Automated Policy Shaping for Multi-Agent Reinforcement Learning in Pursuit-Evasion Games via Retrieval-Augmented LLMs

Guo Cheng
School of Automation
Southeast University
Nanjing, China
220242203@seu.edu.cn

Haichuan Ye
School of Automation
Southeast University
Nanjing, China
yhc_oc@163.com

Wenwen Jia
Department of Mathematics
Hohai University
Nanjing, China
20250044@hhu.edu.cn

Jian Liu
School of Automation
Southeast University
Nanjing, China
bkliujian@163.com

Yongbao Wu
School of Automation
Southeast University
Nanjing, China
yongbaowu199211@163.com

Lei Xue*
School of Automation
Southeast University
Nanjing, China
leixue@seu.edu.cn

***Abstract*—The paper proposes an automated framework for shaping policies in multi-agent reinforcement learning (MARL) environments, particularly in the pursuit-evasion game (PEG). The framework addresses the challenges of traditional manual reward design, which is time-consuming, labor-intensive, and inflexible. The core innovation is a tight integration of Retrieval-Augmented Generation (RAG) and Large Language Models (LLMs). By leveraging RAG to provide domain-specific knowledge, the framework can generate semantically correct learning signals that guide agents to master the complex dynamics of PEG. Crucially, it integrates RAG with a closed-loop, data-driven evolutionary process that autonomously evolves these guidance signals based on agent performance. This evolutionary mechanism discovers sophisticated emergent behaviors through the systematic optimization of reward components and structure. Finally, empirical validation in complex PEG scenarios demonstrates that our framework significantly outperforms established baselines.**

***Index Terms*—Large Language Models (LLMs), multi-agent reinforcement learning (MARL), Retrieval-Augmented Generation (RAG), pursuit-evasion game (PEG).**

I. Introduction

The PEG, as a fundamental problem in engineering systems, has received widespread attention from academia over the past few decades. Its applications are diverse. In the military domain, these include missile guidance and interception, as well as aircraft operational strategies [1], [2]. In the civilian sphere, applications extend to criminal pursuit, collision avoidance systems, wireless network deployment, and the development of intelligent transportation infrastructures [3]–[6].

Conventional methodologies primarily rely on control-theoretic design frameworks, which necessitate rigorous modeling of the target system and its operational context, coupled with carefully engineered control policies to attain performance optimization [7]–[10]. However, these approaches are often task-specific, inflexible, and require specialized expertise. These constraints collectively result in prolonged development cycles [11]. As multi-agent systems grow in complexity, these limitations become more pronounced, necessitating more scalable and adaptable solutions.

Recent advances in MARL have opened up new possibilities for multi-agent systems. Within the MARL framework, the control strategy for each agent undergoes automatic optimization during the training phase. Crucially, this process eliminates the need for precise models of the scenario or the agents themselves. Instead, optimization is driven primarily by a suitably designed incentive structure. This shifts the core challenge from engineering control algorithms to defining effective learning objectives that can guide policy optimization. In [12], it employed MARL with topology-constrained communication (ring/line) for cooperative pursuit, rewarding proximity/contact and penalizing evader speed. However, it did not consider whether the pursuers had a cooperative relationship. In [13], it designed a geometrically complex incircle-based incentive structure for Multi-Agent Deep Deterministic Policy Gradient (MADDPG) to enable encircling behaviors. A multi-agent pursuit-evasion strategy leveraging Apollonius partition and a role-based backbone QMIX algorithm was proposed in [14]. The incentive structure centered on pursuers cooperatively minimizing the evader's dominant region area. However, their dependency on triangular formations or hand-tuned parameters limits adaptability to varying swarm sizes. It is noteworthy that these diverse approaches all encounter significant difficulties in crafting artificial reward functions for PEG. This process demands a deep understanding of the task's fundamental logic and is often characterized by extensive iterative refinement. Consequently, efficiency is frequently low and the incentive structure design problem becomes even more challenging in complex scenarios.

Consequently, MARL researchers are increasingly exploring methods to automate reward design. For instance, AIRS [15] employs a multi-armed bandit mechanism driven by task return estimates to adaptively select intrinsic exploration bonuses from a predefined pool, automating intrinsic reward shaping. However, a limitation of AIRS is that persistent exploration can hinder training convergence. Alternative approaches leverage demonstrations. Adversarial Inverse Reinforcement Learning (AIRL) [16] learns robust rewards invariant to environmental dynamics by structuring the reward function into a state-only term and a dynamics-compensating shaping term. DemPref [17] combines high-level demonstrations with fine-grained preference queries, using information gain maximization to actively generate interpretable questions and minimize human burden. A significant drawback common to IRL-based methods like AIRL and DemPref, however, is their dependence on substantial demonstration data to yield an effective reward function. Despite these advancements, existing methods often fail to adequately address the unique challenges inherent in multi-agent systems.

The rise of LLM has established a new paradigm for designing multi-agent systems. Leveraging their rich knowledge, LLMs have found applications across diverse aspects of robotics. Recent work has integrated the knowledge and skills of LLMs with reinforcement learning (RL) to enhance RL efficiency. In [18], the authors propose a self-refined LLM framework for automated reward function design. Eureka [19] leverages LLMs to generate and evolve optimized reward code, effectively solving complex low-level tasks like dexterous pen manipulation. Building on this, Auto MC-Reward [20] innovatively employs LLMs to produce executable rewards for intricate scenarios within Minecraft. However, these methods are currently confined to single-agent scenarios. They typically address only simple tasks. The domain of multi-agent collaboration remains largely unexplored. Compared to single-agent systems, multi-agent systems involve agents possessing local perspectives and require coordination with neighbors. Given the complexity of such tasks, LLMs lack task-specific knowledge and have limited reasoning capabilities, making them prone to issues like hallucination, which undermines their effectiveness in specialized domains like PEG.

Building upon the seminal research previously discussed, this paper proposes a new method for automated policy shaping in MARL, leveraging a RAG-driven LLM to autonomously establish effective learning objectives for complex multi-agent systems in PEG. The main contributions of this paper are as follows:

- This paper presents a new framework that automates the crafting of learning incentives via a closed-loop data-driven evolutionary process. This system transforms reward design from a static manual task into a dynamic self-correcting cycle. By continuously leveraging agent performance feedback, the framework iteratively refines the incentive structure and its associated reward magnitudes, thereby enabling the autonomous discovery of complex emergent strategies.
- The framework's efficacy is rooted in two technical innovations that ensure the generation and validation of high-quality reward functions. The integration of a RAG module grounds the LLM in domain-specific knowledge to mitigate hallucinations and ensure semantic correctness. Concurrently, a Reward Analyzer automates the quality control process by quantitatively evaluating functions against structured metrics such as sparsity, stability, and convergence. This process provides robust and essential feedback to guide the evolutionary loop.
- Extensive empirical validation demonstrates that our framework consistently outperforms established baselines across diverse multi-agent scenarios. The resulting strategies achieve higher capture success rates and reduced capture times, thereby validating the framework's efficiency and its capacity to foster advanced cooperative behaviors.

