# LLM-TOC: LLM-Driven Theory-of-Mind Adversarial Curriculum for Multi-Agent Generalization

Chenxu Wang , Jiang Yuan , Tianqi Yu , Xinyue Jiang , Liuyu Xiang , Junge Zhang , Zhaofeng He *

Posted Date: 14 February 2026

doi: 10.20944/preprints202602.1147.v1

Keywords: multi-agent reinforcement learning; large language models; adversarial curriculum learning; zero-shot generalization; gradient saliency

 doi:10.20944/preprints202602.1147.v1



*Article*

# LLM-TOC: LLM-Driven Theory-of-Mind Adversarial Curriculum for Multi-Agent Generalization

**Chenxu Wang [1], Jiang Yuan [2], Tianqi Yu [1], Xinyue Jiang [3], Liuyu Xiang [1], Junge Zhang [4] and Zhaofeng He [1,*]**

[1] School of Artificial Intelligence, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2] Beijing Institute of Astronautical Systems Engineering, Beijing, China
[3] School of Mathematics and Physics, Beijing University of Posts and Telecommunications, Beijing 100876, China
[4] Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
* Correspondence: zhaofenghe@bupt.edu.cn

**Abstract**

Zero-shot generalization to out-of-distribution (OOD) teammates and opponents in open-ended multi-agent systems (MAS) remains a fundamental challenge for general-purpose AI. Existing multi-agent reinforcement learning (MARL) paradigms, such as self-play and population-based training, often collapse to a limited subset of Nash equilibria, leaving agents brittle when faced with semantically diverse, unseen behaviors. Recent approaches that invoke large language models (LLMs) at run time can improve adaptability but introduce substantial latency and can become less reliable as task horizons grow; in contrast, LLM-assisted reward-shaping methods remain constrained by the inefficiency of the inner reinforcement-learning loop. To address these limitations, we propose LLM-TOC (LLM-Driven Theory-of-Mind Adversarial Curriculum), which casts generalization as a bi-level Stackelberg game: in the inner loop, a MARL agent (the follower) minimizes regret against a fixed population, while in the outer loop an LLM serves as a semantic oracle that generates executable adversarial or cooperative strategies in a Turing-complete code space to maximize the agent's regret. To cope with the absence of gradients in discrete code generation, we introduce Gradient Saliency Feedback, which transforms pixel-level value fluctuations into semantically meaningful causal cues to steer the LLM toward targeted strategy synthesis. We further provide PAC-Bayes guarantees showing that LLM-TOC converges at rate $O(1/\sqrt{K})$ and yields a tighter generalization error bound than parameter-space exploration. Experiments on the Melting Pot benchmark demonstrate that LLM-TOC consistently improves zero-shot performance over self-play baselines (IPPO, MAPPO) and the LLM-inference method Hypothetical Minds, while reducing training cost by more than 60%.

**Keywords:** multi-agent reinforcement learning; large language models; adversarial curriculum learning; zero-shot generalization; gradient saliency

## 1. Introduction

Developing autonomous agents capable of seamless interaction with diverse counterparts, including humans and other artificial agents, in open-ended environments remains a fundamental challenge in the pursuit of Artificial General Intelligence (AGI) [1,2]. Although Multi-Agent Reinforcement Learning (MARL) has achieved superhuman performance in zero-sum games such as StarCraft II [3] and Dota 2 [4], these successes primarily rely on the Self-Play (SP) paradigm within closed systems. In such settings, the focal agent co-evolves with a fixed set of partners, typically converging to specific Nash equilibrium subsets [5]. Consequently, when deployed in mixed-motive, open-ended environments such as Melting Pot [6,7], these specialized agents frequently fail to generalize to Out-of-Distribution (OOD) teammates or opponents exhibiting behaviors distinct from the training population, as illustrated in Figure 1 [8]. This phenomenon, termed the Zero-Shot Coordination (ZSC) gap [9], underscores the critical need for training methodologies that foster robust and adaptable social intelligence.

