating them with larger-scale AI systems. By refining credit assignment at both multi-agent and temporal levels, this research helps bridge the gap between theoretical RL and deployable solutions.

Acknowledgments

I would like to extend my heartfelt thanks to my advisor, Prof. Jeff Schneider, whose exceptional mentorship, unwavering support, and genuine kindness have profoundly shaped my journey as a researcher. His dedication to impactful research continues to inspire me deeply. I am forever grateful to my family, whose constant encouragement and belief in me have been my greatest strength throughout this path. My sincere appreciation also goes to Dr. Jiayu Chen, Dr. Zhongyu Li, Xintong Duan, Fahim Tajwar, Dr. Hao Zhu, Prof. Aviral Kumar, Minh Nguyen, Dr. Shiyu Huang, Rohit Sonder, Grace Liu, Alex Wu, Vedant Mundheda, Brian Yang, Dr. Yeeho Song, Zhuoming Chen and Silong Yong. Each of you has contributed immensely—with your insights, support, and encouragement—to both my academic growth and my personal development. Finally, I would like to thank my committee members, Prof. Jeff Schneider, Prof. Jean Oh, and Yufei Wang, for their time, thoughtful feedback, and generous guidance. This work would not have been possible without your help.

Funding

This work was supported in part by the U.S. Army Futures Command under Contract No. W519TC-23-C-0030.

Contents

1 Introduction **1**
1.1 Multi-Agent Credit Assignment 2
1.2 Temporal Credit Assignment 3

2 Multi-Agent Credit Assignment **5**
2.1 Introduction 5
2.2 Related Works 7
2.2.1 Multi-Agent Credit Assignment 7
2.2.2 Maximum Entropy MARL 8
2.3 Preliminary 9
2.3.1 Multi-Agent Reinforcement Learning 9
2.3.2 Individual-Global-Max Condition 9
2.4 Misalignment Between Local Policies and the Maximum Global Q-Value 10
2.5 Order-Preserving Transformation 12
2.5.1 Formulation of OPT 12
2.5.2 Training Scheme of OPT 13
2.6 The Overall Framework: ME-IGM 14
2.7 Experiments 16
2.7.1 Evaluation on Matrix Games 16
2.7.2 Evaluation on SMAC-v2 18
2.7.3 Ablation Study 19
2.8 Conclusion 22

3 Temporal Credit Assignment **25**
3.1 Introduction 25
3.2 Related Works 26
3.3 Preliminary 28
3.4 Retrospective In-Context Learning for Temporal Credit Assignment . 29
3.4.1 LLMs as Policies 29
3.4.2 Implementation Details 30
3.4.3 Retrospective In-Context Learning 31
3.5 Retrospective In-Context Online Learning 31
3.5.1 Policy Improvement based on RICL 32
3.5.2 Pipeline of RICOL 33

3.6 Experiments . . . 34
3.6.1 Environments . . . 34
3.6.2 Sample Efficiency of RICL in Credit Assignment (RQ1) . . . 35
3.6.3 RICL Enables Safer In-Context Updates via Retrospective Design (RQ2) . . . 36
3.6.4 Benchmarking RICOL (RQ3) . . . 37
3.6.5 RICOL Benefits from Credit Assignment (RQ3) . . . 40
3.6.6 RICOL is Robust to Noisy Verbal Feedback (RQ2) . . . 40
3.7 Conclusion . . . 41

4 Conclusion **43**

A Appendix **45**
A.1 TD(λ) . . . 45
A.2 Proof of Theorem 2.5.1 . . . 46
A.3 Hyper-network Structure . . . 50
A.4 Comparison with Baselines . . . 50
A.5 Hyper-parameters . . . 52
A.6 Derivation of the Loss Function . . . 58
A.7 Proof of Theorem 3.4.1 . . . 59
A.8 Comparison of Training Times . . . 61
A.9 Environments . . . 62
A.9.1 1D Key-Door Environment . . . 62
A.9.2 BabyAI Environment . . . 64
A.10 RICL can Identify Critical States in Sequential Decision-Making . . . 65
A.11 Comparisons between Retroformer and RICOL . . . 66

Bibliography **71**

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

2.1 The figure illustrates the improvement of our approach compared to existing maximum entropy MARL methods. The left figure shows a straightforward approach to applying maximum entropy MARL in the CTDE context, where blue texts represent the desired objectives, and black texts indicate corresponding constraints. It reveals that existing maximum entropy MARL methods under the CTDE framework implicitly constrain the global Q-value to be the sum of local Q-values, significantly limiting the expressiveness of the critic network. The right figure depicts the improvements in ME-IGM. ME-IGM first applies any credit assignment mechanism that satisfies the IGM condition to obtain Q_i for each agent i. Given the meaningful order of local Q-values (for different actions), an order-preserving transformation f_i converts Q_i to $f_i(Q_i)$ as the policy logits. This transformation is optimized using a loss function that minimizes the expected difference between $\sum_i f_i(Q_i)$ and Q_{tot}, which guarantees monotonic policy improvement in maximum entropy MARL. 6

2.2 **Illustration of Misalignment Between Local Policies and the Maximum Global Q-Value**: When naively combining the IGM condition with maximum entropy MARL, local policies often select suboptimal joint actions, leading to a lower Q-value, depicted by the blue curve. In contrast, the optimal joint actions achieve a higher Q-value, shown in the orange curve. Bridging the gap caused by misalignment presents an opportunity for improvement, as highlighted by the discrepancy between the two curves. 11

2.3 The overall pipeline of ME-IGM. 15

2.4 Comparison of exploration strategies in CTDE MARL: ME-QMIX with maximum entropy vs. QMIX with epsilon-greedy and extended annealing. 20

2.5 The impact of different hyperparameters for training α_ω on the performance of ME-QMIX. Results indicate that a lower learning rate for α_ω leads to increased exploration, slower convergence, and ultimately higher final performance. 22

2.6 The impact of different hyperparameters for training α_ω on the performance of ME-QMIX. Results indicate that ME-QMIX's performance is not sensitive to the choice of the target entropy $\bar{H}$ in Equation (2.10). 23

3.1 The pipeline of retrospective in-context online learning, where step ② and step ③ represent retrospective in-context learning. 32

3.2 Comparison of error in advantage function estimation. The x-axis represents the number of trajectories used for estimation, while the y-axis shows the mean error between the estimated advantage and the ground-truth advantages. RICL achieves significantly lower error with fewer samples (around 10) compared to Monte Carlo, which requires about 1000 samples for similar accuracy. Additionally, the error of RICL is more stable across trials (lower variance). 35

3.3 Accuracy comparison of ICL and RICL on predicting expert actions in the BabyAI *goto* scenario across 1000 trajectories. The Base bar shows the zero-shot performance of LLaMA-3.1-8B-Instruct. RICL outperforms ICL by 7.2%, demonstrating the effectiveness of retrospective updates. 37

3.4 Comparison of our method (RICOL) against four baseline algorithms across four BabyAI scenarios. RICOL consistently demonstrates superior sample efficiency, achieving strong performance with significantly fewer interactions. Notably, RICOL outperforms both PPO (10M) and PPO (3B), by over 50× and 10× fewer environment steps, respectively. Compared to Reflexion, an in-context learning method using trajectory-level verbal feedback, RICOL exhibits better convergent performance by leveraging temporal credit assignment (from RICL) and state-specific feedback. Additionally, RICOL surpasses GPT-4o mini, despite using a smaller policy model (LLaMA-3.2-3B-Instruct), underscoring the importance of interactive learning from the environment. **As a useful trick to boost performance, we use the real environment rewards as the advantage and apply advantage-weighted regression during the second stage of training, after RICOL completes its predefined training schedule in the first stage.** 38

3.5 RICOL employs in-context credit assignment to generate dense learning signals, enabling more sample-efficient policy training. In contrast, RWR lacks credit assignment, depends on strong base policies with high initial success rates, and performs poorly on tasks with sparse rewards. 40

3.6 The performance of RICOL under varying verbal feedback accuracy. The agent is trained with hand-crafted verbal feedback of different accuracy levels. Despite the presence of noise, RICOL maintains strong performance. Note that 50% accuracy corresponds to random feedback. 41

A.1 (a) The overall network architecture. Single Q-Net takes local observations as input and outputs a distribution over local actions u^i, also denoted as local Q_i. Functions OPTs are order-preserving transformations, ensuring the input and output dimensions are identical and have the same argmax values. Both the input and output of the OPTs are k-dimensional, where k is the action dimension. The output of the OPT, after undergoing a softmax operation, becomes the policy, from which action u_t^i is sampled. The Mixer network takes local Q_i as input and outputs the global Q_{tot}. Green blocks utilize a recurrent neural network as the backbone, with all agents sharing parameters. For decentralized execution, only the green network is needed, and the maximum values of their output are selected. (b) The blue blocks' network structure. The red blocks represent the hyper-network, which takes the global state s_t as input and outputs the network's weights and biases. The weights are constrained to be non-negative. The activation function is denoted by $\sigma(\cdot)$. The mixer network takes an n-dimensional Q function as input and outputs a one-dimensional Q_{tot}. 53
A.2 An illustration for the 1D Key-Door scenario. 62
A.3 The prompt used for the LLM policy in the 1D Key-Door scenario. . 63
A.4 The prompt used for the LLM reflector in the 1D Key-Door scenario. 63
A.5 Screenshots of four BabyAI scenarios, where the agent is partial-observable. 64
A.6 The prompt used for the LLM policy in all the BabyAI scenarios. . . 65
A.7 The prompt used for the LLM reflector in all the BabyAI scenarios. Only the **Feedback** part is extracted and used for RICL's in-context policy updates. 68
A.8 Comparison of critical states identified by RICL and verbal credit assignment. The x-axis represents a sequence of states: the agent first moves toward the key over 9 steps, picks up the key, then moves toward the door over another 9 steps, and finally opens the door. The y-axis indicates the score reflecting how critical each state is, where GT denotes the ground truth. The results show that both algorithms correctly identify the "pick up the key" action as a critical state-action pair. However, only RICL additionally identifies some of the earlier states in the "move to the key" phase as critical. 69

List of Tables

2.1 Payoff Matrix 18

2.2 Result of ME-QMIX (Ours) 18

2.3 Result of FOP 18

2.4 Result of QMIX 18

2.5 (a) presents the payoff matrix for a one-step matrix game involving two agents: the row player and the column player. Each agent has three possible actions: {A, B, C}. The rewards for each joint action are given in a 3×3 matrix. This payoff structure is non-monotonic, as the optimal action for one player depends on the action chosen by its teammate. (b)–(d) illustrate the results of ME-QMIX, FOP, and QMIX, respectively. The first row and first column display the policies adopted by the row and column players after $10k$ training steps. The 3×3 matrix in the bottom right corner represents the global Q-function Q_{tot} learned by each algorithm. We can see that ME-QMIX is the only method that assigns the highest probability to selecting the optimal joint action, highlighting its effectiveness in overcoming the misalignment issue and achieving optimal coordination. Note that the ϵ in (d) denotes the exploration rate used in ϵ-greedy exploration. 18

2.6 Average win rates on **SMAC-v2 (Protoss)** tasks. The results demonstrate that our method, ME-IGM, outperforms all previous baseline algorithms and achieves performance comparable to IL+MARL methods, despite not having access to expert opponent demonstrations or opponent modeling. 19

2.7 Average win rates on **SMAC-v2 (Terran)** tasks. 20

2.8 Average win rates on **SMAC-v2 (Zerg)** tasks. 21

2.9 Comparison between our algorithm (+OPT) and its two ablated versions: +Entropy, obtained by removing OPTs, and QMIX, obtained by further removing softmax local policies. 21

2.10 Comparison between ME-QMIX (OPT) and its ablated version, ME-QMIX (MLP), where the OPT is replaced with a three-layer MLP. . 22

3.1 We propose RICL for LLMs to learn from environmental feedback, fundamentally different from intrinsic self-correction work like SELF-REFINE [35]. Unlike STaR [61] and Reflexion [50], our method utilizes environmental feedback to retrospectively update the policy. Additionally, we do not fine-tune an extra reflector, as done in RLMEC [10] and Retroformer [58]. By generating dense training signals from sparse environmental feedback, our approach is more sample-efficient than methods that rely solely on sparse preference rewards, such as Iterative RPO [39]. 28

A.1 Hyperparameters used for hyper-networks. 50
A.2 HASAC hyperparameters used for SMACv2. We use the hyperparameters for SMAC as specified in the original paper [32]. 54
A.3 ME-QMIX hyperparameters used for SMACv2. We utilize the hyperparameters used in SMACv2 [14]. 55
A.4 ME-QPLEX hyperparameters used for SMACv2. We utilize the hyperparameters used in SMACv2 [14]. 56
A.5 FOP hyper-parameters used for SMACv2. We use the hyperparameters for SMAC as specified in the original paper [63]. 57
A.6 Training time (in minutes) required for each algorithm on different tasks, along with the GPU setup used. 61
A.7 Approximation errors vs. number of sample trajectories. 67

Chapter 1

Introduction

Recent advances have shown that scaling up supervised learning can lead to human-level performance across a wide range of domains, including mathematics, coding, and image generation [1, 26]. However, when the goal is to achieve superhuman performance, supervised learning becomes increasingly inefficient [55]. A key limitation is that many real-world tasks lack accessible expert demonstrations. In such cases, the available supervision signals are often sparse or limited, making it difficult for supervised learning methods to effectively learn and generalize from them.

For example, consider a real-world software development scenario in which a team of engineers collaborates to build a complex system. The task is highly challenging and requires the team to take a long sequence of coordinated actions. Crucially, feedback is sparse and typically only available upon successful completion of the project. In such settings, the reward is not only delayed and sparse, but also shared among all team members. This makes it especially difficult to assess the contribution of each individual agent or action, as the final outcome does not directly reveal the impact of any single decision or participant.

This example highlights a fundamental and often underexplored challenge in reinforcement learning (RL): credit assignment. When a reward is shared across multiple time steps or among multiple agents, how can credit be accurately attributed to each individual action or agent based on their contribution to the final outcome? Designing effective credit assignment mechanisms is crucial for improving the training efficiency of RL algorithms. More importantly, credit assignment enables RL agents

to learn from weakly structured or sparse feedback signals—far beyond the fully labeled data typically required by supervised learning. As a result, credit assignment lies at the heart of what makes RL fundamentally different from supervised learning, and it is a key reason why RL holds the potential to achieve superhuman intelligence in domains where supervised learning alone falls short.

In this work, we analyze two key aspects of credit assignment: multi-agent credit assignment and temporal credit assignment. Each will be introduced in the following sections.

1.1 Multi-Agent Credit Assignment

In previous research, various approaches such as monotonic value decomposition [46] and counterfactual reasoning [16] have proven effective in addressing the credit assignment problem within multi-agent collaboration tasks. However, these methods predominantly focus on Q-learning or naive policy gradient methods, leaving a notable gap in the exploration of credit assignment solutions tailored for maximum entropy RL. This challenge arises because maximum entropy RL limits the expressive capabilities of multi-agent RL algorithm, often conflicting with the objectives of traditional credit assignment algorithms.

In Chapter 2, we begin by applying monotonic value decomposition to assign credit among agents. Local Q-values are obtained from the global Q-values via this decomposition and are then used to train local policies. However, we identify a critical limitation in this approach: a misalignment between the global Q-values and the local policies during inference. Specifically, the actions selected by local policies may not correspond to those that yield the highest global Q-values. This misalignment leads to suboptimal performance, as the difference between the value of the selected global Q-value and the optimal one represents a potential performance gain that is left unrealized. Empirically, we demonstrate that this gap can reach up to a twofold difference on SMAC-v2 [14] tasks.

To address this issue, we propose an order-preserving transformation that converts local Q-values into local policies while maintaining the relative action preferences implied by the global Q-values. This ensures that agents can consistently select actions corresponding to the highest global Q-values, even when executing in a

decentralized manner. Building on this idea, we introduce ME-IGM, an algorithm that enhances multi-agent credit assignment in maximum entropy reinforcement learning frameworks, improving coordination and performance across agents. We demonstrate the effectiveness of our algorithm on fifteen SMAC-v2 scenarios, where the proposed ME-IGM consistently outperforms baseline MARL algorithms.

1.2 Temporal Credit Assignment

Temporal credit assignment has been extensively studied in classical RL, typically addressed by learning a critic network. However, such networks are often domain-specific and sample inefficient to train from scratch. In contrast, large language models (LLMs) have demonstrated strong in-context learning capabilities for acquiring new knowledge efficiently.

In Chapter 3, we introduce a method that leverages in-context learning to perform temporal credit assignment. Instead of learning a critic network, we use the LLM's own reasoning ability to retrospectively attribute rewards to prior states along the trajectory based on their contributions to the outcome, a process we call retrospective in-context learning. This approach is sample efficient, as it avoids training a domain-specific value function.

We further integrate this approach with a policy improvement step using Advantage-Weighted Regression (AWR). Similar to standard online RL, the resulting algorithm, Retrospective In-Context Online Learning, alternates between policy improvement and policy evaluation. However, instead of learning a value function, the policy evaluation step is replaced with in-context learning to improve the sample efficiency of the online learning process. Experiments on four BabyAI tasks demonstrate that our method is 20 to 50 times more sample efficient than PPO, which must learn the value function from scratch.

In summary, this work introduces two complementary approaches to the credit assignment problem in RL, multi-agent credit assignment and temporal credit assignment. Together, these approaches highlight the importance of credit assignment as a foundational component in online RL, especially in real-world scenarios characterized by sparse, delayed, or weak feedback. By tackling both multi-agent and temporal dimensions of credit assignment, our work paves the way for the design of more

robust, efficient, and general-purpose online RL algorithms that can operate beyond the limitations of supervised learning.

Chapter 2

Multi-Agent Credit Assignment

2.1 Introduction

Collaborative multi-agent tasks, where a team of agents works together to complete a task and receives joint rewards, are inherently challenging for RL agents. This difficulty arises due to the absence of individual reward signals for each agent and the significantly higher exploration requirements compared to single-agent scenarios. The challenge is further exacerbated by the exponential growth of the joint action space as the number of agents increases. In this chapter, we propose leveraging maximum entropy RL to promote exploration and adopting the individual-global-max (IGM) credit assignment mechanism [46] to effectively distribute joint rewards among the agents.

Maximum entropy RL [29], recognized for promoting exploration [20], smoothing the objective landscape [4], and improving robustness [15], has been extensively applied in single-agent RL. However, as illustrated in Figure 2.1, naively extending maximum entropy RL to multi-agent settings under the centralized training and decentralized execution (CTDE) paradigm leads to uniform credit assignment, which limits the ability of multi-agent reinforcement learning (MARL) agents to effectively learn from joint rewards.