The paper is meticulously organized into distinct sections. Some basic preparations are presented in Section II. Section III designs the proposed RAG-driven LLM method, detailing its workflow components. Section IV introduces the experimental setup and results, which validate the method in different PEG environments. Section V outlines conclusions and the subsequent plan for future research directions.

II. Preliminaries

A. System model

In this study, the pursuit-evasion game (PEG) is considered involving N pursuers (blue agents) and M evaders (red agents) in a two-dimensional Euclidean plane, $\mathbb{R}^2$. Each agent is modeled as a point mass with distinct kinematic constraints and the game unfolds in continuous time, as illustrated in Fig. 1. To rigorously describe agent kinematics and inter-team spatial relationships, we establish a coordinate system with its origin at point O, enabling precise analysis of relative motion dynamics.

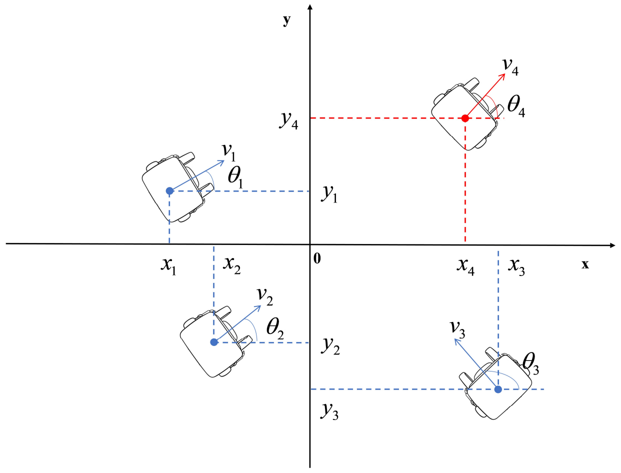


Fig. 1. The environment of multi-agent pursuit-evasion game.

Let $\mathcal{P} = \{1, 2, \ldots, N\}$ denote the set of N pursuers and $\mathcal{E} = \{N+1, N+2, \ldots, N+M\}$ denote the set of M evaders.

Each agent $i \in \mathcal{P} \cup \mathcal{E}$ is characterized by its position in the plane, denoted by $\mathbf{p}_i(t) = (x_i(t), y_i(t)) \in \mathbb{R}^2$ at time $t \geq 0$. The motion of each agent is governed by a single-integrator kinematic model, where the velocity is determined by a control input comprising a speed (displacement) and a heading angle.

For an agent i, the dynamics are described by:

$$\begin{cases} \dot{\mathbf{p}}_i(t) = v_i(t) \begin{bmatrix} \cos\theta_i(t) \\ \sin\theta_i(t) \end{bmatrix}, \\ v_i(t) \in [0, v_{i,max}], \end{cases} \tag{1}$$

where $v_i(t)$ is the speed, $v_{i,max}$ is the maximum speed of the agent, $\theta_i(t)$ is the heading angle.

The objective of the evaders is to maximize their survival time by avoiding capture, while the pursuers aim to minimize the time to capture by coordinating their actions effectively.

B. Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

To address the challenges of learning in multi-agent environments, this work utilizes the MADDPG algorithm. In MADDPG [21], each agent i maintains an actor network μ_{ϕ_i} with parameters ϕ_i and a critic network Q_{Θ_i} with parameters Θ_i, where ϕ_i and Θ_i are updated through distinct optimization processes.

$$\nabla_{\phi_i} J(\phi_i) = \mathbb{E}_{s,a\sim\mathcal{D}} \left[\nabla_{\phi_i} \mu_{\phi_i}(o_i) \nabla_{a_i} Q_{\Theta_i}(s,a) \big|_{a_i=\mu_{\phi_i}(o_i)} \right], \tag{2}$$

where $\mathcal{D}$ denotes the experience replay buffer and o_i denotes the local observation of agent i. Crucially, only local observations o_i are required for action selection during execution, preserving decentralization.

The MADDPG algorithm employs the centralized training with decentralized execution (CTDE) paradigm, where critics utilize global information during training while actors execute locally during deployment. This framework stabilizes learning by mitigating non-stationarity in multi-agent systems.

The critic is updated by minimizing the loss:

$$\mathcal{L}(\Theta_i) = \mathbb{E}_{(s,a,r_i,s')} \left[\left(Q_{\Theta_i}(s,a) - y_i \right)^2 \right], \tag{3}$$

where target value y_i is computed as:

$$y_i = r_i + \gamma \bar{Q}_{\Theta_i'}(s',a') \big|_{a'=\bar{\mu}_{\phi_i'}(o_i)}, \tag{4}$$

with $\bar{\mu}_{\phi_i'}$ and $\bar{Q}_{\Theta_i'}$ being target networks.

C. Problem statement

The primary challenge in applying MARL to complex, dynamic systems like the PEG lies in the design of an effective reward function. The quality of the reward function is the principal factor guiding agent learning and directly dictates the efficacy and strategic sophistication of the emergent behaviors.

Current methodologies predominantly depend on manual, expert-driven design. This approach is not only labor-intensive and time-consuming but also produces rigid, inflexible reward structures that struggle to adapt to the evolving complexities of multi-agent interaction. Therefore, this paper addresses the fundamental problem of automating the reward design process in a scalable and adaptive manner. We focus on developing an automated framework capable of generating effective reward functions by leveraging modern generative AI. The ultimate objective is to overcome the limitations of manual design, thereby enhancing the strategic coordination and effectiveness of agents in complex MARL environments.

III. METHOD

The workflow consists of four components: Retrieval-Augmented Generation (RAG), Reward Designer, Reward Analyzer and Evolution. Given the environmental information and the task description, RAG matches relevant knowledge to generate an initial prompt. The Reward Designer proposes a reward function through initial prompt input. The Reward Analyzer first verifies if the proposed function is self-consistent and meets all formatting requirements. Subsequently, this validated function is used to train the agents in the environment. In order to improve the designed reward function based on experience, we summarize the training indicators of the agents after a period of training, then provide feedback to the Reward Designer. Meanwhile, The Reward Analyzer reviews the indicators to determine whether the evolution process can be stopped. Fig. 2 provides the overview of the workflow.

A. Retrieval-Augmented Generation

The RAG component is designed to bridge the gap between the general knowledge of an LLM and the domain-specific requirements of the PEG. It achieves this by integrating the generative capabilities of the LLM with a retrieval mechanism that accesses a curated knowledge base of domain-specific information. This process ensures the generation of reward functions that are not only syntactically correct but also semantically aligned with the complex dynamics of PEG.

The RAG workflow is structured into three distinct stages, as depicted in Fig. 2: (1) knowledge base preparation, (2) query-based information retrieval, and (3) prompt integration for generation.