 doi:10.20944/preprints202602.1147.v1



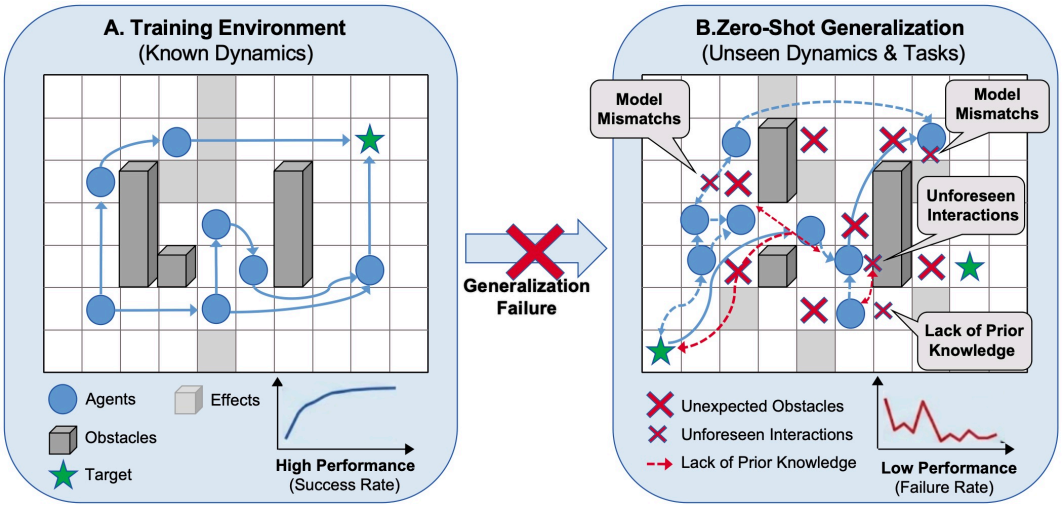


**Figure 1. Illustration of generalization failure in multi-agent systems.** (A) In the training environment, agents overfit to known dynamics and fixed scenarios, achieving high performance. (B) During zero-shot generalization, these specialized agents encounter unseen dynamics, model mismatches, and unforeseen interactions, leading to a significant performance drop and generalization failure.

To mitigate this limitation, recent research has focused on enhancing the diversity of training populations. Approaches such as Fictitious Co-Play (FCP) [10] and Population-Based Training (PBT) [11] introduce diversity by retaining ensembles of past checkpoints or constructing heterogeneous policy pools. However, these methods typically operate within the low-level parameter space of neural networks. Consequently, the resulting diversity is often superficial, manifesting primarily as variations in randomness or execution speed rather than distinct high-level semantic strategies, such as deception, altruism, or retaliation [12]. Therefore, when confronted with unseen counterparts exhibiting complex and semantically meaningful behaviors, agents trained via these methods may still struggle to maximize returns.

The emergence of Large Language Models (LLMs) offers a promising avenue for injecting semantic diversity into MARL. LLMs encapsulate vast amounts of human social prior knowledge and possess robust reasoning capabilities [13,14]. Frontier studies, such as ProAgent [15] and Hypothetical Minds [16], integrate LLMs directly into the agent's reasoning loop to facilitate real-time Theory of Mind (ToM) inference. Although these LLM-as-agent approaches have demonstrated efficacy, they incur substantial inference latency and high computational costs, rendering them unsuitable for real-time interaction in high-frequency environments. Conversely, an alternative paradigm, exemplified by SEMDIV [17], utilizes LLMs during the training phase to generate diverse reward functions, subsequently training partner strategies via Reinforcement Learning (RL). While this methodology circumvents real-time inference latency, it introduces a significant bottleneck: training a new strategy from scratch for each generated reward function. This RL inner loop process is not only computationally prohibitive but also suffers from poor training stability [18].

This paper presents LLM-Driven Theory-of-Mind Adversarial Curriculum (LLM-TOC), a novel framework that fundamentally redefines the role of LLMs within MARL training. Diverging from conventional approaches that utilize LLMs for online inference or indirect reward shaping, this framework positions the LLM as an offline semantic oracle capable of directly generating executable Python code for rule-based policies. We formalize the generalization problem as a bi-level Stackelberg game: in the outer loop, the LLM functions as a teacher, generating an adversarial population of code-based policies specifically designed to exploit the current weaknesses of the student agent; in the inner loop, the student agent, implemented as a lightweight neural network, efficiently optimizes its best-response strategy via MARL.