On the other hand, numerous studies have explored the development of effective credit assignment mechanisms. Among them, the IGM condition has demonstrated strong empirical performance by aligning the maximum local Q-values with the

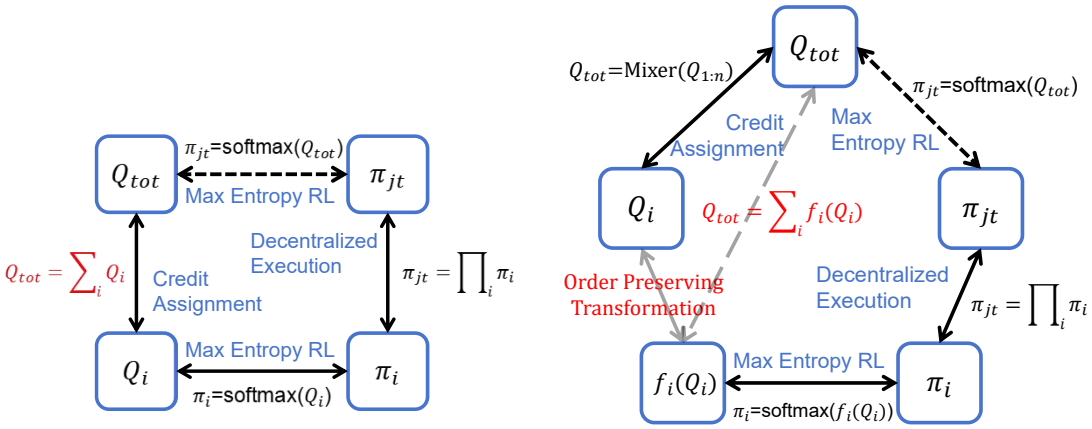


Figure 2.1: The figure illustrates the improvement of our approach compared to existing maximum entropy MARL methods. The left figure shows a straightforward approach to applying maximum entropy MARL in the CTDE context, where blue texts represent the desired objectives, and black texts indicate corresponding constraints. It reveals that existing maximum entropy MARL methods under the CTDE framework implicitly constrain the global Q-value to be the sum of local Q-values, significantly limiting the expressiveness of the critic network. The right figure depicts the improvements in ME-IGM. ME-IGM first applies any credit assignment mechanism that satisfies the IGM condition to obtain Q_i for each agent i. Given the meaningful order of local Q-values (for different actions), an order-preserving transformation f_i converts Q_i to $f_i(Q_i)$ as the policy logits. This transformation is optimized using a loss function that minimizes the expected difference between $\sum_i f_i(Q_i)$ and Q_{tot}, which guarantees monotonic policy improvement in maximum entropy MARL.

maximum global Q-value. Previous works have primarily applied the IGM condition in deterministic policy settings, where policies are derived using the argmax of local Q-values. However, in maximum entropy MARL methods, stochastic local policies are learned, and misalignment can occur between local policies and the maximum global Q-values. That is, selecting joint actions based on the highest logits of local policies does not necessarily lead to maximizing the global Q-value. This limitation highlights a critical challenge in achieving effective credit assignment within the maximum entropy MARL framework.

We propose ME-IGM, the first CTDE maximum entropy MARL method that is compatible with all credit assignment schemes satisfying the IGM condition. In

particular, to address the misalignment between stochastic local policies and the maximum global Q-values, we introduce an order-preserving transformation (OPT) that maps local Q-values to policy logits while preserving their relative order. This ensures that selecting actions with the highest logits collectively leads to a joint action that maximizes the global Q-value. Additionally, the OPT is trained with a theoretically grounded objective, guaranteeing monotonic policy improvement in CTDE MARL. Moreover, compared with value decomposition MARL methods such as QMIX, ME-IGM incorporates maximum entropy RL, using stochastic policies and entropy regularization to enhance exploration and robustness.

Our main contributions are summarized as follows:

- We are the first to identify and empirically demonstrate a critical issue of existing maximum entropy MARL methods: the misalignment between stochastic local policies and the maximum global Q-values.
- We explore Individual-Global-Max in maximum entropy MARL to resolve such misalignment, and introduce ME-IGM, the first CTDE maximum entropy MARL method that fully complies with the IGM condition.
- The core of ME-IGM is an order-preserving transformation (OPT), which can be seamlessly integrated into mainstream value decomposition MARL methods for a maximum entropy extension, enhancing exploration. Additionally, we propose a theoretically grounded and straightforward objective for training these transformation operators.
- We confirm the effectiveness of our algorithm through tests on matrix games and SMAC-v2 [14], where our method attains state-of-the-art results.

2.2 Related Works

2.2.1 Multi-Agent Credit Assignment

In the CTDE framework, MARL primarily consists of two algorithmic branches: policy gradient methods [16, 27, 31, 33, 59, 65] and value function decomposition methods [46, 53, 56]. Policy gradient methods first learn a centralized critic network, and then distill local policies using losses like the KL divergence between local

policies and the softmax of the global Q-value. On the other hand, value function decomposition addresses the credit assignment problem by decomposing the global value function into multiple local value functions. VDN [53] assumes that the global Q-function is the sum of local Q-functions. QMIX [40, 46] permits the mixer function to be any function with non-negative weights. QTRAN [51] optimizes an additional pair of inequality constraints to construct a loss function. QPLEX [56] uses the dueling architecture to decompose the Q-function, achieving the same expressive power as QTRAN while being easier to optimize. The expressive power of the critic networks in these algorithms increases sequentially. Our method, ME-IGM, enhances the exploration capability of these MARL algorithms by leveraging the maximum entropy RL framework and is compatible with all types of multi-agent value decomposition methods. Specifically, we demonstrate the performance of ME-QMIX and ME-QPLEX in the experiment section.

2.2.2 Maximum Entropy MARL

Maximum entropy RL [29] is demonstrated to have advantages such as encouraging exploration [20] and increasing robustness [15] in single-agent RL scenarios. In maximum entropy MARL, most works utilize an actor-critic architecture and typically aim to minimize the KL divergence between the local actor and the softmax of the centralized critic [18, 21, 45, 63]. However, although these methods use QMIX to train the critic, they optimize the policy function by minimizing the expected KL divergence over the state space. As a result, they cannot guarantee that the joint action (at each state) determined by local policies maximizes the global Q-value (i.e., the IGM condition), which is fundamental to QMIX. In this case, we propose a novel order preserving transformation to ensure that the IGM condition is met while employing maximum entropy MARL. Moreover, our method introduces global information into local policies during training through a hyper-network [19], which most previous methods either ignored [62] or only applied in centralized training and centralized execution scenarios [32]. Note that the global information is not required for the local policies during execution. In appendix A.4, we provide a more detailed comparison of our work with related works.

2.3 Preliminary

2.3.1 Multi-Agent Reinforcement Learning

Cooperative MARL can be formalized as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [37]. Formally, a Dec-POMDP is represented as a tuple $(\mathcal{A}, S, U, T, r, O, G, \gamma)$, where $\mathcal{A} \equiv \{1, ..., n\}$ is the set of n agents. S, U, O, and γ are state space, action space, observation space, and discount factor, respectively.

During each discrete time step t, each agent $i \in \mathcal{A}$ chooses an action $u^i \in U$, resulting in a collective joint action $\mathbf{u} \in \mathbf{U} \equiv U^n$. The function $r(s, \mathbf{u})$ defines the immediate reward for all agents when the collective action $\mathbf{u}$ is taken in the state s. $\mathcal{P}(s, \mathbf{u}, s') : S \times \mathbf{U} \times S \to [0, \infty)$ is the state-transition function, which defines the probability density of the succeeding state s' after taking action $\mathbf{u}$ in state s. In a Dec-POMDP, each agent receives only partial observations $o \in O$ according to the observation function $G(s, i) : S \times \mathcal{A} \to O$. For simplicity, if a function depends on both o^i and s, we will disregard o^i. Each agent uses a policy $\pi_i(u^i|o^i)$ to produce its action u^i. Note that the policy should be conditioned on the observation history $o^i_{1:t}$. For simplicity, we refer to $o^i_{1:t}$ as o^i_t or o^i in the whole paper. The joint policy is denoted as π_{jt}. $\rho_\pi(s_t, \mathbf{u_t})$ is used to represent the state-action marginals of the trajectory distribution induced by a policy π.

2.3.2 Individual-Global-Max Condition

Individual-Global-Max (IGM) [46] is a commonly used credit assignment method in value-based cooperative MARL and is defined as follows:

Definition 2.3.1. Individual-Global-Max (IGM) For joint q-function Q_{tot}, if there exist individual q-functions $[Q_i]_{i=1}^n$ such that the following holds:

$$\underset{\mathbf{u_t}}{\operatorname{argmax}}\ Q_{tot}(\mathbf{u_t}, s_t) = \begin{pmatrix} \underset{u_t^1}{\operatorname{argmax}}\ Q_1(u_t^1|o_t^1) \\ \vdots \\ \underset{u_t^n}{\operatorname{argmax}}\ Q_n(u_t^n|o_t^n) \end{pmatrix}. \tag{2.1}$$

Then, we say that $[Q_i]_{i=1}^n$ satisfy **IGM** for Q_{tot} under s_t.

For instance, in QMIX, to ensure the validity of IGM, the algorithm restricts the expressive capability of Q_{tot} to the following form: $\frac{\partial Q_{tot}}{\partial Q_i} \geq 0, \forall i \in \mathcal{A}$. Thus, Q_{tot} is a monotonic function. The monotonicity constraint serves as a sufficient, but not necessary, condition for IGM, and can be relaxed as a tradeoff to enhance expressiveness.

2.4 Misalignment Between Local Policies and the Maximum Global Q-Value

In this section, we first introduce the problem of misalignment between local policies and the maximum global Q-value. We then empirically demonstrate that this issue is prevalent in most prior maximum entropy MARL works applying the IGM condition.

The Misalignment Problem: We take QMIX as an example, but the concept discussed here applies to any credit assignment mechanism satisfying the IGM condition. While the IGM condition defines the relationship between local Q-values and the global Q-value, what is often more important in practice is the alignment between local policies and the global Q-value. In other words, joint actions that maximize the global Q-value can be obtained by querying local policies. In prior work leveraging the IGM principle with deterministic policies, this alignment naturally holds because each local policy selects the $argmax$ of its local Q-value, ensuring that the joint action aligns with the global Q-value. However, this alignment breaks under stochastic policies, leading to what we term as misalignment between local policies and the maximum global Q-value.

Previous work in maximum entropy MARL, such as Guo and Wu [18], Zhang et al. [63], often decompose the global Q-value into individual local Q-values to satisfy the IGM condition. These local Q-values are then used to guide the training of local stochastic policies. Compared with value decomposition methods such as QMIX, they train stochastic policy networks in addition to the Q-functions, which can potentially improve multi-agent exploration in complex environments. However, while the value decomposition ensures alignment between global and local Q-values via the IGM condition, there is typically no explicit constraints imposed between the local policies and the local Q-values during the policy training phase. **This lack of coupling**

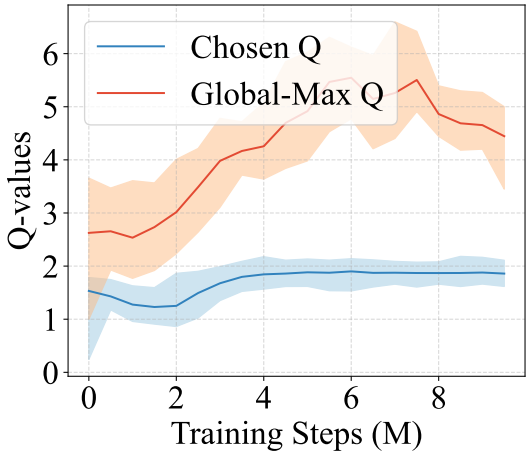


Figure 2.2: **Illustration of Misalignment Between Local Policies and the Maximum Global Q-Value**: When naively combining the IGM condition with maximum entropy MARL, local policies often select suboptimal joint actions, leading to a lower Q-value, depicted by the blue curve. In contrast, the optimal joint actions achieve a higher Q-value, shown in the orange curve. Bridging the gap caused by misalignment presents an opportunity for improvement, as highlighted by the discrepancy between the two curves.

often leads to misalignment during testing, where the local policies fail to consistently select joint actions that maximize the global Q-value, defeating the purpose of enforcing the IGM condition. Consequently, the learned policies may struggle to achieve optimal global coordination during execution.

Empirical Demonstration: To analyze the performance gap caused by the misalignment between local policies and the maximum global Q-value, we evaluate the difference between the maximum global Q-value and the global Q-value selected by local policies in the SMAC-v2 *zerg 5vs5* scenario, using a state-of-the-art Maximum Entropy MARL algorithm [63]. Figure 2.2 shows the results, averaged over five random seeds with standard deviations included. The findings reveal a significant Q-value gap, emphasizing the potential for improvement by developing algorithms that enable the selection of joint actions with the highest global Q-value through local stochastic policies. For evaluation purposes, the maximum global Q-value is calculated by enumerating all possible joint actions, which increases computational cost by 160k times. Yet, this exhaustive approach is used exclusively for evaluation

and does not impact training.

2.5 Order-Preserving Transformation

Here, we propose a novel order-preserving transformation (OPT) to solve the aforementioned misalignment problem between local policies and the maximum global Q-value.

2.5.1 Formulation of OPT

The order-preserving transformation is used to map local Q-values to the logits of local policies while preserving their relative order. According to the IGM condition, this approach ensures that, during testing, local policies – by selecting actions with the highest logits – will consistently identify the joint action that corresponds to the highest global Q-value among all possible joint actions. Formally, the OPT is defined as follows:

$$\mathrm{OPT}(x, s) = W^2\sigma(W^1x + b^1) + b^2 \tag{2.2}$$

Here, $W^1_{(d_1\times 1)}$, $W^2_{(1\times d_1)}$, $b^1_{(d_1\times 1)}$, and $b^2_{(1\times 1)}$ are generated by a hyper-network[1] which takes the global state s as input. Consider x as a local Q-value of dimension $(1 \times d_2)$. Each entry in x is multiplied by the same weight entry in W^1 and then shifted by the same bias entry in b^1, resulting in d_1 hidden variables of dimension $(d_1 \times d_2)$, computed as $W^1x + b^1$. Notably, when all entries in W^1 are non-negative, the relative order of elements in x is preserved in each hidden variable. Next, a nonlinear activation function σ, such as ELU [12], is applied, followed by another order-preserving linear transformation using W^2 and b^2. Consequently, the final output, $\mathrm{OPT}(x, s)$, maintains both the dimension and the relative order of the entries in the input x.

In this chapter, $\mathrm{OPT}(x, s)$ is used as the logits of the local policy corresponding to the local Q-value x. According to the IGM condition, maximum points of the local Q-values align with the maximum point of the global Q-value (i.e., the optimal joint action). Then, through our OPT design, each agent can select its action corresponding to the highest policy logit, collectively forming the optimal joint action. Thus, the

[1]For details of the hyper-network, please refer to Appendix A.3.

proposed OPT addresses the aforementioned misalignment issue commonly found in maximum entropy MARL.

2.5.2 Training Scheme of OPT

In this subsection, we first formulate the ideal objective of Maximum Entropy MARL and its practical counterpart, which admits an analytical-form solution. We then analyze the gap between these two objective functions, providing a theoretically sound and straightforward objective design for Maximum Entropy MARL based on OPTs.

Maximum Entropy MARL trains the individual and global Q-functions as in value decomposition MARL methods, while training a policy network in a Maximum Entropy RL framework based on the global Q-values. Formally, the joint policy is trained by the following equation:

$$\min_{\pi'_{jt}\in\Pi} D_{KL}\left(\pi'_{jt}(\cdot|s_t)\middle\|\frac{\exp(Q_{tot}^{\pi_{\text{old}}}(s_t,\cdot)/\alpha)}{Z^{\pi_{\text{old}}}(s_t)}\right), \tag{2.3}$$

where $Z^{\pi_{\text{old}}}(s_t)=\sum_{\mathbf{u}}\exp(Q_{tot}^{\pi_{\text{old}}}(s_t,\mathbf{u})/\alpha)$ and α is the temperature in the softmax distribution.

To simplify and decentralize the policy improvement process, we propose replacing $Q_{tot}^{\pi_{\text{old}}}(s_t,\cdot)$ with $\sum_i \text{OPT}_i(Q_i^{\pi_{\text{old}}}(o_t^i,\cdot),s_t)$, leading to the following objective function:

$$\min_{\pi'_{jt}\in\Pi} D_{KL}\left(\pi'_{jt}(\cdot|s_t)\middle\|\frac{\exp(\sum_i \frac{\text{OPT}_i(Q_i^{\pi_{\text{old}}}(o_t^i,\cdot),s_t)}{\alpha})}{Z'^{\pi_{\text{old}}}(s_t)}\right), \tag{2.4}$$

where $Z'^{\pi_{\text{old}}}(s_t)$ is the normalization factor. The joint policy π'_{jt} is implemented as a set of independent local policies for decentralized execution. Thus, we have the analytical solution for the local policy of agent i as:

$$\pi_i^{\text{new}}(a|o_t^i)\propto\exp(\text{OPT}_i(Q_i^{\pi_{\text{old}}}(o_t^i,a))/\alpha). \tag{2.5}$$

We can see that the policy is defined based on the local Q-value and utilizes the OPT to ensure that the relative order of policy logits aligns with the local Q-values, as previously introduced.

However, there is a gap between $\sum_i \text{OPT}_i(Q_i^{\pi_{\text{old}}}(o_t^i,\cdot),s_t)$ and $Q_{tot}^{\pi_{\text{old}}}(s_t,\cdot)$. Also,

we find that the policy improvement using Equation (2.4) no longer guarantees a monotonic increase in the global Q-value, unlike the approach based on Equation (2.3). Specifically, we have the following theorem:

Theorem 2.5.1. *Denote the policy before the policy improvement as π_{old} and the policy achieving the optimality of the improvement step as π_{new}. Updating the policy with Equation (2.3) ensures monotonic improvement in the global Q-value. That is, $Q_{tot}^{\pi_{old}}(s_t, \mathbf{u_t}) \leq Q_{tot}^{\pi_{new}}(s_t, \mathbf{u_t}), \forall s_t, \mathbf{u_t}$. In contrast, updating the policy with Equation (2.4) only ensures that $Q_{tot}^{\pi_{old}}(s_t, \mathbf{u_t}) - \epsilon \leq Q_{tot}^{\pi_{new}}(s_t, \mathbf{u_t}), \forall s_t, \mathbf{u_t}$. The gap ϵ satisfies:*

$$\epsilon \leq \frac{\gamma}{1-\gamma}\mathbb{E}_{s_{t+1}\sim\mathcal{P}}\left[C\sqrt{2D_{KL}\left(\pi_{old}(\cdot|s_{t+1})|\pi_{new}(\cdot|s_{t+1})\right)}\right] \tag{2.6}$$

where γ is the discount factor, $\mathcal{P}(s_t, \mathbf{u_t}, \cdot)$ is the transition function and $C = \max_{\mathbf{u}_{t+1}} \Delta_Q(s_{t+1}, \mathbf{u}_{t+1}) = \max_{\mathbf{u}_{t+1}} |Q_{tot}^{\pi_{old}}(s_{t+1}, \mathbf{u}_{t+1}) - \sum_i OPT_i(Q_i^{\pi_{old}}(o_{t+1}^i, u_{t+1}^i))|$.

Proof. See Appendix A.2.

This theorem implies that we must enforce $\epsilon \leq 0$ to guarantee monotonic policy improvement when using Equation (2.4) as the objective. This motivates us to minimize the factor C and adopt the following objective to train the OPTs.

$$\min_{\mathrm{OPT}_{1:n}} \mathbb{E}_{s,\mathbf{u}\sim\rho_{\pi_{jt}^{new}}}[\Delta_Q(s_t, \mathbf{u}_t)^2], \tag{2.7}$$

where $\pi_{jt}^{new} = (\pi_1^{new}, \cdots, \pi_n^{new})$ and π_i^{new} is defined in Equation (2.5).

2.6 The Overall Framework: ME-IGM

Figure 2.3 illustrates the framework of ME-IGM, which consists of three types of networks. Agents 1 to n share a network f_θ for predicting local Q-values. The Mixer network, which aggregates local Q-values and manages the credit assignment among agents, is parameterized through its associated hyper-network as Mixer_θ[2]. $[\mathrm{OPT}_{i,\phi}]_{i=1}^n$ are n order-preserving transformations, and $\mathrm{OPT}_{i,\phi}(f_\theta(o_t^i), s_t)$ defines a local stochastic policy for agent i. The update rules for these networks are described

[2]Our algorithm is compatible with any Mixer network design that satisfies the IGM condition. For instance, during evaluation, we utilize the Mixer networks of QMIX and QPLEX, leading to two variants of our algorithm: ME-QMIX and ME-QPLEX.