1) Knowledge Base Preparation: The foundation of the RAG process is a specialized knowledge base constructed from relevant technical documents, such as academic papers and research reports on reinforcement learning and pursuit-evasion strategies. For this study, a single, comprehensive document serves as the primary source of domain expertise. To facilitate efficient information retrieval, the document is segmented into smaller, semantically coherent chunks. These chunks are then indexed and converted into vector embeddings, which allows for rapid, similarity-based retrieval of the most contextually relevant information. The granularity of these chunks ensures that the retrieval process is both precise and effective.

2) Information Retrieval: The retrieval process is initiated by a high-level user query, such as, "Design a reward function for a 3v1 pursuit-evasion game." The system employs a similarity search to match this query against the vectorized chunks in the knowledge base, retrieving the top-k most relevant pieces of information. These retrieved chunks, which typically

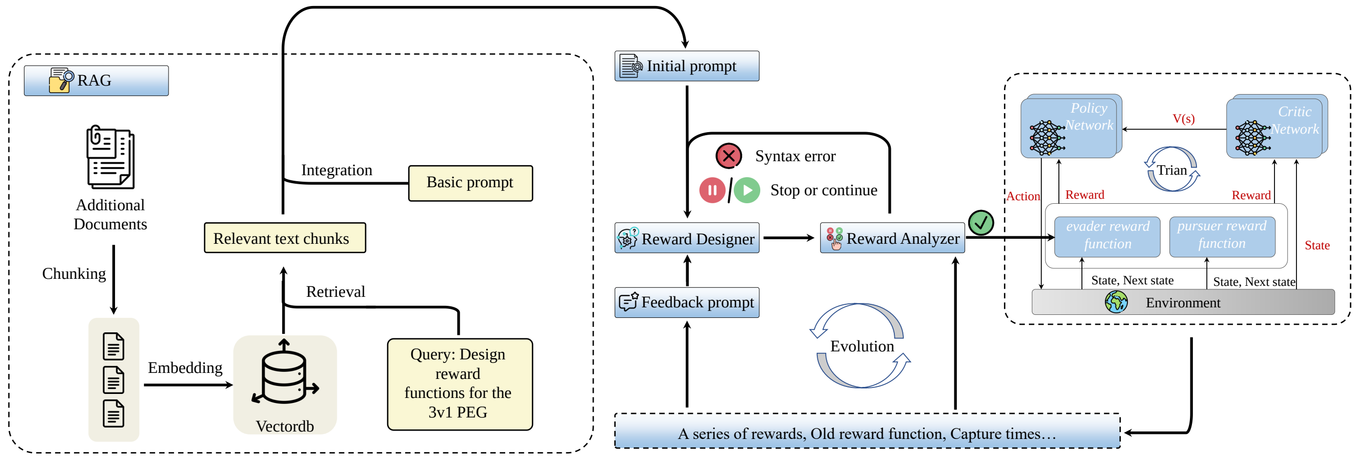


Fig. 2. Overview of the automated reward function design workflow. This workflow consists of four core components: Retrieval-Augmented Generation (RAG), Reward Designer, Reward Analyzer, and Evolution.

contain strategic insights or mathematical formulations pertinent to PEG, are then concatenated to form a rich, contextual foundation for the generation phase.

3) Prompt Integration and Generation: In the final stage, the retrieved contextual information is integrated into a basic prompt. This augmented prompt, which specifies the required function signature, strategic logic, and any performance feedback from previous iterations, is then provided to the LLM. The LLM leverages this context-rich input to generate the reward function, ensuring its alignment with the strategic objectives of the game. This generated function is then deployed within the reinforcement learning environment to guide agent behavior. This ensures that the reward functions are not only grounded in expert knowledge but also responsive to iterative refinement, offering a scalable and effective solution for enhancing agent performance in complex multi-agent systems.

B. Reward Designer

The Reward Designer utilizes the enriched initial prompt from the RAG component to propose executable reward functions. As shown in Fig. 2, the Reward Designer processes the initial prompt as input and generates a reward function tailored to the specified task, such as designing rewards for the evader or pursuer in the 3v1 PEG.

Referring to the Chain-of-Thought (CoT) method [22], we require the LLM to first describe its design approach, including the rationale and details of the reward function design. These ideas are written as comments in the code, which enhances the interpretability of the function and aligns it with the task objectives. This is crucial in PEG, as subtle interactions between agents require careful design of reward structures. This method not only aids reward designers in initializing and updating the function by referring to textual ideas but also helps reviewers assess the soundness of the code.

C. Reward Analyzer

In practice, it is difficult for an LLM to generate a relatively complete reward function in the beginning. There may be errors in understanding parameter formats and data types (syntax errors), failure to consider game-specific information, or misunderstanding of tasks (semantic errors), etc. To eliminate the above errors, which are not easy to find, we design an LLM-based Reward Analyzer to automatically review the designed reward function. In addition to checking for related errors, Reward Analyzer is also required to check the quality of the reward function to further assess the progress of its evolution. Specifically, we require reward reviewers to conduct a three-step review to determine if the current code meets the requirements. If the review fails, continue with the evolutionary process.

Sparsity assessment: Reward Analyzer analyzes the feedback frequency and distribution of the reward function. If rewards are too sparse, for example only available upon reaching the finish line, learning may be slow. If they are too dense, learning may overfit. Sparsity can be quantified by the frequency of non-zero rewards or by the entropy of the reward distribution. Let r_t be the reward received at time step t. The frequency of non-zero rewards over a trajectory of length T can be defined as:

$$F(r_t) = \frac{\sum_{t=1}^{T} \mathbb{I}(r_t \neq 0)}{T}, \tag{5}$$

where $\mathbb{I}(\cdot)$ is the indicator function.

Stability analysis: Reward Analyzer checks whether the reward function leads to a stable learning process. It evaluates learning stability by monitoring reward variance to detect sudden fluctuations that hinder convergence. The variance of rewards within an episode can be calculated as:

$$\text{Var}(r_t) = \frac{1}{T-1} \sum_{t=1}^{T} (r_t - \bar{r})^2, \tag{6}$$

where $\bar{r}$ is the mean reward over T steps.

Reward function convergence: To ensure a consistent and interpretable measure of convergence, we established a fixed benchmark evaluation set, $\mathcal{D}_\text{b}$, comprising K representative state-action pairs. This set was carefully curated to cover a wide spectrum of critical scenarios, including initial states, coordinated encirclements, and common failure modes. This benchmark remains static throughout the entire evolutionary process. The convergence of the reward function between generation e and $e-1$ is then quantified by the Mean Squared Error (MSE) computed exclusively on this fixed set:

$$\text{MSE}(R_e, R_{e-1}) = \frac{1}{K} \sum_{(s,a)\in\mathcal{D}_\text{b}} \big(R_e(s,a) - R_{e-1}(s,a)\big)^2. \tag{7}$$

Using a static benchmark provides a stable yardstick for comparison, decoupling the measurement of functional change from policy-induced distributional shifts. The evolution is terminated when the MSE on this benchmark falls below a predefined threshold ϵ. While this approach risks overfitting to the benchmark, it offers unparalleled consistency for tracking the stabilization of the reward function's core logic.