A core challenge inherent to this framework is the modality gap, where LLMs excel at processing textual information but struggle to interpret the numerical feedback typical of RL environments,

 doi:10.20944/preprints202602.1147.v1



including pixel-based observations and scalar rewards. To bridge this gap and establish a closed training loop, we introduce a Gradient Saliency Feedback mechanism. By computing the gradients of the student agent's value function with respect to input observations, this mechanism extracts pixel-level attention maps to precisely identify the visual determinants of agent failure, such as overlooking an opponent positioned behind a wall. These attention maps are subsequently translated into natural language descriptions, providing the LLM with explicit causal grounding to generate highly targeted adversarial strategies. Our contributions are summarized as follows:

1. We propose the LLM-TOC framework, which leverages the code generation capabilities of LLMs to construct an infinitely scalable and semantically diverse adversarial curriculum. This approach effectively circumvents the computational inefficiency associated with the inner loop training of traditional RL methods.
2. We introduce a gradient-based saliency mechanism designed to translate numerical feedback regarding behavioral failures into semantic prompts. This mechanism enables the LLM to perform semantic gradient ascent within the abstract policy space, thereby optimizing strategies based on explicit causal information.
3. Extensive empirical evaluations on the Melting Pot benchmark demonstrate that LLM-TOC achieves superior zero-shot generalization performance against unseen emergent strategies compared to state-of-the-art baselines, including SP and ToM methods. Furthermore, our framework reduces computational training time by more than 60%.

## 2. Related Work

### *2.1. Zero-Shot Coordination and Generalization in MARL*

ZSC aims to develop agents capable of effective coordination with unseen partners without requiring prior data or fine-tuning [9,19]. Classic baselines in MARL, including Independent Proximal Policy Optimization (IPPO) [20], Multi-Agent Proximal Policy Optimization (MAPPO) [21], and Value Decomposition Networks (QMIX) [22], typically rely on the SP paradigm to optimize joint returns within closed domains such as StarCraft II [3]. Although these methods demonstrate efficacy in fixed settings, they frequently converge to specific, arbitrary coordination conventions, resulting in significant behavioral fragility when paired with out-of-distribution partners [8]. Early improvements such as Other-Play (OP) [23] mitigate reliance on arbitrary coordination signals by enforcing symmetry invariance, yet they struggle to scale effectively in complex, asymmetric environments.

To enhance agent robustness against diverse behaviors, population-based training methods have emerged as a dominant research paradigm. FCP [10], which trains agents against a population of checkpoints saved during SP, serves as a strong baseline for evaluating behavioral robustness; however, the diversity it achieves is strictly constrained by the training trajectory of a single optimization algorithm. Recent studies attempt to explicitly maximize training population diversity through targeted designs: Trajectory Diversity (TrajeDi) [24] and Latent Space Optimization (LIPO) [25] utilize Jensen-Shannon divergence or latent variables to induce differentiated behavioral patterns, while EvoAgent [26] employs evolutionary algorithms to automatically generate diverse agent strategies. Despite these advancements, most existing approaches operate within low-level parameter or trajectory spaces, yielding diversity that manifests primarily as stochastic variations rather than the high-level semantic strategies—such as deception or sacrifice—essential for human-level coordination [27,28]. In contrast to these methods, the proposed LLM-TOC framework leverages the code generation capabilities of LLMs to explore a Turing-complete semantic space, thereby uncovering complex interaction strategies inaccessible to traditional parameter-space noise injection techniques.

### *2.2. Large Language Models for Autonomous Agents*

Driven by the reasoning and planning capabilities of LLMs, their integration with autonomous agents has witnessed explosive growth [1,2]. Existing research primarily falls into two paradigms: the LLM-as-Agent paradigm and the LLM-Assisted Training paradigm.

 doi:10.20944/preprints202602.1147.v1



The LLM-as-Agent paradigm integrates LLMs directly into the agent's decision-making loop. Studies such as Voyager [29] and Ghost in the Minecraft (GITM) [30] demonstrate that LLMs can facilitate continuous skill acquisition in open-ended environments through code generation. In multi-agent scenarios, MetaGPT [31] and AgentVerse [32] explore role-based collaboration models, applying them to the domain of software development. Most closely related to this work is Hypothetical Minds [16], which utilizes LLMs to infer the behavioral intentions of other agents and plan response strategies in real-time based on ToM. Although Hypothetical Minds has been validated on the Melting Pot benchmark [6] by employing hand-crafted parsers to convert visual observations into textual information, it suffers from high inference latency and substantial token costs. Furthermore, without such manual engineering, it remains difficult to deploy in high-frequency visual control tasks.