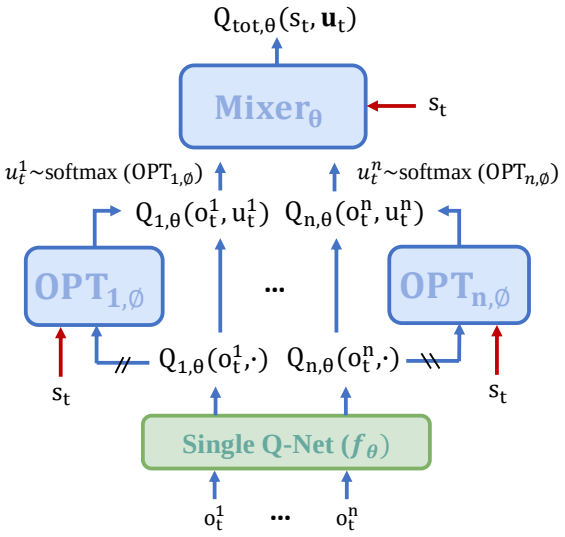


Figure 2.3: The overall pipeline of ME-IGM.

as follows.

ME-IGM optimizes two loss functions to update θ and ϕ separately. In particular, Mixer_θ and f_θ are trained by minimizing $L(\theta)$:

$$\mathbb{E}_{\tau_t}\left[\frac{1}{2}\left(Q_{tot_\theta}(s_t, \mathbf{u}_t) - \widehat{\mathcal{T}}^{\pi_{jt}} Q_{tot}(s_t, \mathbf{u}_t)\right)^2\right] \tag{2.8}$$

where $\tau_t = \{o_{t:t+1}^{1:n}, u_{t:t+1}^{1:n}, s_{t:t+1}, r_t\}$, $Q_{tot_\theta}(\mathbf{u}, s) = \text{Mixer}_\theta\big(f_\theta(o^1, u^1), \ldots, f_\theta(o^n, u^n), s\big)$, and the estimated Q-target $\widehat{\mathcal{T}}^{\pi_{jt}} Q_{tot}(s_t, \mathbf{u}_t) = r_t + \gamma Q_{tot,\theta^-}(s_{t+1}, \mathbf{u}_{t+1})$. As a common practice, we utilize a target Q network Q_{tot,θ^-} to stabilize training, with θ^- being the exponentially weighted moving average of θ. Notably, local actions $u_{t:t+1}^{1:n}$ are sampled from local stochastic policies π_i (as defined in Equation (2.10)) during roll-out, rather than selecting the argmax of local Q-functions. This approach offers two advantages: (1) the softmax policy function enhances exploration, and (2) incorporating state s in the sampling process allows for better utilization of global information during centralized training.

Further, to train $[\text{OPT}_{i,\phi}]_{i=1}^n$, we adopt a sample-based version of Equation (2.7)

introduced in section 2.5.2, denoted as $L(\phi)$:

$$\mathbb{E}_{\tau_t}\left[\left(\sum_{i=1}^{n}\mathrm{OPT}_{i,\phi}(f_\theta(o_t^i,u_t^i),s_t)-Q_{tot,\theta}(s_t,\mathbf{u}_t)\right)^2\right] \tag{2.9}$$

The OPTs define the local policies for each agent as follows:

$$\pi_i(u^i|o^i,s)=\mathrm{softmax}\left(\frac{\mathrm{OPT}_{i,\phi}\left(f_\theta(o^i),s\right)}{\alpha_\omega}\right) \tag{2.10}$$

α_ω is the temperature and, similar to Haarnoja et al. [20], can be updated using the following loss function:

$$L(\omega)=\mathbb{E}_{\tau_t}\left[-\alpha_\omega\log\pi_{jt}(\mathbf{u}_t|s_t)-\alpha_\omega\bar{\mathcal{H}}\right], \tag{2.11}$$

where $\bar{\mathcal{H}}$ is a predefined target entropy. Note that the local policies (i.e., π_i) are only used to collect training roll-outs τ_t. During execution, we would select the argmax of f_θ, which does not require any centralized information (e.g., s), since OPTs preserve the order of the input vectors. **For the pseudo code of ME-IGM, please refer to the Algorithm 1.**

2.7 Experiments

In this section, we first evaluate our method on a classic matrix game, demonstrating its capability to learn a globally optimal policy despite the non-monotonic nature of the overall payoff matrix. Next, we compare our algorithm against a set of baselines on the SMAC-v2 benchmark, highlighting its state-of-the-art performance in cooperative MARL. Finally, we conduct ablation studies to evaluate the benefits of entropy-driven exploration, assess the necessity of incorporating the order-preserving transformation, and analyze the impact of key hyperparameters.

2.7.1 Evaluation on Matrix Games

We first demonstrate the effectiveness of our algorithm in a classic one-step matrix game, shown as Table 2.5. It is a non-monotonic payoff matrix commonly used in

Algorithm 1 ME-IGM

Input: θ, ϕ
Initialize: $\theta^- \leftarrow \theta, D \leftarrow \emptyset$
for each iteration **do**
    for each environment step **do**
        get $\pi_i(u_t^i|o_t^i, s_t)$ via Eq. 2.10
        $u_t^i \sim \pi_i(u_t^i|o_t^i, s_t), \forall i \in \{1, \ldots, n\}$
        $s_{t+1} \sim \mathcal{P}(s_t, \mathbf{u}_t, \cdot)$
        $D \leftarrow D \cup \{(s_t, \mathbf{u}_t, r(s_t, \mathbf{u}_t), s_{t+1})\}$
    end for
    for each gradient step **do**
        calculate $\mathcal{T}_\lambda^{\pi_{jt}} Q_{tot}$ via Eq. A.1
        $\theta \leftarrow \theta - \lambda_\theta \nabla_\theta L(\theta)$
        $\phi \leftarrow \phi - \lambda_\phi \nabla_\phi L(\phi)$
        $\omega \leftarrow \omega - \lambda_\omega \nabla_\omega L(\omega)$
        $\theta^- \leftarrow \tau\theta + (1-\tau)\theta^-$
    end for
end for

previous studies [51, 56] to evaluate the representational capacity of the Q networks. We evaluate ME-QMIX, a variant of ME-IGM built upon QMIX, by comparing its performance against two state-of-the-art MARL methods: QMIX [46], a value decomposition MARL method, and FOP [63], a Maximum Entropy MARL method, on the matrix game.

In particular, we are concerned with whether the joint actions with the highest selection probability correspond to the globally maximal value function, ensuring optimal returns when executing a deterministic policy. For suboptimal joint actions, we also wish to visit them with a certain probability to ensure adequate exploration.

As shown in Table 2.5, QMIX suffers from high estimation error in payoff values due to its monotonicity constraints. Additionally, its deterministic policy lacks effective exploration, causing it to converge to suboptimal joint actions (i.e., (B, B)) with a reward of 0. FOP, which employs a network architecture similar to QPLEX, is capable of accurately estimating value functions (i.e., the payoff). However, its policy tends to select suboptimal joint actions more frequently due to the misalignment between local policies and the maximum global Q-values, as discussed in Section 2.4. ME-QMIX exhibits smaller estimation errors for the payoff of optimal joint

Table 2.1: Payoff Matrix

	A	B	C
A	**8**	-12	-12
B	-12	0	0
C	-12	0	0

Table 2.2: Result of ME-QMIX (Ours)

	1.	0.	0.
1.	**8**	-12	-14
0.	-8	-28	-30
0.	-10	-30	-32

Table 2.3: Result of FOP

	.21	.39	.40
.21	**8**	-12	-12
.40	-12	0	0
.39	-12	0	0

Table 2.4: Result of QMIX

	ϵ	1-2ϵ	ϵ
ϵ	-8.5	-8.5	-8.5
1-2ϵ	-8.5	**0.2**	0.2
ϵ	-8.5	0.2	0.2

Table 2.5: (a) presents the payoff matrix for a one-step matrix game involving two agents: the row player and the column player. Each agent has three possible actions: {A, B, C}. The rewards for each joint action are given in a 3×3 matrix. This payoff structure is non-monotonic, as the optimal action for one player depends on the action chosen by its teammate. (b)–(d) illustrate the results of ME-QMIX, FOP, and QMIX, respectively. The first row and first column display the policies adopted by the row and column players after $10k$ training steps. The 3×3 matrix in the bottom right corner represents the global Q-function Q_{tot} learned by each algorithm. We can see that ME-QMIX is the only method that assigns the highest probability to selecting the optimal joint action, highlighting its effectiveness in overcoming the misalignment issue and achieving optimal coordination. Note that the ϵ in (d) denotes the exploration rate used in ϵ-greedy exploration.

actions (which is our primary concern) while having larger errors for suboptimal ones. Crucially, the joint action with the maximum Q-value in ME-QMIX aligns with the true optimal joint action, and the local policy assigns the highest probability to selecting this optimal joint action, effectively avoiding the misalignment issue through the use of OPTs.

2.7.2 Evaluation on SMAC-v2

We further evaluate ME-IGM on the SMACv2 [14] benchmark, over 15 scenarios. The results, as shown in Table 2.6, Table 2.7, and Table 2.8 are obtained by training each algorithm on each scenario for 10 millions of environmental steps. Each value in the table represents the average performance over three runs with different random seeds.

Method	Variant	5_vs_5	10_vs_10	10_vs_11	20_vs_20	20_vs_23
MARL	MAPPO	58.0	58.3	18.2	38.1	5.1
	IPPO	54.6	58.0	20.3	44.5	4.1
	QMIX	70.2	69.0	42.5	69.7	16.5
	QPLEX	53.3	53.7	22.8	27.2	4.8
	HASAC	22.9	12.1	5.7	-	-
	FOP	37.8	8.6	0.4	2.0	0.3
ME-IGM(Ours)	ME-QMIX	**75.9±3.0**	78.5±2.7	**50.4±2.9**	**73.9±3.0**	**23.4±1.0**
	ME-QPLEX	69.8±1.8	**78.7±3.4**	44.9±3.2	38.3±4.0	18.7±1.3
IL+MARL	IMAX-PPO	68.1	59.6	21.3	76.3	11.8
	InQ	78.7	79.8	48.7	**80.6**	**24.2**

Table 2.6: Average win rates on **SMAC-v2 (Protoss)** tasks. The results demonstrate that our method, ME-IGM, outperforms all previous baseline algorithms and achieves performance comparable to IL+MARL methods, despite not having access to expert opponent demonstrations or opponent modeling.

ME-IGM outperforms the baseline algorithms across all scenarios by a significant margin. MAPPO and IPPO, as on-policy algorithms, have lower sampling efficiency, resulting in lower win rates for models trained with 10M environmental steps. QMIX and QPLEX, lacking exploration, tend to converge to local optimum. HASAC [32] and FOP perform poorly across all scenarios. Surprisingly, ME-IGM achieves performance on par with (or even surpassing) IL+MARL methods, despite not relying on expert opponent demonstrations or opponent modeling. The results of MAPPO, IPPO, QMIX, QPLEX, IMAX-PPO, InQ are taken from Bui et al. [6].

2.7.3 Ablation Study

Benefits of entropy-driven exploration: To demonstrate the benefits of using the maximum entropy framework for exploration, we compare ME-QMIX with a modified version of QMIX that incorporates significantly higher exploration by applying an extended epsilon annealing period in the epsilon-greedy method. Figure 2.4 illustrates that simply increasing the epsilon annealing time for exploration does not improve QMIX's performance. In contrast, with the systematic exploration enabled by the maximum entropy framework, ME-QMIX achieves superior performance.

The necessity of using OPTs: In Table 2.9, we compare QMIX with its two

Method	Variant	5_vs_5	10_vs_10	10_vs_11	20_vs_20	20_vs_23
MARL	MAPPO	52.0	58.1	28.6	52.8	11.2
	IPPO	56.2	57.3	31.0	49.6	10.0
	QMIX	58.4	65.8	39.4	57.6	10.0
	QPLEX	70.0	66.1	41.4	23.9	7.0
	HASAC	37.9	20.4	9.1	-	-
	FOP	57.1	15.8	7.5	0.2	0.0
ME-IGM (Ours)	ME-QMIX	**70.2±2.6**	**72.6±1.8**	49.6±1.6	**59.3±1.3**	**18.7±0.7**
	ME-QPLEX	70.0±4.8	69.4±0.2	**51.0±0.4**	45.1±7.2	17.4±2.9
IL+MARL	IMAX-PPO	53.3	58.4	28.4	35.9	4.7
	InQ	69.9	72.2	53.9	65.4	17.7

Table 2.7: Average win rates on **SMAC-v2 (Terran)** tasks.

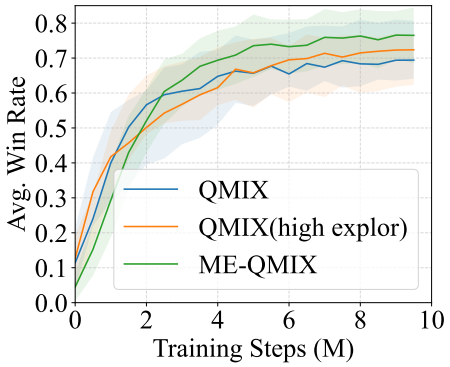


Figure 2.4: Comparison of exploration strategies in CTDE MARL: ME-QMIX with maximum entropy vs. QMIX with epsilon-greedy and extended annealing.

variants: +Entropy and +OPT. Both variants learn local stochastic policies based on Q-functions trained with the QMIX objective and softmax operations. Specifically, +OPT corresponds to our proposed ME-IGM, while +Entropy removes the OPTs from ME-IGM (i.e., no OPTs in Equations (2.4), (2.9), and (2.10)). We test each algorithm on three SMAC-v2 scenarios across four random seeds, reporting the average win rate along with its standard deviation in the table. Without OPTs, +Entropy suffers from the misalignment problem discussed in Section 2.4 and fails to ensure monotonic policy improvement, as analyzed in Section 2.5.2, leading to a performance decline in two out of three scenarios. On the other hand, +OPT achieves a substantial performance improvement across all evaluated scenarios.

Method	Variant	5_vs_5	10_vs_10	10_vs_11	20_vs_20	20_vs_23
MARL	MAPPO	41.0	39.1	31.2	31.9	15.8
	IPPO	37.2	**49.4**	26.0	31.2	8.3
	QMIX	37.2	40.8	28.0	30.4	10.1
	QPLEX	47.8	41.6	31.1	15.8	6.7
	HASAC	29.1	17.9	14.0	-	-
	FOP	35.2	8.0	1.5	0.2	0.2
ME-IGM (Ours)	ME-QMIX	50.8±4.6	49.1±0.6	32.6±1.7	34.3±5.2	14.8±6.4
	ME-QPLEX	**53.8±3.9**	42.9±1.8	**45.3±0.6**	**42.1±5.1**	**17.5±3.2**
IL+MARL	IMAX-PPO	48.6	50.6	34.8	26.7	8.2
	InQ	55.0	57.6	41.5	43.3	21.3

Table 2.8: Average win rates on **SMAC-v2 (Zerg)** tasks.

	QMIX	**+Entropy**	**+OPT**
Protoss 5vs5	0.68 ± 0.04	0.73 ± 0.05	0.74 ± 0.04
Terran 5vs5	0.68 ± 0.05	0.67 ± 0.04	0.70 ± 0.05
Zerg 5vs5	0.41 ± 0.05	0.40 ± 0.04	0.48 ± 0.05

Table 2.9: Comparison between our algorithm (+OPT) and its two ablated versions: +Entropy, obtained by removing OPTs, and QMIX, obtained by further removing softmax local policies.

To further validate the design of the order-preserving transformation, we replace the OPT in ME-QMIX with a three-layer MLP, which does not enforce positive weights, referring to this variant as ME-QMIX (MLP). As shown in Table 2.10, our method, ME-QMIX (OPT), consistently outperforms ME-QMIX (MLP) across three different SMAC-v2 scenarios. This underscores the importance of the order-preserving transformation in resolving the misalignment between local policies and the maximum global Q-value, thereby enhancing overall performance.

The impact of hyperparameters: Compared to QMIX, ME-IGM introduces two additional hyperparameters: the learning rate and the target entropy $\bar{H}$ for updating α_ω in Equation (2.10). We set the other hyperparameters of ME-IGM by following the design choices of QMIX. To analyze the impact of these additional parameters, we conduct a parameter scan on the learning rate of α_ω and the target entropy $\bar{H}$. Figure 2.5 illustrates that the learning rate of α_ω influences the exploration level of the algorithm. A lower learning rate results in higher policy entropy at the

	ME-QMIX (MLP)	ME-QMIX (OPT)
Protoss 5v5	0.58 ± 0.03	0.74 ± 0.04
Terran 5v5	0.61 ± 0.04	0.70 ± 0.05
Zerg 5v5	0.41 ± 0.04	0.48 ± 0.05

Table 2.10: Comparison between ME-QMIX (OPT) and its ablated version, ME-QMIX (MLP), where the OPT is replaced with a three-layer MLP.

beginning of training, promoting exploration. While this initially leads to weaker performance (e.g., the blue curve in Figure 2.5), it eventually achieves higher returns. Furthermore, the results in Figure 2.6 suggest that the algorithm's performance is not sensitive to the choice of the target entropy. Notably, in this study, we did not fine-tune configurations for each individual testing scenario.

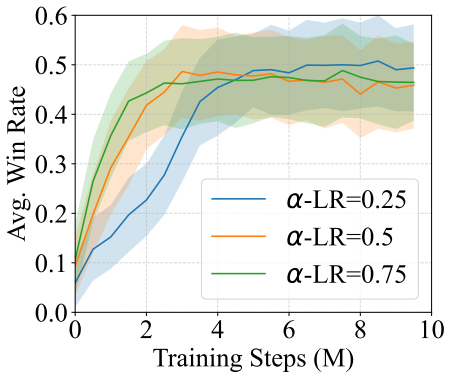


Figure 2.5: The impact of different hyperparameters for training α_ω on the performance of ME-QMIX. Results indicate that a lower learning rate for α_ω leads to increased exploration, slower convergence, and ultimately higher final performance.

2.8 Conclusion

In this chapter, we introduce ME-IGM, a novel maximum entropy MARL approach that applies the CTDE framework and satisfies the IGM condition. We identify and address the misalignment problem between local policies and the maximum global Q-value (in maximum entropy MARL) by introducing an order-preserving transformation (OPT). We further propose a theoretically sound and straightforward objective for updating these transformation operators. Empirically, we demonstrate

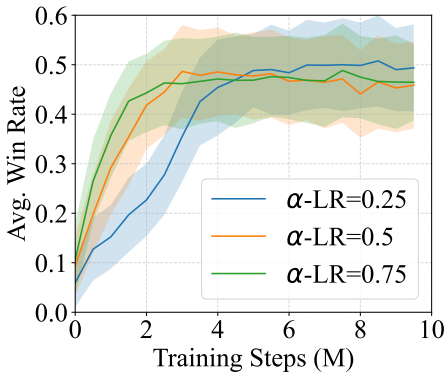


Figure 2.6: The impact of different hyperparameters for training α_ω on the performance of ME-QMIX. Results indicate that ME-QMIX's performance is not sensitive to the choice of the target entropy $\bar{H}$ in Equation (2.10).

the effectiveness of ME-IGM on matrix games and its state-of-the-art performance on the challenging SMAC-v2 benchmark. However, as a value-based method, ME-IGM is currently limited to tasks with discrete action spaces. In future work, we plan to extend its applicability to tasks involving continuous action spaces.

Chapter 3

Temporal Credit Assignment

3.1 Introduction

Online learning with LLMs, guided by self-generated data and environmental feedback, offers a promising research direction for advancing beyond the constraints of demonstration [13]. However, this task is inherently difficult because valuable environmental feedback is often sparse [5, 52]. This challenge is particularly evident in multi-turn settings, where agents must execute a sequence of correct actions to receive a reward signal. Sparse environmental feedback increases both the sample complexity and the instability of the learning process [7, 9]. In this work, we explore effective temporal credit assignment methods to convert sparse feedback into dense training signals based on a creative use of LLMs, enabling the identification of critical states and actions in the environment and facilitating more efficient online learning (of LLM agents).