D. Evolution

The Evolution component focuses on iteratively enhancing the reward function based on the agent's environmental performance. After a designated training period, we summarize critical indicators, including reward sequences, the previous reward function, and capture frequency. These quantitative measures assess reward function effectiveness (see Fig. 2).

A Feedback Prompt consolidates these metrics for the Reward Designer. The Reward Designer iteratively updates the reward function based on this feedback prompt and the Reward Analyzer's signal, initiating a continuous design evaluation training cycle. This data-driven evolution ensures continual reward optimization, improving PEG agent behaviors through empirical refinement.

Fig. 3 delineates this core framework. This figure provides a detailed explanation of the workflow aimed at optimizing the reward function of MARL in PEG with three pursuers and one evader.

Iteration 0 (initial): This represents the initial reward function generated by the RAG-driven LLM. Its primary driver is a distance reward ($r_{0,d}$), which encourages direct pursuit by rewarding the pursuers for closing the distance to the evader. Simultaneously, a coordinated punishment term ($r_{0,c}$) promotes basic team formation by penalizing large angular differences between pursuers to encourage coordinated movement. Furthermore, a rudimentary and imprecise surrounding reward ($r_{0,s}$) offers a basic incentive for encirclement by rewarding specific pursuit angles. A simple capture reward term is also included to provide a strong terminal bonus. At this stage, the logic is straightforward, focusing on basic pursuit and rudimentary coordination without designing any advanced strategic concepts.

Iteration 1&2 (evolution): Following the initial performance assessment, the framework enters a crucial evolutionary phase. The reward function is systematically adjusted and enriched based on agent performance data. This iterative refinement process involves both the tuning of existing component weights (Weight Adjustment), adjusting the existing structure (Reward Component Adjustment) and the introduction of new reward terms (New Component Added) to address the strategic limitations of the initial formulation. For instance, the surrounding rewards ($r_{1,s}, r_{2,s}$) component is refined from a simple form towards a more precise geometric objective. The geometric constraint terms are becoming explicit, laying the foundation for subsequent bounding strategies.

Iteration 3 (final): This is the final, converged reward function. Its key innovation lies in the "surrounding reward" component, which has evolved into the formula:

$$\lambda_3 = -2\sum_{i<j} \left| \cos|\theta_{t+1,i} - \theta_{t+1,j}| - \cos\frac{2\pi}{3} \right|.$$

This expression explicitly rewards the three pursuers for forming 120-degree ($\frac{2\pi}{3}$ radians) angles relative to each other, which corresponds to the optimal geometry for a stable triangular encirclement. This precise mathematical objective transforms the agents' behavior from simple pursuit to strategic containment, leading to a significant improvement in coordination and capture efficiency.

IV. Experiment

This section presents a series of simulations to validate the effectiveness of the proposed model. Comparative analyses were also conducted using reward functions from different stages to show the advantages of iterative evolution. A thorough evaluation was performed, testing the algorithm across different PEG environments to assess its ability to generate reward functions and solve tasks. DeepSeek V3 is used as the backbone LLM for all LLM-based reward design methods.

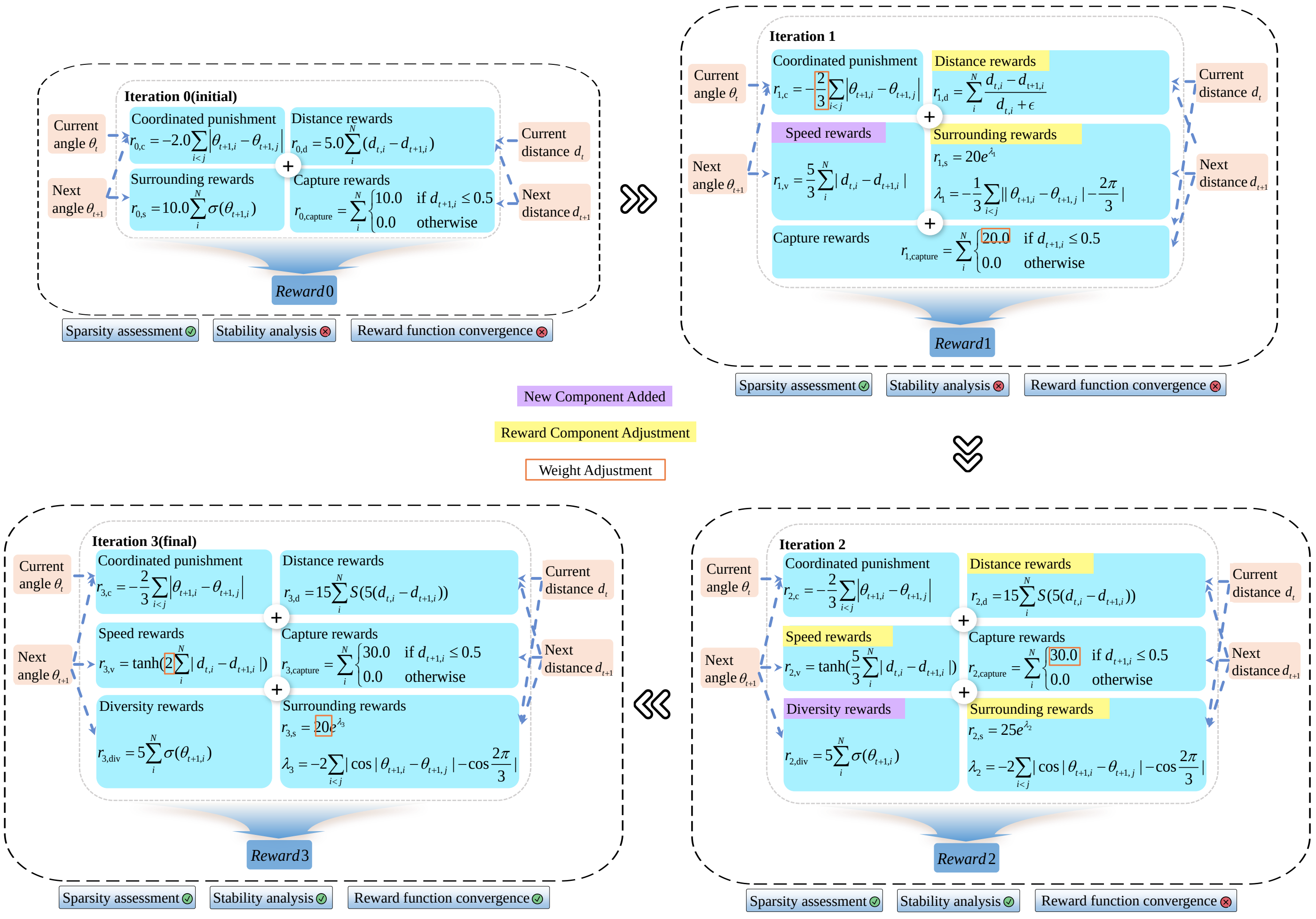
A. Environments

To evaluate how well the algorithm generates reward functions and solves tasks, we employ diverse PEG scenarios. These environments are designed to test the algorithm across varying levels of complexity and agent interactions. The specific settings are detailed as follows:

3v1 Scenario: This scenario features three pursuers and one evader. The pursuers must collaborate to capture the single evader. This scenario tests the algorithm's ability to design reward functions that promote coordination among a group of pursuers against an evader.