Conversely, the LLM-Assisted Training paradigm leverages LLMs to support the training of lightweight RL policies. Approaches such as Eureka [33] and Text2Reward [18] employ LLMs to synthesize dense reward functions, thereby guiding the training of RL agents. Although SEMDIV [17] attempts to generate diverse partner agents through reward shaping, it relies on a computationally expensive RL inner loop, necessitating the training of a new policy from scratch for each generated reward function. Distinct from these methods, the proposed LLM-TOC framework innovates by positioning the LLM as an offline semantic oracle that directly generates executable policy code. This approach combines the semantic generalization capabilities of Hypothetical Minds with the execution efficiency of lightweight MARL agents, such as MAPPO. Consequently, it circumvents the latency associated with real-time inference while ensuring strong generalization capabilities against semantically diverse opponents [13,34].

### *2.3. Automated Curriculum Learning and Environment Design*

Research on Automatic Curriculum Learning (ACL) and Unsupervised Environment Design (UED) aims to generate a sequence of training tasks that maximizes agent learning efficiency [35, 36]. The PAIRED algorithm [37] employs an adversary to generate training environments that are challenging for the student agent yet solvable by the adversary itself. Building upon this, ACCEL [38] enhances the process by utilizing Quality-Diversity (QD) algorithms, such as MAP-Elites [39], to evolve more robust training environments. Similarly, Prioritized Level Replay (PLR) [40] focuses on replaying levels with high regret values to improve agent behavioral robustness.

In the context of social generalization, the concept of an environment encompasses not only physical scenarios but also the interacting counterparts. Recent studies in Open-Ended Learning (OEL) [41,42] emphasize that the training process requires a continuous stream of novel challenges to sustain learning. However, the vast majority of existing ACL methods generate only environmental features, such as grid-world layouts or physical parameters, rather than interactive policies that reflect distinct behavioral patterns. The proposed method incorporates a Gradient Saliency Feedback mechanism [43,44] to deeply align the curriculum generation process with the specific failure modes of the student agent. This design enables the LLM to perform a semantic gradient ascent, generating highly targeted adversarial interaction scenarios—such as sneak attacks from behind—that efficiently expose and rectify the student agent's performance deficits. Compared to unguided evolutionary search methods [45–47], this approach significantly improves the training convergence efficiency of the agent.

## 3. Problem Formulation

### *3.1. Partially Observable Markov Games*

We formalize open-ended multi-agent interaction as a Partially Observable Markov Game (POMG) [48,49], defined by the tuple $\mathcal{G} = \langle \mathcal{S}, N, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, \mathcal{O}, \gamma \rangle$. Here, $\mathcal{S}$ denotes the global state space, and $N = \{1, \ldots, n\}$ represents the set of agents. We partition $N$ into the student agent ($E$), which acts as the focal agent, and the other agents ($-E$), comprising potential teammates or opponents. The joint action space is defined as $\mathcal{A} = \mathcal{A}_E \times \mathcal{A}_{-E}$, while the state transition dynamics are governed

 doi:10.20944/preprints202602.1147.v1



by $\mathcal{P}(s'|s, a_E, a_{-E})$. The term $\mathcal{R}_E(s, a_E, a_{-E})$ specifies the reward function for the student agent. Additionally, $\Omega_i$ denotes the set of partial observations for agent $i$, generated by the observation function $\mathcal{O}_i(s)$, and $\gamma \in [0, 1)$ represents the discount factor.