In particular, we adopt LLMs as policies and update them using a novel Retrospective In-Context Learning (RICL) algorithm. RICL is designed specifically for multi-turn setting. At each turn (i.e., environment time step), RICL first analyze environmental feedback (i.e., reward signals) by leveraging the pretrained knowledge of an LLM reflector, then incorporate this analysis into the prompt to perform an in-context update of the LLM policy. Further, we introduce a novel paradigm to estimate the advantage function of an LLM policy by utilizing the log-probabilities of both the original policy and its in-context updated version. This process is theoreti-

cally grounded, and our experiments demonstrate that it can accurately estimate the advantage function with high sample efficiency. **In this way, we transform sparse reward signals into advantage functions, which are dense training signals facilitating both temporal credit assignment and policy training.** These improvements stem from the pretrained knowledge in LLMs and our new approach to estimating advantage functions using LLMs.

Further, we introduce RICOL, a novel online learning framework for LLMs that iteratively refines the policy based on the credit assignment results from RICL. Leveraging the accurate credit assignment performed by RICL and the strong generalization ability of LLMs, RICOL is significantly more sample-efficient than traditional online RL methods across multiple language-conditioned, sparse-reward sequential decision-making tasks. This demonstrates the potential of our algorithm to enable LLMs to self-improve through online learning.

In summary, our main contributions are as follows: (1) We propose RICL for temporal credit assignment, which transforms sparse reward signals into advantage functions by comparing the log-probabilities of LLM policies before and after an in-context update. (2) We propose RICOL, an online learning framework that uses advantage weighted regression to incorporate credit assignment results (i.e., advantage functions) from RICL into the LLMs' parameters. This integration enables LLM agents to self-improve based on sparse environmental feedback. (3) Empirical results demonstrate the effectiveness of RICL in temporal credit assignment and the efficacy and efficiency of RICOL in iterative policy improvement. Altogether, RICL and RICOL represent a more sample-efficient and generalizable RL paradigm for LLM agents.

3.2 Related Works

Intrinsic Self-Correction: In recent studies, LLMs have demonstrated advanced capabilities in self-verifying generated responses and correcting potential issues through in-context learning [24, 35]. However, critiques have suggested that intrinsic self-correction, in the absence of ground truth environment feedback, may not always enhance and could potentially degrade performance of LLMs [23, 36, 54]. In contrast, our focus is on learning from environmental feedback, a challenging task due to its

sparse and complex nature.

Self-Correction based on an Extra Reflector Network: Another line of research aims to improve the self-correction capability of LLMs by fine-tuning it on an external dataset [58]. It often requires to collect an additional dataset especially tailored for the training of a reflector network. For instance, Chen et al. [10] design an erroneous solution rewriting task for reflector training. In contrast, we perform self-correction using free-form environmental feedback and do not need to train an extra network.

Self-Correction based on Induction: Prior work has explored encoding past experiences into text-form memory and incorporating this memory into prompts to enhance decision-making in subsequent trials [50, 61, 64]. There is also a line of works leveraging LLMs to generate rewards – either as codes or natural language prompts – and iteratively refine these rewards based on environmental feedback through in-context learning [28, 30, 34, 60]. These studies assume that LLMs, through in-context learning, can infer environmental rules and adapt to new trajectories/experiences with only a few observed transitions. In contrast to these approaches, we assume that the rewards or reflections obtained from one trajectory are not directly transferable to other trajectories. Thus, in this work, we proposed to use in-context learning in a "retrospective" manner.

Self-Correction Through Preference Alignment: Iterative RPO [39] labels preference data based on environmental feedback and uses this feedback as a supervised signal to fine-tune LLMs. RICO-GRPO [57] use trajectory level reward and ground normalization to estimate the advantage function for multi-turn online RL training. However, these methods don't perform explicit temporal credit assignment. In contrast, our approach generates dense rewards by leveraging the pretrained knowledge of LLMs through in-context learning. This enables the policy to be trained with more informative supervision signals.

Temporal Credit Assignment using LLMs: The goal of temporal credit assignment is to identify the critical states and actions in sequential decision-making. A common approach to credit assignment involves estimating a value or advantage function based on learning experiences [3, 49]. Unlike RICO-PPO [57] training a value network from scratch, which can be sample inefficient, our approach leverages in-context learning with pretrained LLMs to perform credit assignment more effectively.

Please refer to Table 3.1 for a more intuitive comparison of our proposed method with related works on self-correction in LLMs.

	Involving Env Feedback	Requiring an Extra Reflector	Retrospective Updating	Multi-turn Credit Assignment
SELF-REFINE	✗	✗	✗	✗
STaR, Reflexion	✓	✗	✗	✗
RLMEC, Retroformer	✓	✓	✗	✗
Iterative RPO, RICO-GRPO	✓	✗	✓	✗
RICL (Ours)	✓	✗	✓	✓

Table 3.1: We propose RICL for LLMs to learn from environmental feedback, fundamentally different from intrinsic self-correction work like SELF-REFINE [35]. Unlike STaR [61] and Reflexion [50], our method utilizes environmental feedback to retrospectively update the policy. Additionally, we do not fine-tune an extra reflector, as done in RLMEC [10] and Retroformer [58]. By generating dense training signals from sparse environmental feedback, our approach is more sample-efficient than methods that rely solely on sparse preference rewards, such as Iterative RPO [39].

3.3 Preliminary

Markov Decision Process: In contrast to the MARL setting used in the previous chapter, we define the Markov Decision Process (MDP) in this chapter as a tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0)$. Here, $\mathcal{S}$ and $\mathcal{A}$ represent the state and action space, respectively; $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition kernel; $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function; $\gamma \in [0, 1)$ is the discount factor; $\rho_0(\cdot)$ denotes a distribution of the initial state.

Maximum Entropy RL: Maximum entropy RL [29], has been extensively applied for training sequential decision-making agents. The goal of maximum entropy RL is to learn a policy π that maximizes the expected return defined with the soft Q function: $\max_\pi \mathbb{E}_{s\sim\rho_0(\cdot),a\sim\pi(\cdot|s)} [Q^\pi(s, a) - \beta \log \pi(a|s)]$, where $\beta > 0$ is a scaling parameter that balances the tradeoff between exploitation and exploration and the soft value functions are defined as $Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{s'\sim P(\cdot|s,a)}[V^\pi(s')]$ and $V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^\pi(s, a) - \beta \log \pi(a|s)]$.

A widely used algorithm to solve the maximum entropy RL is the soft policy iteration [67]. Specifically, at iteration k, the process begins with policy evaluation, where the current policy π_k is used to compute Q^{π_k}. Following this, a policy improvement step is performed as follows:

$$\pi_{k+1}(a|s) \propto \exp\left(A^{\pi_k}(s,a)/\beta\right), \forall (s,a) \in \mathcal{S} \times \mathcal{A}, \tag{3.1}$$

where $A^{\pi_k}(s,a) = Q^{\pi_k}(s,a) - V^{\pi_k}(s)$ is the advantage function, which preserves the relative ranking of actions within the Q-function while reducing the sample-based estimation variance by using the value function as a baseline.

3.4 Retrospective In-Context Learning for Temporal Credit Assignment

Given the current policy and its experience, the goal of temporal credit assignment is to identify critical state-action pairs for sequential decision-making. Critical states are those where policy decisions significantly influence the expected return-to-go, while critical actions indicate policy update directions at those states. In RL, credit assignment is typically performed by estimating the advantage function, which measures the impact of different actions at a given state on the future return-to-go under the current policy. **Thus, in this section, we propose leveraging LLMs to estimate advantage functions for temporal credit assignment, using retrospective in-context learning guided by environmental feedback.**

3.4.1 LLMs as Policies

Consider the case where an LLM is used as a policy π_0[1]. This policy can be refined through in-context learning by embedding task descriptions or goal-related hints into the prompt, resulting in an in-context updated policy, π'. Unlike standard RL, the transition from π_0 to π' in in-context learning is not an explicit optimization process.

[1]For an LLM-based policy, each state s corresponds to a prompt, while each action a is represented as a sequence of tokens (e.g., "turn right" or "move forward"). Consequently, the probability $\pi(a|s)$ can be computed as the token prediction probability of a occurring after s in the LLM's autoregressive generation.

However, we can infer the implicit advantage function driving this policy improvement by analyzing the log-probabilities of π_0 and π'. The theoretical foundation of this approach is established in the following theorem.

Theorem 3.4.1. *Let $\pi_0 : \mathcal{S} \times \mathcal{A} \to (0,1)$ and $\pi' : \mathcal{S} \times \mathcal{A} \to (0,1)$ be any two policies in an MDP with transition kernel $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ and discount factor $\gamma \in [0,1]$. Then, there exists a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ such that the following relationship holds:*

$$\beta \log \frac{\pi'(a|s)}{\pi_0(a|s)} \propto A_r^{\pi_0}(s,a), \tag{3.2}$$

where $\beta > 0$ is a known scaling parameter, and $A_r^{\pi_0}(s,a)$ denotes the advantage function under policy π_0 in the finite MDP $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$.

Proof. See Appendix A.7.

Our algorithm, as detailed later, assumes the existence of a reward function such that the corresponding advantage function explains the discrepancy between the in-context updated policy π' and the actor policy π_0. Theorem 3.4.1 establishes the existence of such a reward function for any pair of policies, thereby supporting the soundness of our proposed online RL framework.

3.4.2 Implementation Details

The procedure for estimating the advantage function $A_r^{\pi_0}(s,a)$ of a policy π_0 is as follows: First, we collect n trajectories starting from the state-action pair (s,a), by executing the policy π_0. Then, for each trajectory $\tau^{(i)}$, we perform retrospective in-context learning (introduced in the following subsection) to update the LLM and get an updated policy $\pi'^{(i)}$.

According to Theorem 3.4.1, we have $\beta \log \pi'^{(i)}(a|s) = \beta \log \pi_0(a|s) + A_{r_i}^{\pi_0}(s,a) - \beta \log Z^{(i)}(s)$, where $Z^{(i)}(s)$ is a partition function to ensure that $\pi'^{(i)}$ is a valid probability distribution. In this paper, we consider only tasks with a discrete and enumerable action space, allowing $Z^{(i)}(s)$ to be computed by summing over the probabilities of all possible actions. $A_{r_i}^{\pi_0}(s,a)$ is derived from $\log \pi'^{(i)}(a|s)$ and $\log \pi_0(a|s)$ and is a sample estimate for the ground-truth advantage function $A_{r^*}^{\pi_0}(s,a)$. Thus, we can use the sample mean $\bar{A}_r^{\pi_0}$ as the estimated advantage function, which is

defined as:

$$\bar{A}_r^{\pi_0}(s,a) = \frac{\beta}{n}\sum_{i=1}^{n}\left(\log\frac{\pi'^{(i)}(a|s)}{\pi_0(a|s)} + \log Z^{(i)}(s)\right). \tag{3.3}$$

Empirical results demonstrate that $\bar{A}_r^{\pi_0}(s,a)$ closely approximates the ground-truth advantage function $A_{r^*}^{\pi}(s,a)$. **Recall the policy improvement step shown in Eq. (3.1), this alignment suggests that the in-context learning with LLMs implicitly performs maximum entropy RL.**

3.4.3 Retrospective In-Context Learning

As aforementioned, π' is updated from π_0 using Retrospective In-Context Learning (RICL), a novel in-context learning algorithm for refining LLM agents. RICL is illustrated as step ② and step ③ in Figure 3.1. First, given a state s_t and its corresponding hindsight trajectory – a sequence of future states, actions, and rewards $\{s_{t:T}, a_{t:T-1}, r_{t:T-1}\}$ where T denotes the episode horizon –we input this trajectory into a reflector LLM, $\pi_{reflect}$, to generate a verbal feedback f_t. The reflector can be any LLM, prompted to analyze the hindsight trajectory and provide corrective feedback to improve the agent's performance at state s_t. Next, we integrate the feedback f_t into the original LLM's prompt, yielding an in-context updated policy $\pi'(\cdot|s_t) = \pi_0(\cdot|s_t, f_t)$. This policy is then used to estimate the advantage function for temporal credit assignment, as introduced in the previous subsection.

RICL offers two key advantages over previous in-context learning algorithms [50]. First, it generates verbal feedback for each action individually, whereas prior methods provide a single feedback for the entire trajectory, allowing for more fine-grained guidance. Second, it employs retrospective updates, meaning the policy is updated specifically for the states it has just encountered, rather than requiring the reflector LLM to generalize to unseen states. This method reduces the complexity of tasks assigned to in-context learning and lessens the dependence on reflector LLMs.

3.5 Retrospective In-Context Online Learning

Once the log-probabilities of the in-context updated policy are obtained, which indicate the advantage function, policy improvement can be performed straightforwardly

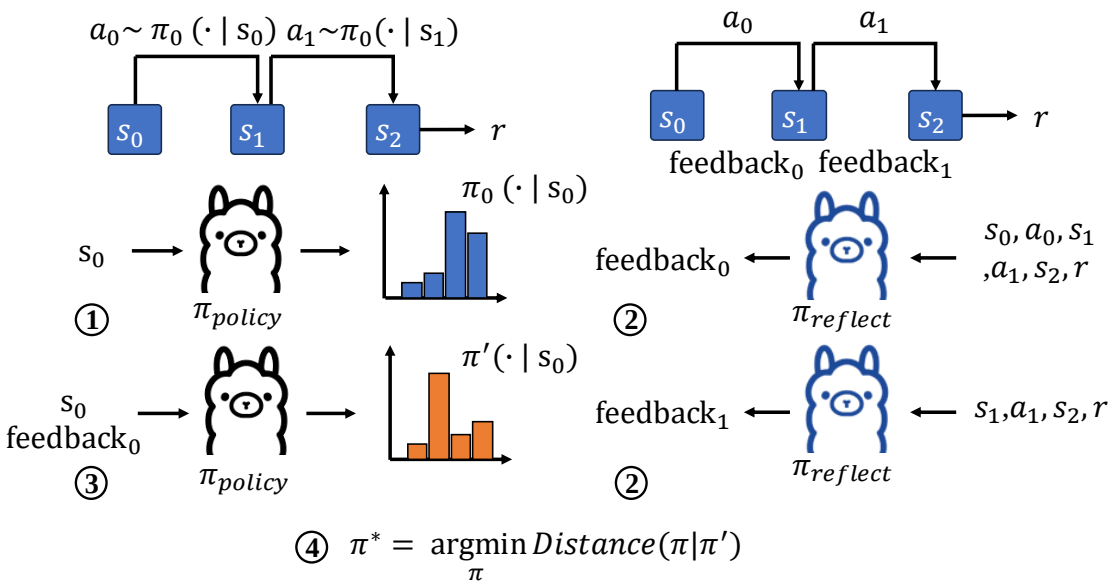


Figure 3.1: The pipeline of retrospective in-context online learning, where step ② and step ③ represent retrospective in-context learning.

using advantage-weighted regression [41]. In this section, we integrate the policy improvement phase with the previously introduced policy evaluation phase (i.e., RICL) to develop a practical online RL algorithm: Retrospective In-Context Online Learning (RICOL).

3.5.1 Policy Improvement based on RICL

To learn from the in-context improved policy π', we can adopt the following objective:

$$\pi^* = \underset{\pi}{\text{argmax}}\, \mathbb{E}_{s,a\sim\pi}\left[A^\pi(s,a)\right] = \underset{\pi}{\text{argmax}}\, \mathbb{E}_{s\sim d_\pi, a\sim\pi(\cdot|s)}\left[\log \pi'(a|s) - \log \pi_0(a|s)\right], \tag{3.4}$$

where $d_\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \rho_t^\pi(s)$ represents the discounted state occupancy measure for policy π and $\rho_t^\pi(s)$ denotes the probability that policy π visits state s at time t.

Inspired by AWR [41], we can reuse samples collected with π_0 to avoid resampling from π and so improve the algorithm's sample efficiency, with a smarter objective design:

$$\min_\pi \mathbb{E}_{s\sim d_{\pi_0}}\left[D_{KL}\left(\frac{1}{Z(s)} \odot \exp\left((1-\alpha)\log \pi_0(\cdot|s) + \alpha \log \pi'(\cdot|s)\right) || \pi(\cdot|s)\right)\right] \tag{3.5}$$

where $Z(s)$ is a partition function , $\odot$ denotes element-wise multiplication, and hyperparameter α controls the size of the trust region. The derivation from Eq. (3.4) to Eq. (3.5) is detailed in Appendix A.6.

Integrating the sampling policy π_0 into the objective function introduces trust region constraints, akin to those used in online RL [48]. These constraints regulate policy updates, ensuring that deviations from the current sampling policy remain bounded during optimization. By anchoring updates within a local region around π_0, the trust region term helps mitigate overfitting to potentially noisy verbal feedback generated through in-context learning.

3.5.2 Pipeline of RICOL

Algorithm 2 RICOL

Input: $\pi_0, \pi_{reflect}, K, n$
for $k = 0 \cdots K-1$ **do**
  for $i = 1 \cdots n$ **do**
    $\tau_k^{(i)} \sim \pi_k(\cdot|s_0)$
    for $s_t = s_0 \cdots s_{T-1} \in \tau_k^{(i)}$ **do**
      $\text{feedback}_t \sim \pi_{reflect}(\cdot|s_{t:T}, r_{t:T-1})$
      $\pi'^{(i)}_{k+1}(\cdot|s_t) \leftarrow \pi_k(\cdot|s_t, \text{feedback}_t)$
    end for
  end for
  $\pi_{k+1} \leftarrow \text{argmin}_\pi \mathbb{E}_{s\sim\tau_k}[D_{KL}\big(\frac{1}{Z(s)} \odot \exp($
  $(1-\alpha)\log\pi_k(\cdot|s) + \alpha \log \pi'(\cdot|s))||\pi(\cdot|s)\big)]$
end for

Trajectory Collection: During iteration k, the policy π_k interacts with the environment to collect a batch of trajectories τ_k. Each trajectory comprises state-action-reward sequences $(s_{1:T}, a_{1:T-1}, r_{1:T-1})$.

Policy Evaluation: For each state s_t in $\tau_k^{(i)}$, RICOL applies RICL (detailed in Section 3.4.3) to perform an in-context update on the policy $\pi_k(\cdot|s_t)$, resulting in an improved policy $\pi'^{(i)}_{k+1}(\cdot|s_t)$. To estimate $\pi'_{k+1}(\cdot|s_t)$, we follow the process: $[\pi'^{(i)}_{k+1}(\cdot|s_t), \pi_k(\cdot|s_t)]_{i=1}^n \rightarrow [A_{r_i}^{\pi_k}(s_t, \cdot)]_{i=1}^n \rightarrow \bar{A}_r^{\pi_k}(s_t, \cdot) \rightarrow \pi'_{k+1}(\cdot|s_t)$, where the first two

steps are detailed in Section 3.4.2 and the last step is according to $\beta \log \frac{\pi'_{k+1}(a_t|s_t)}{\pi_k(a_k|s_t)} \propto \bar{A}_r^{\pi_k}(s_t, a_t)$ (i.e., Eq. (3.2)).

Policy Improvement: After computing $\pi'_{k+1}(\cdot|s_t)$ across the batch of states in τ_k, the policy π_k is updated via gradient descent to minimize the objective in Eq. (3.5). For environments with small, discrete action spaces, the KL-divergence in Eq. (3.5) is computed exactly by enumerating all actions. This requires querying the LLM $|\mathcal{A}|$ times per state. For general tasks, the KL-divergence can be estimated by sampling from $\pi^*(\cdot|s) \propto \exp\left((1-\alpha)\log \pi_k(\cdot|s) + \alpha \log \pi'(\cdot|s)\right)$, avoiding exhaustive action enumeration.

3.6 Experiments

In this section, we present a series of experiments to address the following research questions (RQs): **RQ1**: How effective is RICL at credit assignment in multi-turn tasks? Compared to traditional RL methods, how sample-efficient is RICL in credit assignment? **RQ2**: The proposed method relies on LLMs, which may be unreliable. How does RICL compare to other reflection-based LLM methods in terms of reliability? Given that RICL may be unreliable, can RICOL still learn effectively from noisy labels? **RQ3**: Can RICOL efficiently learn from sparse rewards in multi-turn tasks? How does its performance compare to baseline methods, and what types of tasks is it best suited for?