4v2 Scenario: Similar to the 3v1 Scenario, this scenario includes four pursuers and two evaders. The increased number of evaders introduces additional complexity, requiring the reward functions to account for multiple targets and more intricate multi-agent dynamics.

In the above scenarios, the evaders adopt two different strategies to test the robustness and adaptability of the generated reward function:



Iteration 0(initial)
Current angle θ_t
Next angle θ_{t+1}
Coordinated punishment
$r_{0,c} = -2.0\sum_{i<j}^{N}|\theta_{t+1,i} - \theta_{t+1,j}|$
Distance rewards
$r_{0,d} = 5.0\sum_{i}^{N}(d_{t,i} - d_{t+1,i})$
Surrounding rewards
$r_{0,s} = 10.0\sum_{i}^{N}\sigma(\theta_{t+1,i})$
Capture rewards
$r_{0,capture} = \sum_{i}^{N}\begin{cases}10.0 & \text{if } d_{t+1,i} \le 0.5\\ 0.0 & \text{otherwise}\end{cases}$
Current distance d_t
Next distance d_{t+1}
Reward0
Sparsity assessment ✓
Stability analysis ✗
Reward function convergence ✗
Iteration 1
Coordinated punishment
$r_{1,c} = -\frac{2}{3}\sum_{i<j}^{N}|\theta_{t+1,i} - \theta_{t+1,j}|$
Distance rewards
$r_{1,d} = \sum_{i}^{N}\frac{d_{t,i} - d_{t+1,i}}{d_{t,i} + \epsilon}$
Speed rewards
$r_{1,v} = \frac{5}{3}\sum_{i}^{N}|d_{t,i} - d_{t+1,i}|$
Surrounding rewards
$r_{1,s} = 20e^{\lambda_1}$
$\lambda_1 = -\frac{1}{3}\sum_{i<j}||\theta_{t+1,i} - \theta_{t+1,j}| - \frac{2\pi}{3}|$
Capture rewards
$r_{1,capture} = \sum_{i}^{N}\begin{cases}20.0 & \text{if } d_{t+1,i} \le 0.5\\ 0.0 & \text{otherwise}\end{cases}$
Reward1
Sparsity assessment ✓
Stability analysis ✗
Reward function convergence ✗
New Component Added
Reward Component Adjustment
Weight Adjustment
Iteration 2
Coordinated punishment
$r_{2,c} = -\frac{2}{3}\sum_{i<j}^{N}|\theta_{t+1,i} - \theta_{t+1,j}|$
Distance rewards
$r_{2,d} = 15\sum_{i}^{N}S(5(d_{t,i} - d_{t+1,i}))$
Speed rewards
$r_{2,v} = \tanh(\frac{5}{3}\sum_{i}^{N}|d_{t,i} - d_{t+1,i}|)$
Capture rewards
$r_{2,capture} = \sum_{i}^{N}\begin{cases}30.0 & \text{if } d_{t+1,i} \le 0.5\\ 0.0 & \text{otherwise}\end{cases}$
Diversity rewards
$r_{2,div} = 5\sum_{i}^{N}\sigma(\theta_{t+1,i})$
Surrounding rewards
$r_{2,s} = 25e^{\lambda_2}$
$\lambda_2 = -2\sum_{i<j}|\cos|\theta_{t+1,i} - \theta_{t+1,j}| - \cos\frac{2\pi}{3}|$
Reward2
Sparsity assessment ✓
Stability analysis ✓
Reward function convergence ✗
Iteration 3(final)
Coordinated punishment
$r_{3,c} = -\frac{2}{3}\sum_{i<j}^{N}|\theta_{t+1,i} - \theta_{t+1,j}|$
Distance rewards
$r_{3,d} = 15\sum_{i}^{N}S(5(d_{t,i} - d_{t+1,i}))$
Speed rewards
$r_{3,v} = \tanh(2\sum_{i}^{N}|d_{t,i} - d_{t+1,i}|)$
Capture rewards
$r_{3,capture} = \sum_{i}^{N}\begin{cases}30.0 & \text{if } d_{t+1,i} \le 0.5\\ 0.0 & \text{otherwise}\end{cases}$
Diversity rewards
$r_{3,div} = 5\sum_{i}^{N}\sigma(\theta_{t+1,i})$
Surrounding rewards
$r_{3,s} = 20e^{\lambda_3}$
$\lambda_3 = -2\sum_{i<j}|\cos|\theta_{t+1,i} - \theta_{t+1,j}| - \cos\frac{2\pi}{3}|$
Reward3
Sparsity assessment ✓
Stability analysis ✓
Reward function convergence ✓

Fig. 3. The detailed evolution process of the pursuer reward function in a 3v1 PEG scenario.

Nonlinear Path Strategy: This strategy causes the evaders to move along a complex and tortuous path, with an unpredictable trajectory, posing a challenge to the pursuer. This strategy aims to extend the pursuit time and test whether the reward function can prompt pursuers to demonstrate predictive ability and strategic position adjustment. Evader j generates complex motion trajectories by introducing time-varying nonlinear disturbances in the basic direction $\theta_{j,base}$. The mathematical expression of this strategy is as follows:

$$\theta_j(t) = \theta_{j,base} + \frac{\pi}{4}\delta(t) + \Phi(t), \tag{8}$$

where $\delta(t) = 0.6\sin(0.5t) + 0.4\cos(1.2t) + 0.3\sin(3.0t)\cos(2.5t)$ and $\Phi(t) = 0.1\sin(0.3t)\pi$ are the nonlinear offsets.

Greedy Strategy: In this strategy, each evader directly moves away from the nearest pursuer to maximize the immediate distance from the threat. For evader e, solve $\arg\min_{p\in\mathcal{P}} \|\mathbf{p}_e - \mathbf{p}_p\|_2$ to identify the nearest pursuer. The escape vector aligns with the gradient $\nabla_{\mathbf{p}_e}\|\mathbf{p}_e - \mathbf{p}_p\|_2$, and displacement occurs at maximum speed. This strategy provides single-step optimal responses in continuous action space. This strategy is simple but effective in certain scenarios, forcing pursuers to constantly adjust their paths and challenging whether the reward function can incentivize efficient pursuit methods.