The student agent operates according to a policy $\pi_E(a_E|o_E; \theta)$, parameterized by $\theta$, such as the weights of a neural network. Conversely, the other agents follow a joint policy $\pi_{-E}(a_{-E}|o_{-E})$, which may be driven by diverse or unknown logic. The expected return for the student agent against a specific counterpart $\pi_{-E}$ is defined as:

$$J(\pi_E, \pi_{-E}) = \mathbb{E}_{\tau \sim (\pi_E, \pi_{-E})} \left[ \sum_{t=0}^{\infty} \gamma^t \mathcal{R}_E(s_t, a_{E,t}, a_{-E,t}) \right]. \tag{1}$$

### 3.2. Zero-Shot Generalization as Minimax Regret

A central challenge within open-ended environments such as Melting Pot [6] is zero-shot generalization. The student agent $\pi_E$ is required to demonstrate robust performance when paired with test-time counterparts $\pi_{-E}^{test}$ sampled from an unknown distribution $\mathcal{D}_{test}$, without the opportunity for fine-tuning. Conventional MARL approaches typically assume that $\pi_{-E}$ originates from a restricted set of behaviors, such as checkpoints derived from self-play. However, in realistic scenarios, $\pi_{-E}$ resides within a vast semantic strategy space, denoted as $\Pi_{sem}$. This space encompasses all behaviorally distinct policies expressible through logical rules, code, or natural language, including strategies such as tit-for-tat, deceptive cooperation, or aggressive blocking. Notably, this space is significantly larger and exhibits a discrete structure distinct from the continuous parameter space of neural networks.

To achieve robust generalization, we formulate the training objective as the minimization of maximum regret over the entire semantic strategy space $\Pi_{sem}$. We first define the regret of a student policy $\pi_E$ against a specific counterpart $\pi_{-E}$ as the difference between the maximal achievable return against $\pi_{-E}$ and the actual return:

$$\text{Regret}(\pi_E, \pi_{-E}) = \underbrace{\max_{\pi'} J(\pi', \pi_{-E})}_{V^*(\pi_{-E})} - J(\pi_E, \pi_{-E}), \tag{2}$$

where $V^*(\pi_{-E})$ represents the oracle performance achievable assuming perfect prior knowledge of $\pi_{-E}$. Our objective is to identify an optimal student policy $\pi_E^*$ that minimizes the worst-case regret across all possible semantic strategies:

$$\pi_E^* = \arg\min_{\pi_E} \left( \max_{\pi_{-E} \in \Pi_{sem}} \text{Regret}(\pi_E, \pi_{-E}) \right). \tag{3}$$

### 3.3. The Challenge of Semantic Coverage

Directly solving the optimization problem presented above is intractable due to two primary factors: **Infinite Space:** The semantic space $\Pi_{sem}$ is effectively infinite and non-differentiable, rendering traversal via standard gradient-based optimization or simple population sampling, such as FCP, infeasible. **Modality Gap:** Conventional RL methods operate within the parameter space $\Theta$. Exploration of $\Theta$ through Gaussian noise—a technique commonly employed in PBT—rarely yields high-level semantic strategies, such as complex deception. Consequently, such methods fail to adequately cover the support of $\Pi_{sem}$.

Therefore, an efficient mechanism is required to identify the worst-case $\pi_{-E}$ within $\Pi_{sem}$ to serve as a curriculum. This necessity motivates our proposed LLM-TOC framework, which leverages LLMs as a semantic engine to generate $\pi_{-E}$ directly within the code space, thereby approximating the $\max_{\pi_{-E}}$ operator in a computationally feasible manner. In Appendix A, we list all the symbols used in this paper and provide their definitions.

 doi:10.20944/preprints202602.1147.v1



## 4. Methodology

Building upon the established problem formulation, we propose LLM-TOC, a framework designed to approximate the intractable minimax regret objective through a bi-level iterative process. This methodology comprises two coupled loops, as illustrated in Figure 2: an outer loop wherein a Large Language Model functions as a semantic oracle to generate adversarial policies $\pi_{-E}$ directly within the code space, and an inner loop in which the student agent $\pi_E$ performs robust optimization against the generated population. To facilitate this interaction, we introduce a Gradient Saliency Feedback mechanism that effectively bridges the modality gap between numerical reinforcement learning signals and semantic reasoning.