3.6.1 Environments

We evaluate our algorithms on a 1D Key-Door environment and four BabyAI scenarios. In all settings, both the policy inputs (observations) and outputs (actions) are represented in text form. The action space is discrete and enumerable. In the 1D Key-Door environment, the agent operates on a 1D grid world, moving along the x-axis. The key is placed in the leftmost cell, and the door in the rightmost. The agent starts without the key and must first move left to retrieve it, then move right to unlock the door. This setup enables exact computation of the ground-truth advantage at each cell, making it ideal for evaluating RICL's credit assignment capabilities. We test RICOL on four BabyAI [8] tasks: *goto*, *pickup*, *pick_up_seq_go_to*, and *open*.

BabyAI is a 2D grid world where the agent must complete language-conditioned tasks. These tasks are challenging to LLMs, due to their limited spatial reasoning abilities and the long episode lengths. As shown in Figure 3.4, even GPT-4o mini achieves a success rate of no more than 40% across all four BabyAI scenarios. More details on the tasks and associated prompt templates are available in Appendix A.9.

3.6.2 Sample Efficiency of RICL in Credit Assignment (RQ1)

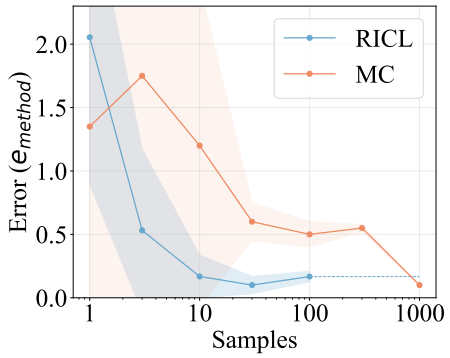


Figure 3.2: Comparison of error in advantage function estimation. The x-axis represents the number of trajectories used for estimation, while the y-axis shows the mean error between the estimated advantage and the ground-truth advantages. RICL achieves significantly lower error with fewer samples (around 10) compared to Monte Carlo, which requires about 1000 samples for similar accuracy. Additionally, the error of RICL is more stable across trials (lower variance).

In this subsection, we demonstrate that RICL achieves greater sample efficiency ($100\times$) compared to classic Monte Carlo (MC) methods when estimating the advantage function in the 1D Key-Door scenario. Since RL methods alternate between policy evaluation (value function estimation) and policy improvement, a more sample-efficient policy evaluation approach can significantly speed up the overall training process.

We measure the discrepancy between estimated and ground-truth advantage functions for both MC and RICL. For each (s_t, a_t), we generate n trajectories starting from s_t using the policy π_0. MC estimates the Q-value $\hat{Q}^{\pi_0}(s_t, a_t)$ as the average return-to-go across the n trajectories. The estimated advantage function is defined as $\hat{A}^{\pi_0}(s_t, a_t) = \hat{Q}^{\pi_0}(s_t, a_t) - \hat{V}^{\pi_0}(s_t)$, where $\hat{V}^{\pi_0}(s_t) = \mathbb{E}_{a\sim\pi_0(\cdot|s_t)}[\hat{Q}^{\pi_0}(s_t, a_t)]$. The updated policy

under MC is accordingly defined as $\pi_{\text{MC}}(a_t|s_t) \propto \pi_0(a_t|s_t)\exp(\hat{A}^{\pi_0}(s_t,a_t)/\beta)$. On the other hand, RICL estimates the advantage function $\bar{A}^{\pi_0}(s_t,a_t)$ directly from n trajectories using the procedure outlined in Section 3.4.2. The corresponding policy $\pi_{\text{RICL}}(a_t|s_t)$ is similarly defined as $\pi_{\text{RICL}}(a_t|s_t) \propto \pi_0(a_t|s_t)\exp(\bar{A}^{\pi_0}(s_t,a_t)/\beta)$. Finally, to evaluate both methods, we leverage ground-truth advantage function $A^{\pi_0}(s_t,a_t)$ to compute the reference policy $\pi_{\text{gt}}(a_t|s_t) \propto \pi_0(a_t|s_t)\exp(A^{\pi_0}(s_t,a_t)/\beta)$.

The performance of MC and RICL is assessed by computing the expected KL divergence between their respective policies and the ground-truth policy. This expectation is taken over states sampled from ρ_0, a uniform distribution over all possible initial states in the 1D grid-world. Formally, the error for each method is defined as: $e_{\text{method}} = \mathbb{E}_{s\sim\rho_0}\left[D_{KL}\left(\pi_{\text{method}}(\cdot|s) \,\|\, \pi_{\text{gt}}(\cdot|s)\right)\right]$. By definition of the policies, when $\pi_{\text{method}}(\cdot|s)$ closely approximates $\pi_{\text{gt}}(\cdot|s)$, the estimated advantage function likewise approaches the ground-truth advantage function.

For each value of n (number of trajectories), we run eight trials with different random seeds. We report the mean KL divergence and standard deviation. These results are visualized in Figure 3.2, where the x-axis represents the number of samples n. As shown in Figure 3.2, LLMs achieve an accurate estimate of advantage functions with just 10 samples, while MC requires around 1000 samples to reach a similar accuracy. Furthermore, the standard deviation of the error of RICL is much lower than that of MC. This suggests that RICL is particularly advantageous for tasks that require high sample efficiency, where leveraging the pretrained knowledge of LLMs is crucial. **In Appendix A.11, we compare RICL with an LLM-based method (Retroformer [58]) in terms of sample efficiency for value function estimation. Further, in Appendix A.10, we show that RICL can identify critical states in sequential decision-making, based on the result of credit assignment.**

3.6.3 RICL Enables Safer In-Context Updates via Retrospective Design (RQ2)

Standard ICL follows these steps: (1) Use an actor LLM to generate a trajectory τ; (2) Use a reflector LLM to generate verbal feedback based on τ; (3) Use the verbal feedback to perform an in-context update of the actor. While integrating ICL into

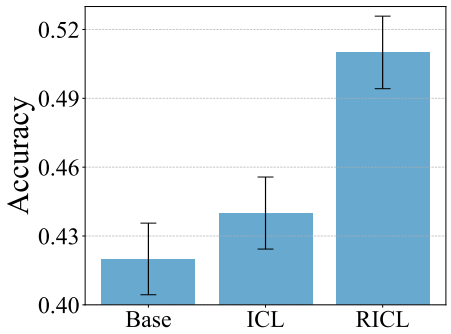


Figure 3.3: Accuracy comparison of ICL and RICL on predicting expert actions in the BabyAI *goto* scenario across 1000 trajectories. The Base bar shows the zero-shot performance of LLaMA-3.1-8B-Instruct. RICL outperforms ICL by 7.2%, demonstrating the effectiveness of retrospective updates.

the training loop improves sample efficiency, it also introduces instability, because we cannot assume that experiences from one trajectory can be applied to any other state in the environment. As an improvement, RICL shares steps (1)–(3) with ICL, but uses the in-context updated policy to compute the action distribution over **the states in τ only**. In this section, we empirically demonstrate that using environmental feedback to retrospectively in-context update the policy on the same trajectory that produced the feedback (i.e., RICL) is more effective and safer than updating the policy on a different trajectory (i.e., ICL).

Figure 3.3 compares RICL and ICL in the BabyAI *goto* scenario and uses LLaMA-3.1-8B-Instruct as actor and reflector LLMs. We conduct experiments on 1000 different trajectories. We use accuracy as the evaluation metric, defined as the probability that the policy selects the expert action. The base policy reflects the performance of the zero-shot LLaMA-3.1-8B-Instruct model. In terms of accuracy, RICL outperforms ICL by 7.2%, confirming that retrospectively updating is safer and more effective.

3.6.4 Benchmarking RICOL (RQ3)

Here, we compare RICOL against four baseline algorithms across four BabyAI scenarios and demonstrate that it achieves state-of-the-art performance in solving multi-turn tasks with higher sample efficiency. We adopt Llama-3.2-3B-Instruct as the policy and GPT-4o mini as the reflector. Regarding evaluation metrics, we use environment time steps (for sampling) and task success rates to evaluate sample

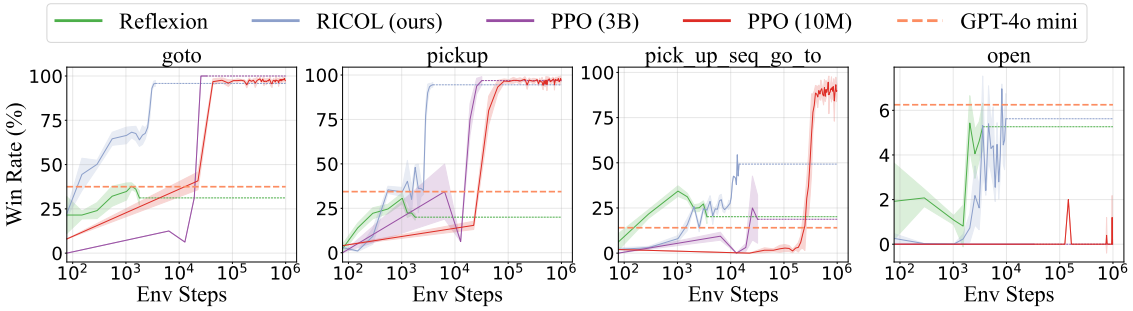


Figure 3.4: Comparison of our method (RICOL) against four baseline algorithms across four BabyAI scenarios. RICOL consistently demonstrates superior sample efficiency, achieving strong performance with significantly fewer interactions. Notably, RICOL outperforms both PPO (10M) and PPO (3B), by over 50× and 10× fewer environment steps, respectively. Compared to Reflexion, an in-context learning method using trajectory-level verbal feedback, RICOL exhibits better convergent performance by leveraging temporal credit assignment (from RICL) and state-specific feedback. Additionally, RICOL surpasses GPT-4o mini, despite using a smaller policy model (LLaMA-3.2-3B-Instruct), underscoring the importance of interactive learning from the environment. **As a useful trick to boost performance, we use the real environment rewards as the advantage and apply advantage-weighted regression during the second stage of training, after RICOL completes its predefined training schedule in the first stage.**

efficiency and effectiveness of the algorithms, respectively. The remaining experiments in the paper report the mean and standard deviation across three different random seeds.

Baselines: (1) GPT-4o mini: This baseline uses GPT-4o-mini as the policy, providing a measure of the zero-shot performance of state-of-the-art LLMs on the benchmark tasks. Moreover, since GPT-4o-mini serves as the reflector in our algorithm, improvements over this baseline demonstrate that simple behavioral cloning of the reflector is insufficient, highlighting the necessity of our algorithmic design for effective policy improvement. **(2) Reflexion:** Reflexion [50] relies on in-context learning with iterative self-reflection. A base policy first interacts with the environment to collect a trajectory, which is used to prompt an LLM for verbal feedback. The policy then uses this feedback to generate a new trajectory, which is again used to update the feedback. This process repeats iteratively. **(3) PPO (3B):** We fine-tune a Qwen2.5-3B-Instruct model using PPO, where the 3B model serves as both the policy's and the critic's

backbone. We use Qwen instead of LLaMA due to prior findings [17] suggesting Qwen exhibits stronger reasoning behaviors such as verification and backtracking. **(4) PPO (10M):** This baseline uses two randomly initialized 3-layer MLPs for the policy and critic, respectively. It serves to test whether LLMs' pretrained knowledge provides any advantage during learning.

Results: (1) Figure 3.4 shows that the in-context learning method *Reflexion* is sample-efficient, but its performance gains saturate quickly. We attribute this to two factors. First, unlike other algorithms, in-context learning does not update the LLM's parameters, limiting its ability to effectively acquire new knowledge. Second, *Reflexion* relies on verbal feedback at the trajectory level, while our method provides verbal feedback for each specific state. As a result, *Reflexion* saturates early because the general (trajectory-level) rules it can easily capture from the environment are limited. Here is an example of the verbal feedback learned by *Reflexion*: "In future navigation tasks, look for opportunities to reposition yourself to move closer to your goal". In contrast, our method generates more precise, state-specific feedback: "Prioritize picking up the green key first by taking action D when you are in front of it, rather than moving towards the grey box." **(2)** Regarding comparisons with PPO-based methods, PPO (10M) requires approximately 0.1 to 1 million time steps to converge across most scenarios, highlighting the sample inefficiency of training from scratch without leveraging LLMs' pretrained knowledge. In contrast, our method achieves comparable performance while using nearly 50 times less data. PPO (3B), which adopts LLMs as both the policy and critic, converges in about 20,000 time steps on most scenarios. Our method is approximately 10× **more sample efficient** than this approach. While our algorithm also uses an LLM for the policy, it does not require training a critic. Instead, our value function approximation method, RICL, is more sample-efficient than Monte Carlo-based ones (as used in PPO), further supporting the argument made in Section 3.6.2. **(3)** Finally, our method outperforms GPT-4o-mini in success rates across all four scenarios, despite using a smaller 3B model as the policy. This underscores the importance of learning through interaction with the environment, rather than merely imitating the outputs of a larger models. In addition to sample efficiency, we also report the training time of each algorithm on each task in Table A.6. In summary, RICOL is well-suited for settings with limited simulation budgets, typically ranging from 1,000 to 10,000 environment steps. In

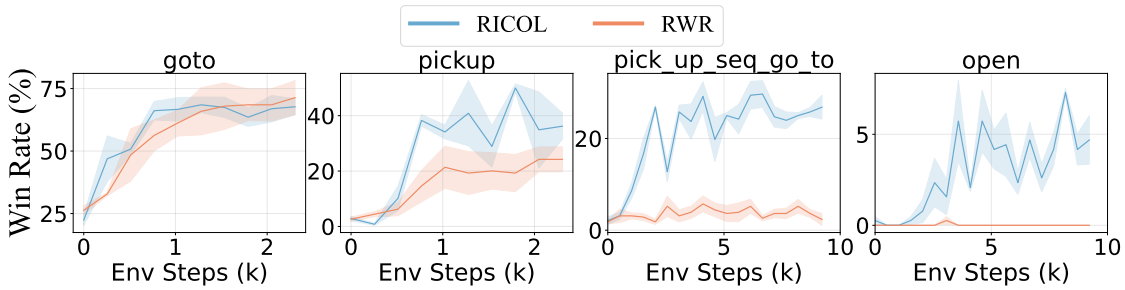


Figure 3.5: RICOL employs in-context credit assignment to generate dense learning signals, enabling more sample-efficient policy training. In contrast, RWR lacks credit assignment, depends on strong base policies with high initial success rates, and performs poorly on tasks with sparse rewards.

these scenarios, where environment interactions are costly, sample efficiency is crucial, making our method a compelling and practical choice.

3.6.5 RICOL Benefits from Credit Assignment (RQ3)

RICOL can be interpreted as a combination of RICL for credit assignment and AWR for policy improvement. In contrast, RWR [42] directly uses trajectory-level rewards as returns to perform AWR updates on an LLM policy. Unlike RICOL, RWR does not assign credit to individual time steps within an episode, making it less effective for tasks with sparse rewards. Here, we compare RWR with RICOL to show that RICOL benefits from credit assignment.

Figure 3.5 shows that RWR achieves competitive performance only in the *goto* scenario, where the base policy already attains a relatively high success rate of 25%. This highlights the limitation of relying solely on trajectory-level rewards. In contrast, our method leverages RICL to generate dense supervised signals, as described in Section 3.4, thereby enabling more effective policy training for LLMs, particularly in sparse-reward environments.

3.6.6 RICOL is Robust to Noisy Verbal Feedback (RQ2)

Figure 3.6 illustrates how the accuracy of verbal feedback affects the performance of RICOL in the BabyAI *goto* scenario. In this experiment, we train the agent

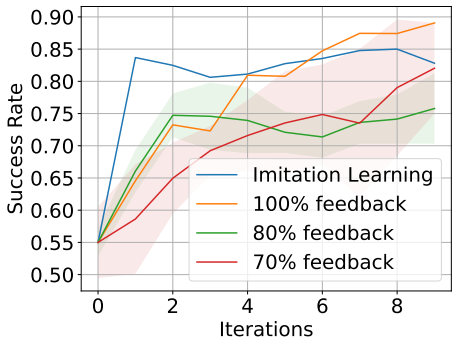


Figure 3.6: The performance of RICOL under varying verbal feedback accuracy. The agent is trained with hand-crafted verbal feedback of different accuracy levels. Despite the presence of noise, RICOL maintains strong performance. Note that 50% accuracy corresponds to random feedback.

using hand-crafted verbal feedback and systematically vary its accuracy to assess RICOL's robustness. The hand-crafted verbal feedback acts as a binary correctness indicator for the actions in trajectories, framed as: "In the previous attempt, I chose action [Action]. This time, I will maintain/change the selected action." To manipulate feedback accuracy, we randomly flip the correctness labels in the verbal feedback at different rates, thus simulating various levels of feedback reliability. An accuracy of 100% corresponds to perfectly accurate feedback, while random feedback yields an accuracy of approximately 50%. As shown in Figure 3.6, RICOL remains effective even when verbal feedback accuracy drops to 70%. This result demonstrates that RICOL does not rely heavily on accurate verbal feedback during training. We attribute this robustness to the trust region term introduced in the loss function during the policy improvement step (Eq. 3.5), which helps normalize the impact of noisy supervision and stabilize learning.

3.7 Conclusion

We propose a novel in-context learning algorithm, RICL, to convert sparse environmental feedback into dense training signals for estimating advantage functions and conducting temporal credit assignment. We further propose an online learning framework, RICOL, for iterative policy improvement based on the RICL results. Our method mitigates the instability of in-context learning and enables continuous policy

refinement. Empirical results on the 1D Key-Door scenario demonstrate RICL's effectiveness in credit assignment, while results on the BabyAI benchmark show that RICOL is significantly more sample-efficient than traditional online RL methods. As a future direction, we aim to extend and test our algorithm on token-level MDPs and reasoning tasks.

Limitations: RICOL only supports tasks with discrete, finite action spaces, as when calculating the KL divergence in Eq. 3.5, it needs to calculate the partition term Z by enumerating all the actions. We believe this limitation is acceptable, as we use LLMs as policies and their output space, i.e., the token space, is inherently discrete. When the action space is not enumerable, for example, when the policy generates extremely long chains of thought before producing the next action, we can replace action enumeration with action sampling. We leave this extension for future work.

Chapter 4

Conclusion

This thesis advances reinforcement learning by addressing two fundamental challenges in credit assignment: multi-agent coordination and temporal reward attribution. The proposed ME-IGM framework resolves misalignment between local policies and global objectives in cooperative MARL, introducing order-preserving transformations to maintain consistency while maximizing entropy for exploration. For sequential decision-making, RICOL leverage LLMs to perform sample-efficient temporal credit assignment through retrospective analysis, bypassing traditional critic networks learning. By converting sparse rewards into dense learning signals, these methods enable rapid policy improvement, outperforming PPO and reflection-based baselines in language-conditioned agent tasks.

Together, these contributions offer practical solutions for RL in real-world settings, where sparse feedback and multi-agent coordination are common. Future work could extend ME-IGM to continuous control or apply RICOL to broader reasoning tasks. By refining how agents attribute credit, across both time and teammates, this research moves RL closer to scalable, generalizable deployment in complex environments.

Appendix A

Appendix

A.1 TD(λ)

In Chapter 2, we use TD(λ) to update the Q-value. In this paragraph, we will derive the expression of TD(λ) in the context of Maximum Entropy Reinforcement Learning. Specifically, we aim to demonstrate the derivation of the following equation:

$$\mathcal{T}_{\lambda}^{\pi_{jt}} Q_{tot}(s_t, \mathbf{u_t}) - Q_{tot}(s_t, \mathbf{u_t}) = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V \tag{A.1}$$

where $\delta_t^V = -Q_{tot}(s_t, \mathbf{u}_t) + r_t + \gamma V_{tot}(s_{t+1})$.