B. Baselines

To assess the effectiveness of the proposed LLM-based reward function design approach, the study compares it against multiple baseline methods within the MARL framework, specifically using the MADDPG algorithm as the foundational MARL algorithm. The baselines are described as follows:

Greedy Strategy: This baseline employs a heuristic approach. The greedy pursuit strategy directs each pursuer to move directly towards its assigned evader at maximum velocity. The displacement vector is computed as the direction from the pursuer's current position to the evader's position,

TABLE I: Performance Comparison of Different Methods in 3v1 and 4v2 scenarios

Method	3v1				4v2			
	Nonlinear		Greedy Strategy		Nonlinear		Greedy Strategy	
	Success Rate	Avg. Time(s)	Success Rate	Avg. Time(s)	Success Rate	Avg. Time(s)	Success Rate	Avg. Time(s)
Greedy Strategy	100%	17.00	100%	18.00	100%	22.00	100%	21.30
Apollonius Partitions+MADDPG	97%	15.13	100%	19.14	81%	37.22	100%	15.78
Apollonius Partitions+QMIX	73%	37.06	87%	24.51	76%	33.70	63%	41.97
RAG+DeepSeekV3+Evolution	**100%**	**8.00**	**100%**	**12.00**	**100%**	**20.49**	**100%**	**14.62**
RAG+DeepSeek V3	100%	17.00	100%	13.00	96%	29.63	100%	17.10
DeepSeek V3+Evolution	100%	24.00	100%	29.39	78%	21.22	99%	19.14

with movement magnitude capped at the pursuer's maximum speed. The action of each pursuer p is defined as

$$\mathbf{a}_p = v_{\max} \cdot \frac{\mathbf{p}_e - \mathbf{p}_p}{\|\mathbf{p}_e - \mathbf{p}_p\|_2}. \tag{9}$$

where $e \in \mathcal{E}$.

Apollonius Partitions-based MARL: This more advanced baseline incorporates Apollonius Partitions, a geometric method, into the reward function design for MARL algorithms. Apollonius Partitions divide the environment into distinct dominant regions based on the relative positions and speeds of the pursuers and evaders. By rewarding pursuers for actions that expand their collective dominant area, this approach embeds domain-specific geometric information to enhance agent coordination. We evaluate the principle of reward shaping on two different MARL backbone algorithms (MADDPG [13] and QMIX [14]), representing the most advanced expert driven reward design method currently available.

Table I provides a comprehensive empirical evaluation of the proposed RAG-driven LLM framework against several baseline across two distinct multi-agent PEG environments: 3v1 scenario and a more complex 4v2 scenario. The performance of each method was rigorously tested against two different evader strategies: a sophisticated Non-linear Path Strategy and a reactive Greedy Strategy. The primary evaluation metrics are success rate and average steps. Success rate is the proportion of successful captures over 1000 trials, where a capture is defined as the pursuer being within a Euclidean distance of 0.1 from the evader within 50 steps. Average steps is the mean number of time steps required for a successful capture.

The quantitative data presented in Table I unequivocally demonstrates the effectiveness and efficiency of the proposed RAG+DeepSeekV3+Evolution framework. Across different environments, the LLM-based MADDPG consistently achieves the highest success rate and the lowest average steps, highlighting the effectiveness of the RAG-driven LLM approach in designing reward functions. The Greedy Strategy achieves reliable captures but exhibits inefficiency through higher step counts. MADDPG with Apollonius Partitions provides an advanced baseline, but our method outperforms it. This advantage comes from the LLM's ability to incorporate domain-specific knowledge into reward design. Across both 3v1 and 4v2 environments and against varied evader strategies, this method not only achieves perfect success rates but also consistently minimizes the time to capture.

C. Ablation studies

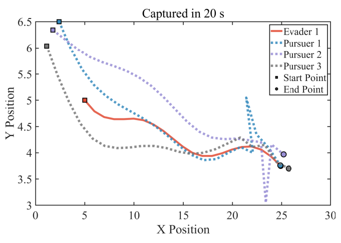
RAG+DeepSeek V3: Generates an initial reward function with an RAG-driven LLM but omits evolutionary refinement. It evaluates the initial quality without iterative adaptation.

DeepSeek V3+Evolution: Uses the LLM with the evolutionary loop but excludes RAG. Reward functions are generated without the curated domain knowledge base.

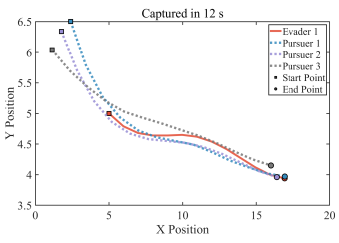
Results indicate that removing either RAG or the evolutionary process reduces performance relative to the fully integrated system, as shown in Table I.

To further elucidate the iterative effects of the reward function design, we perform a detailed analysis of the agent trajectories, correlating the visual improvements in strategy with the mathematical evolution of the reward function. The progression from initial to final iterations reveals how the framework autonomously discovers and encodes complex cooperative behaviors.

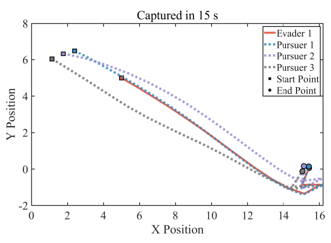
The strategic impact of the evolutionary process is most evident in the 3v1 PEG scenario. By linking the pursuer trajectories in Fig. 4 to the evolving reward function in Fig. 3, we can observe a clear progression from suboptimal to highly coordinated behavior. Initially, the pursuit strategy seen in Figs. 4a and 4c is suboptimal, characterized by circuitous paths and a lack of effective coordination. This behavior is a direct consequence of the rudimentary reward function from Iteration 0, which relies on a simple distance-closing heuristic ($r_{0,d} = 5.0 \sum (d_{t,i} - d_{t+1,i})$) and a naive angle-based penalty for coordination. These components are insufficient to foster a sophisticated encirclement strategy. In stark contrast, the trajectories in Figs. 4b and 4d, which use the final evolved reward, exhibit highly efficient and coordinated behavior, culminating in a triangular encirclement pattern that quickly traps evaders. This strategic leap is grounded mathematically



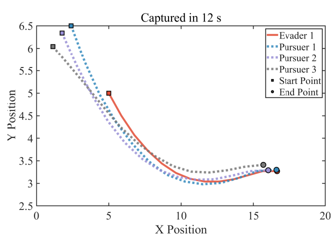
(a) pursuer reward function at Iteration 0 (with evaders under a Nonlinear Path Strategy).



(b) pursuer reward function at the final iteration (with evaders under a Nonlinear Path Strategy).

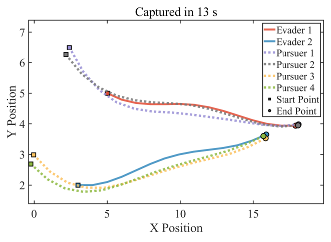


(c) pursuer reward function at Iteration 0 (with evaders under a Greedy Strategy).

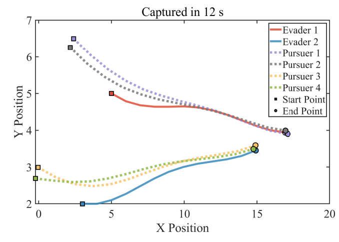


(d) pursuer reward function at the final iteration (with evaders under a Greedy Strategy).

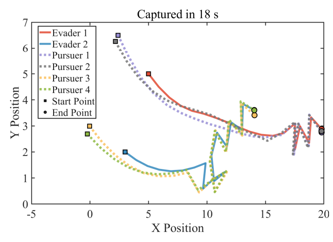
Fig. 4. Trajectory of pursuers and evaders in 3v1 PEG.