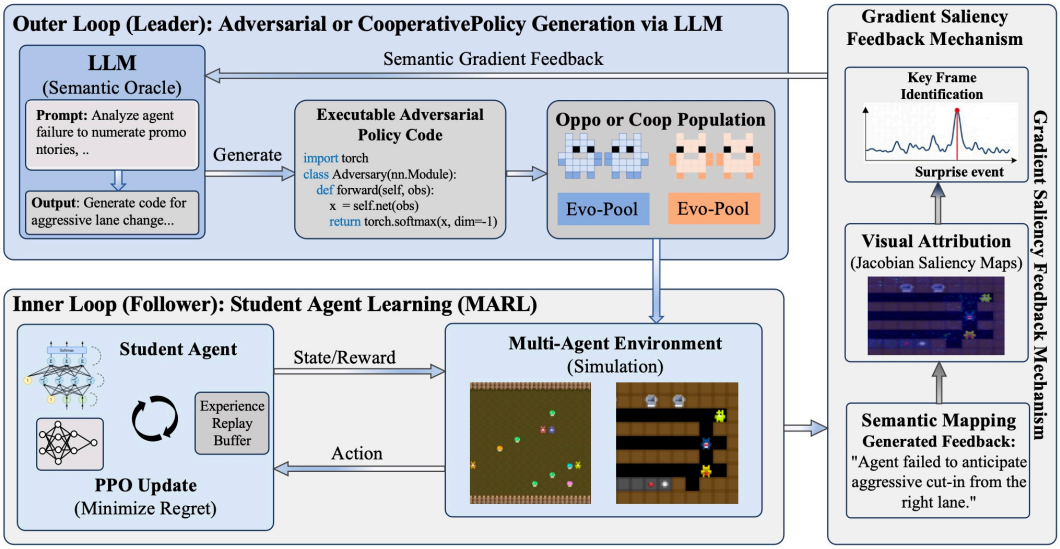


**Figure 2. Schematic overview of the LLM-TOC framework.** The architecture operates as a bi-level iterative process comprising three key components: (1) The Outer Loop (Leader), where the LLM functions as a semantic oracle to generate executable policy code for the adversarial or cooperative population (Evo-Pool); (2) The Inner Loop (Follower), where the student agent optimizes its policy via MARL, specifically using PPO updates to minimize regret; and (3) The Gradient Saliency Feedback Mechanism, which identifies key failure frames, computes visual attributions via Jacobian saliency maps, and translates these numerical signals into semantic textual feedback to guide the LLM's subsequent generation.

### *4.1. The Bi-Level Optimization Framework*

We formally structure the training process as a Stackelberg game, modeled as a bi-level optimization problem. In this hierarchy, the leader initiates the interaction by selecting a distribution of opponent strategies designed to exploit the weaknesses of the follower. In response, the follower optimizes its policy against this distribution. Let $\pi_E \in \Pi_{net}$ denote the student policy parameterized by $\theta$, and $\mathcal{P} \subset \Pi_{sem}$ represent the population of strategies for the other agents. The framework is defined by two nested optimization objectives:

**The Follower Objective (Inner Loop):** Given a fixed population $\mathcal{P}_k$ provided by the leader at iteration $k$, the student seeks to maximize its expected return. This corresponds to the standard MARL objective:

$$\pi_E^*(\mathcal{P}_k) = \arg\max_{\pi_E \in \Pi_{net}} \mathcal{J}_{student}(\pi_E, \mathcal{P}_k) = \arg\max_{\pi_E} \mathbb{E}_{\pi_{-E} \sim U(\mathcal{P}_k)} \left[ \mathbb{E}_{\tau \sim (\pi_E, \pi_{-E})} \left[ \sum_{t=0}^{T} \gamma^t r_t \right] \right]. \quad (4)$$

This step aims to identify the best response strategy $\pi_E^*$ that exhibits robustness to the current set of counterparts.