Proof. The one-step TD error can be expressed in the following form:

$$\begin{aligned} \delta_t^V &= -Q_{tot}(s_t, \mathbf{u}_t) + r_t + \gamma V_{tot}(s_{t+1}) \\ \delta_{t+1}^V &= -Q_{tot}(s_{t+1}, \mathbf{u}_{t+1}) + r_{t+1} + \gamma V_{tot}(s_{t+2}) \end{aligned} \tag{A.2}$$

Next, the k-step TD error can be represented in the following form:

$$\begin{aligned} \epsilon_t^{(1)} &:= -Q_{tot}(s_t, \mathbf{u}_t) + r_t + \gamma V_{tot}(s_{t+1}) = \delta_t^V \\ \epsilon_t^{(2)} &:= -Q_{tot}(s_t, \mathbf{u}_t) + r_t + \gamma r_{t+1} - \log \pi_{jt}(\mathbf{u_{t+1}}|s_{t+1}) + \gamma^2 V_{tot}(s_{t+1}) = \delta_t^V + \gamma \delta_{t+1}^V \\ \epsilon_t^{(k)} &= \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V \end{aligned} \tag{A.3}$$

$$
\begin{aligned}
\epsilon_t^{td(\lambda)} &:= (1-\lambda)\left(\epsilon_t^{(1)} + \lambda\epsilon_t^{(2)} + \lambda^2\epsilon_t^{(3)} + \cdots\right) \\
&= (1-\lambda)\big(\delta_t^V + \lambda(\delta_t^V + \gamma\delta_{t+1}^V) + \lambda^2(\delta_t^V + \gamma\delta_{t+1}^V + \gamma^2\delta_{t+2}^V) + \cdots\big) \\
&= (1-\lambda)\left(\delta_t^V(1+\lambda+\lambda^2+\cdots) + \gamma\delta_{t+1}^V(\lambda+\lambda^2+\cdots) + \gamma^2\delta_{t+2}^V(\lambda^2+\lambda^3+\lambda^4+\cdots) + \cdots\right) \\
&= (1-\lambda)\left(\delta_t^V\left(\frac{1}{1-\lambda}\right) + \gamma\delta_{t+1}^V\left(\frac{\lambda}{1-\lambda}\right) + \cdots\right) \\
&= \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}^V
\end{aligned}
\tag{A.4}
$$

□

It is important to note that when computing the TD lambda error in the context of a reverse view and maximum entropy reinforcement learning, it is necessary to subtract the $\log(\pi)$ term from the $t+1$ step's return as well as subtract $\log(\pi)$ from Q_{t+1} in order to obtain results consistent with Equation A.4.

A.2 Proof of Theorem 2.5.1

Lemma A.2.1 (Joint Soft Policy Improvement)**.** *Suppose* $\pi_{jt}^{old} \in \Pi$ *and consider* π_{jt}^{new} *to be the solution to Equation 2.3. Then* $Q^{\pi_{jt}^{new}}(s_t, \mathbf{u}_t) \geq Q^{\pi_{jt}^{old}}(s_t, \mathbf{u}_t)$ *for all* $(s_t, \mathbf{u}_t) \in S \times \mathbf{U}$ *with* $|\mathbf{U}| < \infty$.

Proof. Let $\pi_{jt}^{\text{old}} \in \Pi$ and let $Q^{\pi_{jt}^{\text{old}}}$ and $V^{\pi_{jt}^{\text{old}}}$ be the corresponding soft state-action value and soft state value. The update rule of π_{new} can be defined as:

$$
\begin{aligned}
\pi_{\text{new}}(\cdot \mid s_t) &= \arg\min_{\pi' \in \Pi} \mathrm{D}_{KL}\left(\pi'(\cdot) \parallel \exp(Q^{\pi_{jt}^{\text{old}}}(s_t, \cdot)) - \log Z^{\pi_{jt}^{\text{old}}}(s_t)\right) \\
&= \arg\min_{\pi' \in \Pi} J_{\pi_{jt}^{\text{old}}}\left(\pi'(\cdot \mid s_t)\right),
\end{aligned}
\tag{A.5}
$$

where $Z^{\pi_{jt}^{\text{old}}}(s_t)$ is the normalization term.

Since we can always choose $\pi_{\text{new}} = \pi_{\text{old}} \in \Pi$, the following inequality always hold true: $J_{\pi_{jt}^{\text{old}}}(\pi_{\text{new}}(\cdot \mid s_t)) \leq J_{\pi_{jt}^{\text{old}}}(\pi_{jt}^{\text{old}}(\cdot \mid s_t))$. Hence

$$\begin{aligned}&\mathbb{E}_{\mathbf{u}_t\sim\pi_{jt}^{\text{new}}}\left[\log(\pi_{jt}^{\text{new}}(\mathbf{u}_t \mid s_t)) - Q^{\pi_{jt}^{\text{old}}}(s_t,\mathbf{u}_t) + \log Z^{\pi_{jt}^{\text{old}}}(s_t)\right]\\ &\leq \mathbb{E}_{\pi_{jt}^{\text{old}}}\left[\log(\pi_{jt}^{\text{old}}(\mathbf{u}_t \mid s_t)) - Q^{\pi_{jt}^{\text{old}}}(s_t,\mathbf{u}_t) + \log Z^{\pi_{jt}^{\text{old}}}(s_t)\right],\end{aligned} \tag{A.6}$$

and the inequality reduces to the following form since partition function $Z^{\pi_{jt}^{\text{old}}}$ depends only on the state,

$$\mathbb{E}_{\mathbf{u}_t\sim\pi_{jt}^{\text{new}}}\left[Q^{\pi_{jt}^{\text{old}}}(s_t,\mathbf{u}_t) - \log\pi_{jt}^{\text{new}}(\mathbf{u}_t \mid s_t)\right] \geq V^{\pi_{jt}^{\text{old}}}(s_t). \tag{A.7}$$

Next, consider the soft Bellman equation:

$$\begin{aligned}Q^{\pi_{jt}^{\text{old}}}(s_t,\mathbf{u}_t) &= r(s_t,\mathbf{u}_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}\left[V^{\pi_{jt}^{\text{old}}}(s_{t+1})\right]\\ &\leq r(s_t,\mathbf{u}_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}\left[\mathbb{E}_{\mathbf{u}_{t+1}\sim\pi_{jt}^{\text{new}}}\left[Q^{\pi_{\text{old}}}(s_{t+1},\mathbf{u}_{t+1}) - \log\pi_{jt}^{\text{new}}(\mathbf{u}_{t+1} \mid s_{t+1})\right]\right]\\ &\vdots\\ &\leq Q^{\pi_{jt}^{\text{new}}}(s_t,\mathbf{u}_t),\end{aligned} \tag{A.8}$$

where we can repeatedly expand $Q^{\pi_{jt}^{\text{old}}}$ on the RHS by applying the soft Bellman equation, progressing from the second line to the final one.

□

Theorem A.2.2. *Denote the policy before the policy improvement as π_{old} and the policy achieving the optimality of the improvement step as π_{new}. Updating the policy with Equation (2.4) ensures that $Q_{tot}^{\pi_{old}}(s_t,\mathbf{u_t}) - \epsilon \leq Q_{tot}^{\pi_{new}}(s_t,\mathbf{u_t}), \forall s_t,\mathbf{u_t}$. The gap ϵ satisfies:*

$$\epsilon \leq \frac{\gamma}{1-\gamma}\mathbb{E}_{s_{t+1}\sim\mathcal{P}}\left[C\sqrt{2D_{KL}\big(\pi_{old}(\cdot|s_{t+1})|\pi_{new}(\cdot|s_{t+1})\big)}\right] \tag{A.9}$$

where $\mathcal{P}(s_t,\mathbf{u_t},\cdot)$ is the transition function and $C = \max_{\mathbf{u}_{t+1}} \Delta_Q(s_{t+1},\mathbf{u}_{t+1}) = \max_{\mathbf{u}_{t+1}} |Q_{tot}^{\pi_{old}}(s_{t+1},\mathbf{u}_{t+1}) - \sum_i OPT_i(Q_i^{\pi_{old}}(o_{t+1}^i,u_{t+1}^i))|$.

Proof. We define the value function as $V'^{\pi_{jt}^{\text{new}}}(s) = \mathbb{E}_{\mathbf{u}_{t+1}\sim\pi_{jt}^{\text{new}}}\big[\sum_i \text{OPT}_i(Q_i^{\pi_{jt}^{\text{new}}}(o_{t+1}^i,u_{t+1}^i)) - \log\pi_{jt}^{\text{new}}(\mathbf{u}_{t+1} \mid s_{t+1})\big]\big]$. Then, similar to Equation A.7, during policy improvement step, we have:

$$\mathbb{E}_{\mathbf{u}_t \sim \pi_{jt}^{\text{new}}} \left[\sum_i \text{OPT}_i(Q_i^{\pi_{jt}^{\text{old}}}(o_t^i, u_t^i)) - \log \pi_{jt}^{\text{new}}(\mathbf{u}_t \mid s_t) \right] \geq V'^{\pi_{jt}^{\text{old}}}(s_t). \tag{A.10}$$

By expanding the soft Bellman equation, we have:

$$\begin{aligned}
Q_{tot}^{\pi_{jt}^{\text{old}}}(s_t, \mathbf{u}_t) &= r(s_t, \mathbf{u}_t) + \gamma \mathbb{E}_{s_{t+1} \sim p} \left[V_{tot}^{\pi_{jt}^{\text{old}}}(s_{t+1}) \right] \\
&= r(s_t, \mathbf{u}_t) + \gamma \big(\mathbb{E}_{s_{t+1} \sim p} \left[V'^{\pi_{jt}^{\text{old}}}(s_{t+1}) \right] + \epsilon_1 \big) \\
&\leq r(s_t, \mathbf{u}_t) + \gamma \mathbb{E}_{s_{t+1}} \big[\mathbb{E}_{\mathbf{u}_{t+1} \sim \pi_{jt}^{\text{new}}} \big[\sum_i \text{OPT}_i(Q_i^{\pi_{jt}^{\text{old}}}(o_{t+1}^i, u_{t+1}^i)) \\
&\qquad - \log \pi_{jt}^{\text{new}}(\mathbf{u}_{t+1} \mid s_{t+1}) \big] \big] + \gamma \epsilon_1 \\
&= r(s_t, \mathbf{u}_t) + \gamma \mathbb{E}_{s_{t+1}} \big[\mathbb{E}_{\mathbf{u}_{t+1} \sim \pi_{jt}^{\text{new}}} \big[Q_{tot}^{\pi_{\text{old}}}(s_{t+1}, \mathbf{u}_{t+1}) \\
&\qquad - \log \pi_{jt}^{\text{new}}(\mathbf{u}_{t+1} \mid s_{t+1}) \big] \big] + \gamma(\epsilon_1 + \epsilon_2) \\
&\vdots \\
&\leq Q_{tot}^{\pi_{jt}^{\text{new}}}(s_t, \mathbf{u}_t) + \epsilon,
\end{aligned} \tag{A.11}$$

where p is the transition function $\mathcal{P}(s_t, a_t, \cdot)$, and ϵ is the cumulative sum of ϵ_1 and ϵ_2 over time. ϵ_1 and ϵ_2 are defined as follows:

$$\begin{aligned}
\epsilon_1 &= \mathbb{E}_{s_{t+1} \sim p} \left[V_{tot}^{\pi_{jt}^{\text{old}}}(s_{t+1}) - V'^{\pi_{jt}^{\text{old}}}(s_{t+1}) \right] \\
\epsilon_2 &= \mathbb{E}_{s_{t+1} \sim p} \big[\mathbb{E}_{\mathbf{u}_{t+1} \sim \pi_{jt}^{\text{new}}} \big[\sum_i \text{OPT}_i(Q_i^{\pi_{jt}^{\text{old}}}(o_{t+1}^i, u_{t+1}^i)) - Q_{tot}^{\pi_{\text{old}}}(s_{t+1}, \mathbf{u}_{t+1}) \big] \big]
\end{aligned} \tag{A.12}$$

Then, we have:

$$
\begin{aligned}
\epsilon_1 + \epsilon_2 &= \mathbb{E}_{s_{t+1}\sim p}\Big[V_{tot}^{\pi_{jt}^{\text{old}}}(s_{t+1}) - V'^{\pi_{jt}^{\text{old}}}(s_{t+1})\Big] \\
&\quad + \mathbb{E}_{s_{t+1}\sim p}\big[\mathbb{E}_{\mathbf{u}_{t+1}\sim\pi_{jt}^{\text{new}}}\big[\sum_i \text{OPT}_i(Q_i^{\pi_{jt}^{\text{old}}}(o_{t+1}^i, u_{t+1}^i)) - Q_{tot}^{\pi_{\text{old}}}(s_{t+1}, \mathbf{u}_{t+1})\big]\big] \\
&= \mathbb{E}_{s_{t+1}\sim p}\left[\mathbb{E}_{\mathbf{u}_{t+1}\sim\pi_{jt}^{\text{old}}}\big[Q_{tot}^{\pi_{\text{old}}}(s_{t+1}, \mathbf{u}_{t+1}) - \sum_i \text{OPT}_i(Q_i^{\pi_{jt}^{\text{old}}}(o_{t+1}^i, u_{t+1}^i))\big]\right] \\
&\quad + \mathbb{E}_{s_{t+1}\sim p}\big[\mathbb{E}_{\mathbf{u}_{t+1}\sim\pi_{jt}^{\text{new}}}\big[\sum_i \text{OPT}_i(Q_i^{\pi_{jt}^{\text{old}}}(o_{t+1}^i, u_{t+1}^i)) - Q_{tot}^{\pi_{\text{old}}}(s_{t+1}, \mathbf{u}_{t+1})\big]\big] \\
&= \mathbb{E}_{s_{t+1}\sim p}\big[\sum_{\mathbf{u}_{t+1}} \big(\pi_{jt}^{\text{old}}(\mathbf{u}_{t+1}|s_{t+1}) - \pi_{jt}^{\text{new}}(\mathbf{u}_{t+1}|s_{t+1})\big)\big(Q_{tot}^{\pi_{\text{old}}}(s_{t+1}, \mathbf{u}_{t+1}) \\
&\qquad - \sum_i \text{OPT}_i(Q_i^{\pi_{jt}^{\text{old}}}(o_{t+1}^i, u_{t+1}^i))\big)\big] \\
&\leq \mathbb{E}_{s_{t+1}\sim p}\big[\sum_{\mathbf{u}_{t+1}} |\pi_{jt}^{\text{old}}(\mathbf{u}_{t+1}|s_{t+1}) - \pi_{jt}^{\text{new}}(\mathbf{u}_{t+1}|s_{t+1})||Q_{tot}^{\pi_{\text{old}}}(s_{t+1}, \mathbf{u}_{t+1}) \\
&\qquad - \sum_i \text{OPT}_i(Q_i^{\pi_{jt}^{\text{old}}}(o_{t+1}^i, u_{t+1}^i))|\big] \\
&\leq \mathbb{E}_{s_{t+1}\sim p}\big[C \cdot \sum_{\mathbf{u}_{t+1}} |\pi_{jt}^{\text{old}}(\mathbf{u}_{t+1}|s_{t+1}) - \pi_{jt}^{\text{new}}(\mathbf{u}_{t+1}|s_{t+1})|\big],
\end{aligned}
\tag{A.13}
$$

where $C = \max_{\mathbf{u}_{t+1}} |Q_{tot}^{\pi_{\text{old}}}(s_{t+1}, \mathbf{u}_{t+1}) - \sum_i \text{OPT}_i(Q_i^{\pi_{jt}^{\text{old}}}(o_{t+1}^i, u_{t+1}^i))|$. Due to the Pinsker's inequality [44], we have:

$$
\begin{aligned}
\epsilon \leq \frac{\gamma}{1-\gamma}|\epsilon_1 + \epsilon_2| &\leq \frac{\gamma}{1-\gamma}\mathbb{E}_{s_{t+1}\sim p}\big[C \cdot \sum_{\mathbf{u}_{t+1}} |\pi_{jt}^{\text{old}}(\mathbf{u}_{t+1}|s_{t+1}) - \pi_{jt}^{\text{new}}(\mathbf{u}_{t+1}|s_{t+1})|\big] \\
&\leq \frac{\gamma}{1-\gamma}\mathbb{E}_{s_{t+1}\sim p}\big[C \cdot \sqrt{2D_{KL}\big(\pi_{jt}^{\text{old}}(\cdot|s_{t+1})|\pi_{jt}^{\text{new}}(\cdot|s_{t+1})\big)}\big]
\end{aligned}
\tag{A.14}
$$

□

A.3 Hyper-network Structure

layer name	hyper-net_embed	embed_dim	num_layer
mixer	64	32	2
OPT	64	-	1

Table A.1: Hyperparameters used for hyper-networks.

Figure A.1 shows the overall network structure. Specifically, OPTs share the same network structure as the mixer network, but while the mixer network's input dimension is (batch_size, agent_num) and its output dimension is (batch_size, 1), both the input and output dimensions of OPTs are (batch_size, action_num). The network hyperparameters are as shown in the table below. In the SMAC-v2 experiment, we select a single layer for the OPT model to ensure fairness in comparison. Using two layers of OPT would significantly increase the number of parameters, making it less comparable to the baseline algorithms. While this may slightly reduce our method's performance, we adopt this choice for a fair comparison.

A.4 Comparison with Baselines

The following paragraphs present the differences between our method and those based on the Actor-Critic framework:

1. Using existing maximum entropy MARL algorithms may result in the loss of action order learned through credit assignment. Achieving the IGM condition, which outlines the order of local value functions, is crucial in credit assignment. Our work presents a value-based algorithm that introduces order-preserving transformations, ensuring that the order of the local actor matches the order of the local value functions. However, the commonly used maximum entropy MARL methods, employing the actor-critic framework, may not guarantee alignment between the actor's action order and that of the local Q-function due to approximation errors.

2. Actor-critic methods utilize the KL divergence as the loss function, while our approach uses the MSE loss. In the CTDE framework, all methods maintain a

more accurate centralized Q-function alongside multiple imprecise local policies. The core idea is to distill knowledge from the centralized Q-function to the local policies. Actor-critic methods typically minimize the KL divergence between the actor and the softmax critic, whereas our method minimizes the MSE loss between their logits. [25] pointed out that, in distillation contexts, MSE is a superior loss function compared to KL divergence.

3. AC methods can use global information when training the critic, resulting in higher model capacity. Our method introduces global information through hyper-networks and the mixer network, which results in lower model capacity.

4. In fact, it's challenging to theoretically prove that value-based methods are inherently superior to AC methods. However, through experiments, we have demonstrated that our method surpasses both FOP [63] and HASAC [32].

Below are comparisons with individual papers.

mSAC [45] introduces a method similar to MASAC. It employs the AC architecture, thus encountering issues 1 and 2 mentioned above, but does not fully utilize global information to train the critic, hence lacking the advantage noted in point 3.

FOP [63] shares a similar issue with [45], in that the AC framework it utilizes can actually train the critic with global state information. Furthermore, FOP employs a credit assignment mechanism similar to that of QPLEX [56]. [22] experimentally shown that QMIX outperforms QPLEX. Additionally, FOP's proof concept is that under the IGO conditions, if we locally move local policies towards local optimal policies, the distance between the joint policy and the optimal joint policy also decreases. In contrast, our proof approach first establishes that our method is equivalent to the single-agent SAC algorithm and then completes the proof using existing conclusions.

HASAC [32] is a CTCE version of MASAC, with its core contribution being the introduction of sequential decision-making. Therefore, we do not compare it here.

PAC [66] utilizes global information during centralized training, possessing the advantages noted in point 3. However, it still encounter issues 1 and 2 mentioned above.

A.5 Hyper-parameters

For all baseline algorithms, we implemented them using the corresponding open-source frameworks and chose the hyperparameters provided by these frameworks for replication. We used FOP by Zhang et al. and HASAC by Zhong et al.. Our method, ME-IGM is built on Hu et al.. The experiments were conducted on a computer equipped with 92 GB of RAM, a 40-core CPU, and a GeForce RTX 2080 Ti GPU. The following are the hyperparameters used in the experiments.