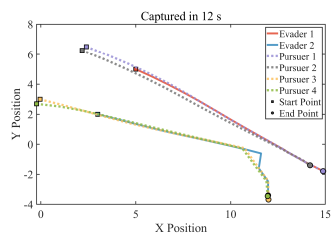
(a) pursuer reward function at Iteration 0 (with evaders under a Nonlinear Path Strategy).



(b) pursuer reward function at the final iteration (with evaders under a Nonlinear Path Strategy).



(c) pursuer reward function at Iteration 0 (with evaders under a Greedy Strategy).



(d) pursuer reward function at the final iteration (with evaders under a Greedy Strategy).

Fig. 5. Trajectory of pursuers and evaders in 4v2 PEG.

in the refinement of the Surrounding-reward component:

$$\begin{aligned} r_{3,s} &= 20e^{\lambda_3}, \\ \lambda_3 &= -2\sum_{i<j}\left|\cos\left|\theta_{t+1,i}-\theta_{t+1,j}\right| - \cos\frac{2\pi}{3}\right|. \end{aligned} \tag{10}$$

This final term explicitly encodes the optimal 120° ($2\pi/3$ radians) angular separation for a three-pursuer encirclement, transforming the agents' objective from simple pursuit to strategic containment and leading to significantly faster capture times.

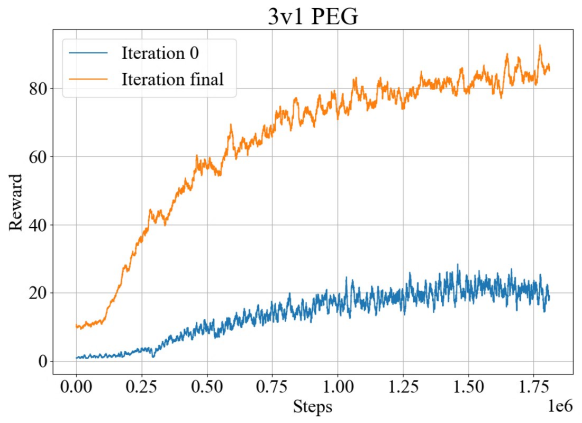


Fig. 6. Training reward curve in 3v1 PEG scenario: iteration 0 vs. iteration final.

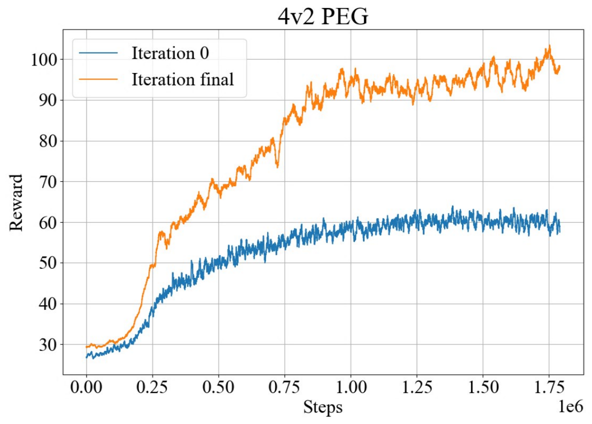


Fig. 7. Training reward curve in 4v2 PEG scenario: iteration 0 vs. iteration final.

Furthermore, the framework's adaptability is validated in the more complex 4v2 scenario, as shown in Fig. 5. Fig. 5a shows the initial reward function with evaders using a Nonlinear Path Strategy, where the four pursuers exhibit inefficient coordination, leading to extended pursuit times. Fig. 5b illustrates improved performance at Iteration final under the same nonlinear strategy. Pursuers effectively split efforts to encircle both evaders, reducing capture time as evidenced by lower average steps. Figs. 5c-5d show similar improvements against a greedy evader strategy, with pursuers quickly trapping the evader.

The trajectory analyses in Figs. 4-5 support the numerical results, visually confirming the efficacy of our evolutionary policy shaping process. The transition from Iteration 0 to Iteration final demonstrates significant improvements in pursuer coordination and capture efficiency, validated by the reduced average steps in the LLM-based MADDPG. Moreover, the ability to handle both nonlinear and greedy evader strategies highlights the adaptability of the final pursuit strategy.

To quantitatively assess the impact of the evolutionary process on learning efficiency, we analyzed the training reward curves for both the initial (Iteration 0) and final (Iteration final) reward functions. As depicted in Figs. 6-7, we can observe that the proposed method achieves the highest return for all scenarios. The learning curve corresponding to the final iteration shows a steeper and more consistent upward trend, converging at a much higher reward level than the curve for Iteration 0. This indicates that the refined reward function provides a clearer and more effective learning signal, enabling the pursuers to more rapidly master cooperative capture strategies.

D. PEG with obstacles

TABLE II: Performance Comparison of Different Methods in 4v2 Scenarios with Obstacles

Method	**Nonlinear**		**Greedy Strategy**	
	Success Rate	Avg. Time(s)	Success Rate	Avg. Time(s)
Greedy Strategy	38%	35.56	39%	34.99
RAG+DeepSeekV3+Evolution	**100%**	**16.61**	**100%**	**19.10**
RAG+DeepSeekV3	39%	40.58	16%	45.36
DeepSeek V3+Evolution	86%	19.57	71%	27.13
AAPC+APF	100%	24.36	61%	33.80

To further challenge the proposed framework, we introduced static obstacles into the 4v2 PEG to simulate a more complex and realistic scenario. Fig. 8 illustrates the agents' trajectories in this constrained setting. The results demonstrate that our method has significant adaptability. The pursuers exhibit intelligent pathfinding, successfully navigating around the obstacles while maintaining effective coordination to corner the evaders. This behavior indicates that the evolved reward function implicitly learned to penalize collisions and incentivize efficient, obstacle-aware routes without requiring explicit programming for this specific constraint. This experiment underscores the robustness of our approach, highlighting its capability to generalize from open-space scenarios to complex environments and produce sophisticated, context-aware pursuit strategies.