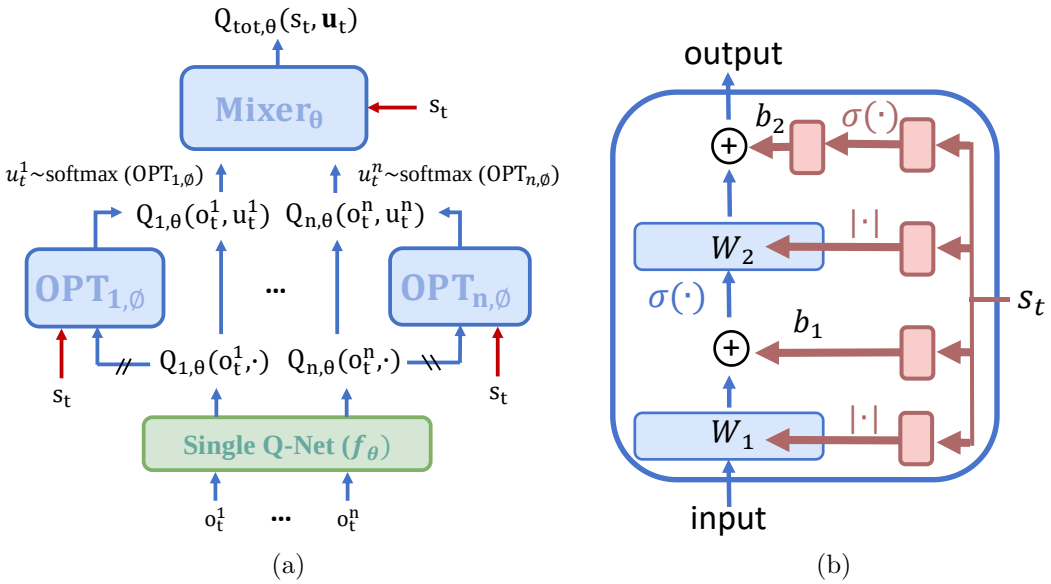


Figure A.1: (a) The overall network architecture. Single Q-Net takes local observations as input and outputs a distribution over local actions u^i, also denoted as local Q_i. Functions OPTs are order-preserving transformations, ensuring the input and output dimensions are identical and have the same argmax values. Both the input and output of the OPTs are k-dimensional, where k is the action dimension. The output of the OPT, after undergoing a softmax operation, becomes the policy, from which action u_t^i is sampled. The Mixer network takes local Q_i as input and outputs the global Q_{tot}. Green blocks utilize a recurrent neural network as the backbone, with all agents sharing parameters. For decentralized execution, only the green network is needed, and the maximum values of their output are selected. (b) The blue blocks' network structure. The red blocks represent the hyper-network, which takes the global state s_t as input and outputs the network's weights and biases. The weights are constrained to be non-negative. The activation function is denoted by $\sigma(\cdot)$. The mixer network takes an n-dimensional Q function as input and outputs a one-dimensional Q_{tot}.

Parameter Name	Value
n_rollout_threads	8
num_env_steps	10000000
warmup_steps	10000
train_interval	50
update_per_train	1
use_valuenorm	False
use_linear_lr_decay	False
use_proper_time_limits	True
hidden_sizes	[256, 256]
activation_func	relu
use_feature_normalization	True
final_activation_func	tanh
initialization_method	orthogonal
gain	0.01
lr	0.0003
critic_lr	0.0005
auto_α	True
α_lr	0.0003
γ	0.99
buffer_size	1000000
batch_size	1000
polyak	0.005
n_step	20
use_huber_loss	False
use_policy_active_masks	True
share_param	False
fixed_order	False

Table A.2: HASAC hyperparameters used for SMACv2. We use the hyperparameters for SMAC as specified in the original paper [32].

Parameter Name	Value
Runner	parallel
Batch Size Run	4
Buffer Size	5000
Batch Size	128
Optimizer	Adam
$t_{\max}$	1005000
Target Update Interval	200
Mac	n_mac
Agent	n_rnn
Agent Output Type	q
Learner	nq_learner
Mixer	qmix
Mixing Embed Dimension	32
Hyper-net Embed Dimension	64
Learning Rate	0.001
λ	0.4
$\bar{H}$ (zerg, protoss)	$0.24 \times$ num_ally
$\bar{H}$ (terran)	$0.32 \times$ num_ally
α Learning Rate	0.3

Table A.3: ME-QMIX hyperparameters used for SMACv2. We utilize the hyperparameters used in SMACv2 [14].

Parameter Name	Value
Runner	parallel
Batch Size Run	8
Buffer Size	5000
Batch Size	128
Optimizer	Adam
$t_{\max}$	1005000
Target Update Interval	200
Mac	basic_mac
Agent	rnn
Agent Output Type	q
Learner	dmaq_qatten_learner
Mixer	dmaq
Mixing Embed Dimension	32
Hyper-net Embed Dimension	64
Learning Rate	0.001
λ	0.6
$\bar{H}$ (zerg, protoss)	0.24 $\times$ num_ally
$\bar{H}$ (terran)	0.32 $\times$ num_ally
α Learning Rate	0.3

Table A.4: ME-QPLEX hyperparameters used for SMACv2. We utilize the hyperparameters used in SMACv2 [14].

Parameter Name	Value
Action Selector	Multinomial
ϵ-Start	1.0
ϵ-Finish	0.05
ϵ-Anneal Time	50000
Buffer Size	5000
$t_{\max}$	1005000
Target Update Interval	200
Agent Output Type	pi_logits
Learner	fop_learner
Head Number	4
Mixing Embed Dimension	32
Learning Rate	0.0005
Burn In Period	100
λ	0.4

Table A.5: FOP hyper-parameters used for SMACv2. We use the hyperparameters for SMAC as specified in the original paper [63].

A.6 Derivation of the Loss Function

The derivation is mainly from Peng et al. [41]. The goal of the gradient update step is to bring π closer to π'. One way to achieve this is by maximizing the following objective:

$$\eta(\pi) = \mathbb{E}_{s\sim d_\pi(\cdot), a\sim\pi(\cdot|s)} \left[\log \pi'(a|s) - \log \pi_k(a|s)\right], \tag{A.15}$$

where $d_\pi(s) = \sum_{t=0}^{\infty} \gamma^t p(s_t = s|\pi)$ represents the unnormalized discounted state distribution induced by the policy π, and $p(s_t = s|\pi)$ is the likelihood of the agent being in state s after following π for t time-steps. For simplicity, we use π_k in the appendix but π_0 in the main text; both denote the sampling policy.

In practice, to enhance sample efficiency, we aim to use $d_{\pi_k}(s)$ instead of $d_\pi(s)$ when estimating $\eta(\pi)$, where π_k is the policy used to sample new data. Therefore, similar to [41], we can employ $\hat{\eta}(\pi)$ as an approximation of $\eta(\pi)$ in practical implementations.

$$\hat{\eta}(\pi) = \sum_s d_{\pi_k}(s) \sum_a \pi(a|s) \left[\log \left(\frac{\pi'(a|s)}{\pi_k(a|s)}\right)\right]. \tag{A.16}$$

$\hat{\eta}(\pi)$ is a reasonable estimator of $\eta(\pi)$ only when π and π_k are sufficiently similar. Therefore, rather than directly solving the optimization problem $\max_\pi \eta(\pi)$, we can instead reformulate it as the following optimization problem:

$$\begin{aligned} \underset{\pi}{\operatorname{argmax}} & \sum_s d_{\pi_k}(s) \sum_a \pi(a|s) \left[\log \left(\frac{\pi'(a|s)}{\pi_k(a|s)}\right)\right] \\ \text{s.t.} & \sum_s d_{\pi_k}(s) D_{KL}\left(\pi(\cdot|s)||\pi_k(\cdot|s)\right) \leq \epsilon \\ & \sum_a \pi(a|s) = 1, \forall s. \end{aligned} \tag{A.17}$$

By applying the method of Lagrange multipliers to the constrained optimization problem, the optimal policy can be expressed as follows:

$$\pi^*(a|s) = \frac{1}{Z(s)} \pi_k(a|s) \exp\left(\frac{1}{\alpha} \log \frac{\pi'(a|s)}{\pi_k(a|s)}\right), \tag{A.18}$$

where $Z(s) = \sum_{a'} \pi_k(a'|s) \exp\left(\frac{1}{\alpha} \log \frac{\pi'(a|s)}{\pi_k(a|s)}\right)$ normalizes the optimal policy and α is the Lagrange multiplier.

Finally, we need to project π^* back onto the manifold of parameterized policies. This can be achieved by optimizing the following objective:

$$
\begin{aligned}
&\operatorname*{argmin}_{\pi} \mathbb{E}_{s\sim d_{\pi_0}(s)} \left[D_{KL}\left(\pi^*(\cdot|s)||\pi(\cdot|s)\right)\right] \\
=&\operatorname*{argmin}_{\pi} \mathbb{E}_{s\sim d_{\pi_0}(s)} \left[D_{KL}\left(\frac{1}{Z(s)} \cdot \pi_k(\cdot|s) \odot \exp(\frac{1}{\alpha} \log \frac{\pi'_{k+1}(\cdot|s)}{\pi_k(\cdot|s)})||\pi(\cdot|s)\right)\right] \\
=&\operatorname*{argmin}_{\pi} \mathbb{E}_{s\sim d_{\pi_0}(s)} \left[D_{KL}\left(\frac{1}{Z(s)} \cdot \exp((1-\frac{1}{\alpha}) \log \pi_k(\cdot|s) + \frac{1}{\alpha} \log \pi'_{k+1}(\cdot|s))||\pi(\cdot|s)\right)\right] \\
=&\operatorname*{argmin}_{\pi} \mathbb{E}_{s\sim d_{\pi_0}(s)} \left[D_{KL}\left(\frac{1}{Z(s)} \cdot \exp((1-\alpha^*) \log \pi_k(\cdot|s) + \alpha^* \log \pi'_{k+1}(\cdot|s))||\pi(\cdot|s)\right)\right] \\
&,
\end{aligned}
\tag{A.19}
$$

where $\odot$ represents the element-wise multiplication. In practice, we treat $\alpha^* = 1/\alpha$ as a hyper-parameter that controls the size of the trust region.

A.7 Proof of Theorem 3.4.1

Proof. Let $\pi_0 : \mathcal{S} \times \mathcal{A} \to (0, 1)$ and $\pi : \mathcal{S} \times \mathcal{A} \to (0, 1)$ be any two policies in a finite MDP $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is unknown. According to Eq. (3.2), we have: $(\forall s \in \mathcal{S}, a \in \mathcal{A})$

$$
\pi(a|s) = \frac{\pi_0(a|s) \exp\left(A_r^{\pi_0}(s, a)/\beta\right)}{Z(s)} \Rightarrow A_r^{\pi_0}(s, a) = \beta \left(\log \frac{\pi(a|s)}{\pi_0(a|s)} + \log Z(s)\right)
\tag{A.20}
$$

Here, $Z(s)$ is a normalization factor and is a function of $\pi(\cdot|s)$ and $\pi_0(\cdot|s)$ since $\sum_a A_r^{\pi_0}(s, a)/\beta = \sum_a \log \frac{\pi(a|s)}{\pi_0(a|s)} + \log Z(s) = h(\pi_0)$. Thus, given π and π_0, we can acquire the advantage function $A_r^{\pi_0}(s, a)$ for all (s, a).

Next, we derive the relation between $A_r^{\pi_0}$ and r under the maximum entropy RL framework. According to [67, 68], the advantage function $A_r^{\pi_0}$, value function V^{π_0}, and Q function Q^{π_0} of a policy π_0 are related as follows: (t is a time step and

$(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$.)

$$A_r^{\pi_0}(s_t, a_t) = Q^{\pi_0}(s_t, a_t) - V^{\pi_0}(s_t)$$
$$V^{\pi_0}(s_t) = \mathbb{E}_{a_t \sim \pi_0(\cdot|s_t)} \left[Q^{\pi_0}(s_t, a_t) - \beta \log \pi_0(a_t|s_t) \right] = \sum_{a_t} \pi_0(a_t|s_t) Q^{\pi_0}(s_t, a_t) + \beta \mathcal{H}(\pi_0(\cdot|s_t)) \tag{A.21}$$

where $\mathcal{H}(\pi_0(\cdot|s_t))$ is the policy entropy at state s_t. **Thus,** $\mathbb{E}_{a_t \sim \pi_0(\cdot|s_t)}[A_r^{\pi_0}(s_t, a_t)] = -\beta \mathcal{H}(\pi_0(\cdot|s_t))$ **and** $h(\pi_0) = -\mathcal{H}(\pi_0(\cdot|s_t))$**.** The Q function satisfies the Soft Bellman Equation:

$$\begin{aligned} Q^{\pi_0}(s_t, a_t) =& r(s_t, a_t) + \gamma \beta \mathbb{E}_{s_{t+1} \sim P(\cdot|s_t, a_t)}[\mathcal{H}(\pi_0(\cdot|s_{t+1}))] \\ &+ \gamma \mathbb{E}_{s_{t+1} \sim P(\cdot|s_t, a_t), a_{t+1} \sim \pi_0(\cdot|s_{t+1})}[Q^{\pi_0}(s_{t+1}, a_{t+1})] \end{aligned} \tag{A.22}$$

Denote $f(s_t, a_t) = \gamma \beta \mathbb{E}_{s_{t+1} \sim P(\cdot|s_t, a_t)}[\mathcal{H}(\pi_0(\cdot|s_{t+1}))]$, which can be calculated based on P and π_0, then we have:

$$Q^{\pi_0} = (I - \gamma P^{\pi_0})^{-1}(r + f) \tag{A.23}$$

where Q^{π_0}, r, and f are vectors of size $|\mathcal{S}||\mathcal{A}|$, I is an identity matrix, $P^{\pi_0} \in [0, 1]^{|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}||\mathcal{A}|}$ is the transition kernel between any two state-action pairs defined with P and π_0. $I - \gamma P^{\pi_0}$ is invertible, according to Corollary 1.5 in [2]. Combining Eq. (A.21) and Eq. (A.23), we have the following linear system:

$$A_r^{\pi_0} + g = (I' - \Pi_0)(I - \gamma P^{\pi})^{-1}(r + f) \tag{A.24}$$

Here, $A_r^{\pi_0}$, $\log \pi_0$, r, f are vectors of $A_r^{\pi_0}(s, a)$, $\log \pi_0(s, a)$, $r(s, a)$, $f(s, a)$, respectively; I' is an identity matrix; g is a vector of size $|\mathcal{S}||\mathcal{A}|$, where each $g(s, a) = \beta \mathcal{H}(\pi_0(\cdot|s))$; Π_0 is a block diagonal matrix of size $|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}||\mathcal{A}|$, composed of $|\mathcal{S}|$ submatrices, each of size $|\mathcal{A}| \times |\mathcal{A}|$. Specifically, the submatrix corresponding to s is $\Pi_0(s) = [\pi_0(\cdot|s) \cdots \pi_0(\cdot|s)]^T = \mathbf{1}\pi_0(\cdot|s)^T$, where $\mathbf{1}$ is an all-one vector. $I'(s) - \Pi_0(s)$ **is a diagonal block of** $I' - \Pi_0$ **corresponding to state** s **and has a rank of** $|\mathcal{A}| - 1$**.** This is because: (1) $\text{rank}(I'(s) - \Pi_0(s) + \Pi_0(s)) = |\mathcal{A}| \leq \text{rank}(I'(s) - \Pi_0(s)) + \text{rank}(\Pi_0(s)) = \text{rank}(I'(s) - \Pi_0(s)) + 1 \Rightarrow \text{rank}(I'(s) - \Pi_0(s)) \geq |\mathcal{A}| - 1$ and (2) $\mathbf{1}$ is an eigenvector of $I'(s) - \Pi_0(s)$ corresponding to an eigenvalue of 0.

Next, we proof that the linear system $Cx = b$ corresponding to Eq. (A.24) is consistent, where $C = (I' - \Pi_0)(I - \gamma P^\pi)^{-1}$, $x = r + f$, and $b = A_r^{\pi_0} + g$. Note that C, f, and b are defined with γ, β, P, π_0, and π and so are known. We can apply elementary row operations, represented by a matrix D, to the augmented matrix $[C \mid b]$. **In particular, D is also a block diagonal matrix, composed of $|S|$ submatrices, each of size $|\mathcal{A}| \times |\mathcal{A}|$. The submatrix corresponding to state s is an identity matrix with the first row replaced by $\pi(\cdot|s)^T$. Notably, the 1st, $(|\mathcal{A}|+1)$-th, $(2|\mathcal{A}|+1)$-th, $\cdots$, $((|\mathcal{S}|-1)|\mathcal{A}|+1)$-th rows of $D(I'-\Pi_0)$ (and so DC) and Db are all 0.** This can be easily proved based on the following facts:

$$\pi_0(a|s)(1-\pi_0(a|s)) - \pi_0(a|s)\left(\sum_{a'\neq a}\pi_0(a'|s)\right) = 0 \Rightarrow \pi_0(\cdot|s)^T(I'(s) - \Pi_0(s)) = \mathbf{0}^T;$$
$$\mathbb{E}_{a\sim\pi_0(\cdot|s)}[A_r^{\pi_0}(s,a)] = -\beta\mathcal{H}(\pi_0(\cdot|s)) \Rightarrow \mathbb{E}_{a\sim\pi_0(\cdot|s)}b(s) = 0. \tag{A.25}$$

Eliminating these rows in $[DC \mid Db]$, we can get a new augmented matrix $[\tilde{C} \mid \tilde{b}]$ where $\tilde{C}$ has a full row rank (based on the facts that $\text{rank}(I'(s) - \Pi_0(s)) = |\mathcal{A}| - 1$, $\forall s$ and $I - \gamma P^\pi$ is invertible). Thus, the original linear system $Cx = b$ (i.e., Eq. (A.24)) is consistent, according to the Rouché-Capelli theorem. □

A.8 Comparison of Training Times

Table A.6: Training time (in minutes) required for each algorithm on different tasks, along with the GPU setup used.

Method	goto	pickup	pick_up_seq_go_to	open	GPU Type
RWR	72	69	232	227	NVIDIA A40 × 2
RICOL	96	95	594	585	NVIDIA A40 × 2
PPO (10M)	258	257	234	198	GeForce RTX 2080 Ti × 1
PPO (3B)	1440	1440	1440	1440	NVIDIA A40 × 4

A.9 Environments

A.9.1 1D Key-Door Environment

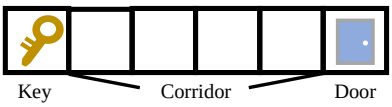


Figure A.2: An illustration for the 1D Key-Door scenario.

As illustrated in Figure A.2, the environment consists of a 1D grid world where the agent can traverse along the x-axis. The key is located at the leftmost grid cell, and the door is positioned at the rightmost grid cell. Initially, the agent starts without holding the key, aiming to move left to retrieve the key and subsequently move right to unlock the door. The corridor length used in the paper is 10. The available action includes {move left, move right, pick up the key, unlock the door}. The prompt can be found in Figure A.3 and Figure A.4.

For any given policy, we can compute the corresponding transition matrix P, where the entry in the i-th row and j-th column represents the probability of the agent transitioning from state s_i to state s_j in a single step. The ground truth value function V can then be computed as:$V = (I - \gamma P)^{-1}R$, where γ is the discount factor and R is the reward function. Once the value function is determined, and given that the environment is deterministic, we can compute the advantage function using the following formula:$A(s, a) = V(s) - r(s, a) - \gamma V(s')$ where s' is the next state resulting from applying action a in state s, and $r(s, a)$ is the immediate reward received when transitioning from s to s' by taking action a.

System Prompt

You are a helpful navigation agent.
Please thoroughly review your goal, the available actions, objects around you, and any advice from experts, if available, and respond to the question strictly adhering to this specified format: ACTION: [your_answer]

User Prompt

Your goal is to open the locked door. You can choose to: **move left, move right, pick up the key, or unlock the door**. You see the locked door 200 meters away on your right. You don't have the key. You see a key on the ground around you. After reviewing the rules of the environment, your goal and objects around you, what do you do next to achieve the goal?

Figure A.3: The prompt used for the LLM policy in the 1D Key-Door scenario.

System Prompt

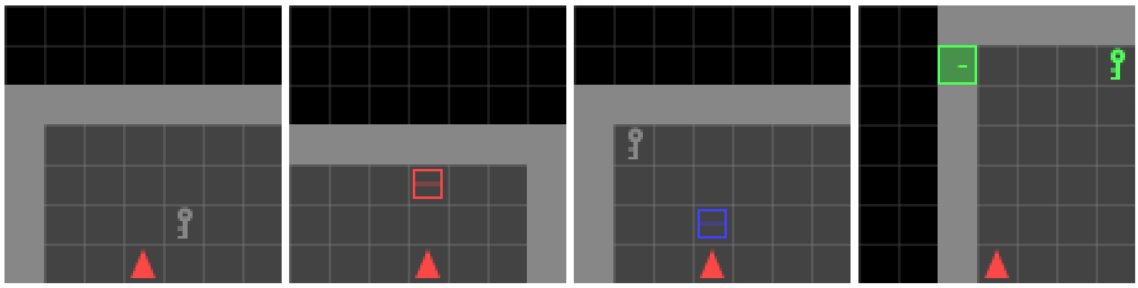
I will provide you with a trajectory starting from step 0 and the state at step 0.
Firstly, you have to analyze the task, the goal, and the current state, generate a rough plan.
Then, based on this trajectory, you need to evaluate whether the decision you made at step 0 was correct.
Additionally, if given another opportunity, how would you modify it, if needed?
Provide concise verbal feedback as a guideline for others new to this task and facing the same state in the future.
If the policy is generally correct, you can reply with "No modifications needed."
After providing your explanation, output your final feedback by strictly following this format: "Verbal Feedback: [your_answer]"

User Prompt

Trajectory/state (skip due to the space limit)

Figure A.4: The prompt used for the LLM reflector in the 1D Key-Door scenario.

A.9.2 BabyAI Environment



(a) goto (b) pickup (c) pick_up_seq_go_to (d) open

Figure A.5: Screenshots of four BabyAI scenarios, where the agent is partial-observable.

We evaluate our algorithm in the BabyAI environment [11], a 2D grid world where the player navigates an agent to accomplish specified tasks. We use BALROG's [38] implementation of the environment. We test our algorithm on four scenarios: *goto*, *pickup*, *pick_up_seq_go_to*, and *open*. As shown in Figure A.5, the player directs the agent (a red triangle) toward a specified local object or manipulate it. The environment is partially observable, with the agent having a 7x7 window of visibility in front of it, viewed from an egocentric perspective. The agent can perform six actions: Turn Left, Turn Right, Move Forward, Pickup, Drop, and Toggle. Both input and output for the tasks are presented in text form. The prompt can be found in Figure A.6 and Figure A.7. We did two editions to the BabyAI environment. Firstly, following Ren et al. [47], we modify the policy prompt so that its output is constrained to a single character between A and F, with each character corresponding to a specific action. This ensures that the probability of each action is not biased by differences in token length. Secondly, we set the maximum episode length to 16 for the *goto* and *pickup* tasks, and 32 for the *pick_up_seq_go_to* and *open* tasks. This early truncation improves sample efficiency. All baseline algorithms are evaluated using the updated environment to ensure a fair comparison.

System Prompt

You are an agent playing a simple navigation game. Your goal is to {mission}.
The following are the possible actions you can take in the game, followed by a short description of each action:
"A": "turn to the left",
"B": "turn to the right",
"C": "take one step forward",
"D": "pick up the object one step in front of you",
"E": "drop the object that you are holding",
"F": "manipulate the object in front of you",
In a moment I will present you an observation. Tips:
- Once the desired object you want to interact or pickup in front of you, you can use the 'toggle' action to interact with it.
- It doesn't make sense to repeat the same action over and over if the observation doesn't change.
- answer the alphanumerical action, not the description.

PLAY!

User Prompt

Current Observation:
a wall 3 steps forward
a wall 1 step right

You always have to output one of the above actions at a time and no other text. You always have to output an action until the episode terminates.

Figure A.6: The prompt used for the LLM policy in all the BabyAI scenarios.

A.10 RICL can Identify Critical States in Sequential Decision-Making

Here, we demonstrate that RICL assigns higher credits to critical states in the environment. Critical states are defined as those where policy adjustments lead to greater performance gains. For the sampling policy π_0 and an improved policy π, $D_{KL}(\pi(\cdot|s)||\pi_0(\cdot|s))$ serves as a measure of how significantly the policy needs to be adjusted at state s to achieve policy improvement (from $\pi_0(\cdot|s)$ to $\pi(\cdot|s)$). Thus, it can be used as a criterion for identifying critical states. Specifically, π can be π_{gt} or π_{RICL} (as defined in Section 3.6.2) to identify the critical states with the ground-truth policy or the RICL-updated policy, respectively.

Figure A.8 depicts the (normalized) critical score (i.e., $v^{\pi} = D_{KL}(\pi(\cdot|s)||\pi_0(\cdot|s))$) in the 1D Key-Door scenario using LLaMA-3.1-8B-Instruct as π_0. The yellow curve shows two peaks in the critical score $v^{\pi_{\text{gt}}}$. The right peak corresponds to the state

– picking up the key, which is critical in the 1D Key-Door scenario. The left peak corresponds to a state where the expert decides to move toward the key. Due to imperfections in π_0, this state also becomes critical. The blue curve in Figure 3.2 represents $v^{\pi_{\text{RICL}}}$. The peaks of the blue curve align with those of the yellow curve (i.e., the ground truth), demonstrating that RICL effectively identifies critical states.

We further compare RICL with methods that explicitly utilize LLMs for credit assignment [43] (represented by the green curve in Figure A.8). Specifically, these methods prompt an LLM with a trajectory and ask it to identify the critical state within the sequence. We evaluate this approach on 1000 distinct trajectories, using the frequency with which a state s is labeled as critical to compute its critical score. As shown by the green curve in Figure A.8, this method can only identify the "pick-up-the-key" state as critical. It fails to detect the left peak of the yellow curve. This limitation arises because identifying this critical state requires knowledge of the sampling policy π_0, which is provided to but overlooked by this baseline approach.

A.11 Comparisons between Retroformer and RICOL

First, Retroformer [58] fine-tunes a reflector LLM to generate prompts for a fixed actor LLM, whereas our method fine-tunes an actor LLM using guidance from a fixed reflector LLM. The objectives of the two approaches are fundamentally orthogonal. Second, Retroformer relies on trajectory-level sparse rewards for LLM fine-tuning, whereas RICL leverages step-level dense training signals. Specifically, Retroformer requires rolling out two trajectories – one with reflector-generated hints (τ_1) and one without (τ_2) – and computes the reward as the return difference between the two, which is then used to fine-tune the reflector. In contrast, our method applies step-level dense supervision by estimating the advantage function at each time step. This is computed based on the change in policy before and after the in-context update: $\log \frac{\pi_{\text{in-context updated}}(a|s)}{\pi_{\text{sampling}}(a|s)}$. **In short, for an episode of length n, RICL collects n supervised signals from n environment steps, while Retroformer requires $2n$ environment steps to obtain just one supervision signal.**

Retroformer estimates $\Delta(s_0, \text{feedback}) = V^{\pi_{\text{in-context updated}}}(s_0) - V^{\pi_{\text{sampling}}}(s_0)$, i.e.,

the value difference between two policies, using the difference in trajectory returns: $Reward(\tau_1) - Reward(\tau_2)$. In contrast, our method only requires estimating the advantage function under a single policy, π_{sampling}, defined as $A(s,a) = Q^{\pi_{\text{sampling}}}(s,a) - V^{\pi_{\text{sampling}}}(s)$. This is generally more sample-efficient. The following experiments use the Monte Carlo method to estimate both quantities. The table below reports the mean squared error (MSE) between the estimated and ground truth values of $\Delta(s_0, \text{feedback})$ and $A(s,a)$.

Table A.7: Approximation errors vs. number of sample trajectories.

Number of Sample Trajectories	**1,000**	**10,000**	**100,000**	**1,000,000**
MSE on $A(s,a)$	0.1287	0.0492	0.0492	0.0454
MSE on $\Delta(s_0, \text{feedback})$	0.2763	0.2788	0.2713	0.2641

The results indicate that, under Monte Carlo sampling, Retroformer requires significantly more trajectories than our method to achieve accurate value estimations. Further, as shown in Figure 3.2 of the main paper, our actual sampling strategy (i.e., RICL) achieves over 100× greater sample efficiency compared to the Monte Carlo baseline. Altogether, these findings demonstarte that our approach is substantially more sample-efficient than Retroformer in terms of value estimation.

System Prompt

You are an agent playing a simple navigation game. Your goal is to {mission}.
The following are the possible actions you can take in the game, followed by a short description of each action:
"A": "turn to the left",
"B": "turn to the right",
"C": "take one step forward",
"D": "pick up the object one step in front of you",
"E": "drop the object that you are holding",
"F": "manipulate the object in front of you",

User Prompt

You are a coach that can provide feedback to the agent.
I will provide you the observation at time step t and the action taken by the agent at time step t.
Then I will provide you the trajectory after that.
Your task is to carefully analyze the trajectory and reflect on the agent's decision at time step t.
If the agent were to face the same situation again (observation at time step t), what advice would you give to better achieve the goal?
Provide your feedback as a concise and informative string.

The observation at time step t is: {o_t}
The action taken by the agent at time step t is: {a_t}
The trajectory after that is: {traj}

Provide a concise, specific verbal feedback that can help the agent improve its performance when it encounters state t again to help it achieve the goal.
Breifly analyze the agent's decision at time step t and the consequences of the action then provide the feedback.

You should always follow the format:
Your goal: [your goal]
state t: [the state at time step t]
action t: [the action taken by the agent at time step t]
state t+1: [the state at time step t+1]
analyze: [The agent took action t in state t and transitioned to state t+1. Analyze the decision and its consequences, does the action help the agent achieve the goal or make progress?]
Conclusion: [Based on the helpfulness, will you suggest the agent to take action t again, why?]
Feedback: [your concise verbal feedback that can help the agent improve its performance]

Figure A.7: The prompt used for the LLM reflector in all the BabyAI scenarios. Only the **Feedback** part is extracted and used for RICL's in-context policy updates.

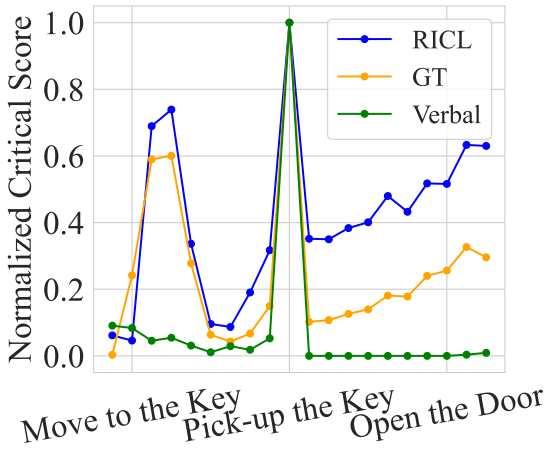


Figure A.8: Comparison of critical states identified by RICL and verbal credit assignment. The x-axis represents a sequence of states: the agent first moves toward the key over 9 steps, picks up the key, then moves toward the door over another 9 steps, and finally opens the door. The y-axis indicates the score reflecting how critical each state is, where GT denotes the ground truth. The results show that both algorithms correctly identify the "pick up the key" action as a critical state-action pair. However, only RICL additionally identifies some of the earlier states in the "move to the key" phase as critical.

Bibliography

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 1

[2] Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 32:96, 2019. A.7

[3] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024. 3.2

[4] Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In *International conference on machine learning*, pages 151–160. PMLR, 2019. 2.1

[5] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017. 3.1

[6] The Viet Bui, Tien Anh Mai, and Thanh Hong Nguyen. Mimicking to dominate: Imitation learning strategies for success in multiagent games. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=06JRFVK88O. 2.7.2

[7] Meng Cao, Lei Shu, Lei Yu, Yun Zhu, Nevan Wichers, Yinxiao Liu, and Lei Meng. Drlc: Reinforcement learning with dense rewards from llm critic. *arXiv preprint arXiv:2401.07382*, 2024. 3.1

[8] Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning*, pages 3676–3713. PMLR, 2023. 3.6.1

[9] Shreyas Chaudhari, Pranjal Aggarwal, Vishvak Murahari, Tanmay Rajpurohit, Ashwin Kalyan, Karthik Narasimhan, Ameet Deshpande, and Bruno Castro da Silva. Rlhf deciphered: A critical analysis of reinforcement learning from human feedback for llms. *arXiv preprint arXiv:2404.08555*, 2024. 3.1

[10] Zhipeng Chen, Kun Zhou, Wayne Xin Zhao, Junchen Wan, Fuzheng Zhang, Di Zhang, and Ji-Rong Wen. Improving large language models via fine-grained reinforcement learning with minimum editing constraint. *arXiv preprint arXiv:2401.06081*, 2024. (document), 3.2, 3.1

[11] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*, 2018. A.9.2

[12] Djork-Arné Clevert. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. 2.5.1

[13] Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. Rlhf workflow: From reward modeling to online rlhf. *arXiv preprint arXiv:2405.07863*, 2024. 3.1

[14] Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob N Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2212.07489*, 2022. (document), 1.1, 2.1, 2.7.2, A.3, A.4

[15] Benjamin Eysenbach and Sergey Levine. Maximum entropy rl (provably) solves some robust rl problems. *arXiv preprint arXiv:2103.06257*, 2021. 2.1, 2.2.2

[16] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 1.1, 2.2.1

[17] Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*, 2025. 3.6.4

[18] Fenggen Guo and Zizhao Wu. Learning maximum entropy policies with qmix in cooperative marl. In *2022 IEEE 2nd International Conference on Electronic Technology, Communication and Information (ICETCI)*, pages 357–361. IEEE, 2022. 2.2.2, 2.4

[19] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 2.2.2

[20] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al.

Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018. 2.1, 2.2.2, 2.6

[21] Zichen He, Lu Dong, Chunwei Song, and Changyin Sun. Multiagent soft actor-critic based hybrid motion planner for mobile robots. *IEEE transactions on neural networks and learning systems*, 2022. 2.2.2

[22] Jian Hu, Siyang Jiang, Seth Austin Harding, Haibin Wu, and Shih-wei Liao. Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2102.03479*, 2021. A.4, A.5

[23] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2023. 3.2

[24] Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36: 39648–39677, 2023. 3.2

[25] Taehyeon Kim, Jaehoon Oh, NakYil Kim, Sangwook Cho, and Se-Young Yun. Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation. *arXiv preprint arXiv:2105.08919*, 2021. A.4

[26] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023. 1

[27] Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. *arXiv preprint arXiv:2109.11251*, 2021. 2.2.1

[28] Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. *arXiv preprint arXiv:2303.00001*, 2023. 3.2

[29] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018. 2.1, 2.2.2, 3.3

[30] Hao Li, Xue Yang, Zhaokai Wang, Xizhou Zhu, Jie Zhou, Yu Qiao, Xiaogang Wang, Hongsheng Li, Lewei Lu, and Jifeng Dai. Auto mc-reward: Automated dense reward design with large language models for minecraft. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16426–16435, 2024. 3.2

[31] Fanqi Lin, Shiyu Huang, Tim Pearce, Wenze Chen, and Wei-Wei Tu. Tizero: Mastering multi-agent football with curriculum learning and self-play. *arXiv preprint arXiv:2302.07515*, 2023. 2.2.1

[32] Jiarong Liu, Yifan Zhong, Siyi Hu, Haobo Fu, Qiang Fu, Xiaojun Chang, and Yaodong Yang. Maximum entropy heterogeneous-agent mirror learning. *arXiv preprint arXiv:2306.10715*, 2023. (document), 2.2.2, 2.7.2, A.4, A.2

[33] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017. 2.2.1

[34] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023. 3.2

[35] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024. (document), 3.2, 3.1

[36] Theo X Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. Demystifying gpt self-repair for code generation. *CoRR*, 2023. 3.2

[37] Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016. 2.3.1

[38] Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wolczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, et al. Balrog: Benchmarking agentic llm and vlm reasoning on games. *arXiv preprint arXiv:2411.13543*, 2024. A.9.2

[39] Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization. *arXiv preprint arXiv:2404.19733*, 2024. (document), 3.2, 3.1

[40] Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems*, 34:12208–12221, 2021. 2.2.1

[41] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019. 3.5, 3.5.1, A.6, A.6

[42] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In Zoubin Ghahramani, editor, *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, volume 227 of *ACM International Conference Proceeding Series*, pages 745–750. ACM, 2007. doi:

10.1145/1273496.1273590. URL https://doi.org/10.1145/1273496.1273590. 3.6.5

[43] Eduardo Pignatelli, Johan Ferret, Tim Rockäschel, Edward Grefenstette, Davide Paglieri, Samuel Coward, and Laura Toni. Assessing the zero-shot capabilities of llms for action evaluation in rl. *arXiv preprint arXiv:2409.12798*, 2024. A.10

[44] Mark S Pinsker. Information and information stability of random variables and processes. *Holden-Day*, 1964. A.2.2

[45] Yuan Pu, Shaochen Wang, Rui Yang, Xin Yao, and Bin Li. Decomposed soft actor-critic method for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2104.06655*, 2021. 2.2.2, A.4

[46] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research*, 21(1):7234–7284, 2020. 1.1, 2.1, 2.2.1, 2.3.2, 2.7.1

[47] Allen Z Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, et al. Robots that ask for help: Uncertainty alignment for large language model planners. *arXiv preprint arXiv:2307.01928*, 2023. A.9.2

[48] John Schulman. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*, 2015. 3.5.1

[49] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015. 3.2

[50] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024. (document), 3.2, 3.1, 3.4.3, 3.6.4

[51] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*, pages 5887–5896. PMLR, 2019. 2.2.1, 2.7.1

[52] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017. 3.1

[53] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-

agent learning. *arXiv preprint arXiv:1706.05296*, 2017. 2.2.1

[54] Karthik Valmeekam, Matthew Marquez, and Subbarao Kambhampati. Investigating the effectiveness of self-critiquing in LLMs solving planning tasks. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023. URL https://openreview.net/forum?id=gGQfkybOKL. 3.2

[55] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019. 1

[56] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020. 2.2.1, 2.7.1, A.4

[57] Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025. 3.2

[58] Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, et al. Retroformer: Retrospective large language agents with policy gradient optimization. *arXiv preprint arXiv:2308.02151*, 2023. (document), 3.2, 3.1, 3.6.2, A.11

[59] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022. 2.2.1

[60] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023. 3.2

[61] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022. (document), 3.2, 3.1

[62] Miaomiao Zhang, Wei Tong, Guangyu Zhu, Xin Xu, and Edmond Q Wu. Sqix: Qmix algorithm activated by general softmax operator for cooperative multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2024. 2.2.2

[63] Tianhao Zhang, Yueheng Li, Chen Wang, Guangming Xie, and Zongqing Lu. Fop: Factorizing optimal joint policy of maximum-entropy multi-agent reinforcement

learning. In *International Conference on Machine Learning*, pages 12491–12500. PMLR, 2021. (document), 2.2.2, 2.4, 2.4, 2.7.1, A.4, A.5, A.5

[64] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642, 2024. 3.2

[65] Yifan Zhong, Jakub Grudzien Kuba, Xidong Feng, Siyi Hu, Jiaming Ji, and Yaodong Yang. Heterogeneous-agent reinforcement learning, 2023. 2.2.1, A.5

[66] Hanhan Zhou, Tian Lan, and Vaneet Aggarwal. Pac: Assisted value factorization with counterfactual predictions in multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 35:15757–15769, 2022. A.4

[67] Brian D. Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Carnegie Mellon University, USA, 2010. 3.3, A.7

[68] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 1433–1438. AAAI Press, 2008. A.7