These observations are substantiated by the quantitative data in Table II, where our method achieved a 100% success rate against both non-linear and greedy evader strategies, significantly outperforming baseline approaches. Notably, the Greedy Strategy's performance collapsed in this complex environment (below 40% success), while our framework also proved superior to the specialized AAPC+APF method [23],

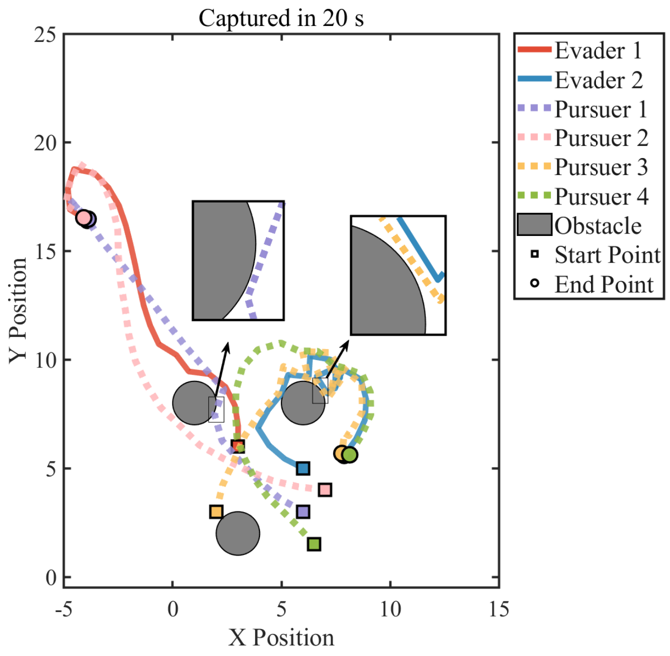


Fig. 8. Trajectory of pursuers and evaders in 4v2 PEG with obstacles.

highlighting its ability to generate more efficient and effective solutions.

Ablation studies further elucidate the synergistic contributions of our framework's core components. The model lacking the iterative evolution process (RAG+DeepSeekV3) struggled significantly, with success rates dropping to 39% and 16%, confirming that a high-quality initial reward function is insufficient without performance-based refinement. Conversely, removing the RAG module (DeepSeekV3+Evolution) also led to a marked performance decline, underscoring the importance of grounding the LLM in domain-specific knowledge to handle new environmental constraints effectively. In summary, this experiment validates that the integration of knowledge retrieval and iterative evolution enables our framework to autonomously generate robust reward functions that produce sophisticated, generalizable strategies for complex multi-agent tasks.

V. Conclusions

In this paper, we propose an automated framework for policy shaping to address the challenges of manual engineering in complex multi-agent tasks. By synergizing the RAG module with the LLM and integrating a data-driven evolutionary loop, our framework successfully transforms reward engineering from a labor-intensive, manual process into an autonomous system. Its effectiveness is validated through experiments, demonstrating superior performance and the emergence of cooperative strategies in complex PEG scenarios. Future work can focus on reducing the computational overhead of evolutionary cycles and further researching autonomous systems and their practical applications.

References

[1] S. Güler, M. Abdelkader, and J. S. Shamma, "Peer-to-peer relative localization of aerial robots with ultrawideband sensors," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 5, pp. 1981–1996, 2020.

[2] L. Xue, B. Ma, Y. Wu, J. Liu, C. Mu, and D. C. Wunsch, "Anti-jamming attack mixed strategy for formation tracking control via game-theoretical reinforcement learning," *IEEE Transactions on Intelligent Vehicles*, 2024.

[3] K. Manandhar, X. Cao, F. Hu, and Y. Liu, "Detection of faults and attacks including false data injection attack in smart grid using kalman filter," *IEEE transactions on control of network systems*, vol. 1, no. 4, pp. 370–379, 2014.

[4] V. G. Lopez, F. L. Lewis, Y. Wan, E. N. Sanchez, and L. Fan, "Solutions for multiagent pursuit-evasion games on communication graphs: Finite-time capture and asymptotic behaviors," *IEEE Transactions on Automatic Control*, vol. 65, no. 5, pp. 1911–1923, 2019.

[5] Z. Sun, H. Sun, P. Li, and J. Zou, "Cooperative strategy for pursuit-evasion problem with collision avoidance," *Ocean Engineering*, vol. 266, p. 112742, 2022.

[6] L. Xue, B. Ma, J. Liu, C. Mu, and D. C. Wunsch, "Extended kalman filter based resilient formation tracking control of multiple unmanned vehicles via game-theoretical reinforcement learning," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 3, pp. 2307–2318, 2023.

[7] R. Van Parys and G. Pipeleers, "Distributed mpc for multi-vehicle systems moving in formation," *Robotics and Autonomous Systems*, vol. 97, pp. 144–152, 2017.

[8] V. G. Lopez, F. L. Lewis, Y. Wan, E. N. Sanchez, and L. Fan, "Solutions for multiagent pursuit-evasion games on communication graphs: Finite-time capture and asymptotic behaviors," *IEEE Transactions on Automatic Control*, vol. 65, no. 5, pp. 1911–1923, 2019.

[9] Y. Xu, H. Yang, B. Jiang, and M. M. Polycarpou, "Multiplayer pursuit-evasion differential games with malicious pursuers," *IEEE Transactions on Automatic Control*, vol. 67, no. 9, pp. 4939–4946, 2022.

[10] L. Xue, J. Ye, Y. Wu, J. Liu, and D. Wunsch, "Prescribed-time Nash equilibrium seeking for pursuit-evasion game," *IEEE/CAA Journal of Automatica Sinica*, vol. 11, no. 6, pp. 1518–1520, 2024.

[11] L. C. Garaffa, M. Basso, A. A. Konzen, and E. P. de Freitas, "Reinforcement learning for mobile robotics exploration: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 8, pp. 3796–3810, 2021.

[12] Y. Wang, L. Dong, and C. Sun, "Cooperative control for multi-player pursuit-evasion games with reinforcement learning," *Neurocomputing*, vol. 412, pp. 101–114, 2020.

[13] C. H. Wu, D. A. Sofge, and D. M. Lofaro, "Crafting a robotic swarm pursuit–evasion capture strategy using deep reinforcement learning," *Artificial Life and Robotics*, vol. 27, no. 2, pp. 355–364, 2022.

[14] L. Xue, Q. Wang, Y. Wu, X. Yuan, and J. Liu, "Apollonius partitions based pursuit-evasion strategies via multi-agent reinforcement learning," *Neurocomputing*, vol. 630, p. 129643, 2025.

[15] M. Yuan, B. Li, X. Jin, and W. Zeng, "Automatic intrinsic reward shaping for exploration in deep reinforcement learning," in *International Conference on Machine Learning*, pp. 40531–40554, PMLR, 2023.

[16] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," *arXiv preprint arXiv:1710.11248*, 2017.

[17] E. Bıyık, D. P. Losey, M. Palan, N. C. Landolfi, G. Shevchuk, and D. Sadigh, "Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences," *The International Journal of Robotics Research*, vol. 41, no. 1, pp. 45–67, 2022.

[18] J. Song, Z. Zhou, J. Liu, C. Fang, Z. Shu, and L. Ma, "Self-refined large language model as automated reward function designer for deep reinforcement learning in robotics," *arXiv preprint arXiv:2309.06687*, 2023.

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

[20] H. Li, X. Yang, Z. Wang, X. Zhu, J. Zhou, Y. Qiao, X. Wang, H. Li, L. Lu, and J. 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*, pp. 16426–16435, 2024.

[21] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[22] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24824–24837, 2022.

[23] M. Chen, X. Zhang, G. Li, W. Lai, and C. Yang, "Multi-evader dynamic pursuit strategy based on improved aapc and auction algorithm amidst static and dynamic obstacles," *Expert Systems with Applications*, vol. 266, no. 000, 2025.