**The Leader Objective (Outer Loop):** The leader aims to expand the population $\mathcal{P}_k$ by introducing a new strategy $\pi_{-E}^{(k+1)}$ that maximizes the regret of the student. As calculating exact regret requires an optimal oracle $V^*$, we approximate this objective by maximizing the exploitability of the

 doi:10.20944/preprints202602.1147.v1



student—specifically, by identifying a strategy that minimizes the performance of the current student policy $\pi_E^*(\mathcal{P}_k)$:

$$\pi_{-E}^{(k+1)} = \arg \min_{\pi_{-E} \in \Pi_{sem}} \mathbb{E}_{\tau \sim (\pi_E^*, \pi_{-E})} \left[ \sum_{t=0}^{T} \gamma^t r_t \right]. \tag{5}$$

However, as the semantic space $\Pi_{sem}$ is discrete and non-differentiable, optimization of the leader's objective via gradient descent is infeasible. Consequently, we employ the LLM as a semantic oracle $\Phi$ to approximate this process:

$$\pi_{-E}^{(k+1)} \approx \Phi(\pi_E^*(\mathcal{P}_k) \mid \text{Feedback}). \tag{6}$$

This bi-level architecture ensures that the student is continuously challenged by worst-case scenarios, thereby driving the expansion of its robust hull within the strategy space.

### *4.2. Gradient Saliency Feedback Mechanism*

To effectively guide the semantic oracle $\Phi$ within the outer loop, bridging the modality gap between numerical reinforcement learning (RL) losses and the semantic reasoning capabilities of LLMs is essential. To this end, we introduce a Gradient Saliency Feedback mechanism designed to perform causal attribution.

#### 4.2.1. Identifying Critical Moments via Value Surprise

We first identify specific instances of policy failure by monitoring the value function $V(s;\phi)$ of the student agent $\pi_E$ during evaluation episodes. We define the value surprise, denoted as $\delta_t$, as the absolute Temporal Difference (TD) error, which quantifies the discrepancy between the expected return of the agent and the actual outcome:

$$\delta_t = |r_t + \gamma V(s_{t+1};\phi) - V(s_t;\phi)|. \tag{7}$$

A high magnitude of $\delta_t$ signifies a critical frame $t^*$, indicating a pivotal event such as an unexpected penalty, an adverse interaction, or a missed reward. We extract a set of critical frames $\mathcal{T}_{crit} = \{t \mid \delta_t > \epsilon_{thresh}\}$ for subsequent detailed analysis.

#### 4.2.2. Visual Attribution via Jacobian Saliency

For each critical frame $t^* \in \mathcal{T}_{crit}$, we elucidate the cause of the value deviation by computing the sensitivity of the value function with respect to the input observation $X_{t^*} \in \mathbb{R}^{C \times H \times W}$. This is achieved by calculating the Jacobian matrix of the value function relative to the input pixels.

Specifically, let $V(s_{t^*})$ represent the scalar value output. We compute the gradient map $\mathcal{G}_{t^*} \in \mathbb{R}^{C \times H \times W}$ via backpropagation:

$$\mathcal{G}_{t^*} = \nabla_{X_{t^*}} V(s_{t^*};\phi) = \frac{\partial V(s_{t^*})}{\partial X_{t^*}}. \tag{8}$$

To obtain a single 2D saliency map $M_{t^*} \in \mathbb{R}^{H \times W}$, we aggregate the gradients across the channel dimension—comprising RGB or feature channels—by computing the maximum absolute value or the $L_2$ norm:

$$M_{t^*}^{(h,w)} = \max_c \left| \mathcal{G}_{t^*}^{(c,h,w)} \right|. \tag{9}$$

This map $M_{t^*}$ highlights specific regions in the visual field where perturbations would induce the most significant shifts in the value estimation of the agent. Mathematically, this approximates the first-order Taylor expansion of the value function, identifying the features to which the agent attends—or fails to attend—during the critical event.

 doi:10.20944/preprints202602.1147.v1



4.2.3. Semantic Mapping

We map the pixel-level saliency $M$ to semantic concepts using ground-truth object masks provided by the environment engine. Let $\mathbb{I}_{obj}$ denote the binary mask for a specific object class $obj$, such as counterparts $\pi_{-E}$ or resources. The attention score for each object is calculated as follows:

$$\text{Score}_{obj} = \frac{\sum_{i,j} M_{i,j} \cdot \mathbb{I}_{obj,i,j}}{\sum_{i,j} M_{i,j} + \epsilon}. \tag{10}$$

By comparing these scores, we generate semantic descriptions. For instance, if $\text{Score}_{-E} \gg \text{Score}_{goal}$ during a significant value drop, the system generates a description indicating that the focal agent focused heavily on the opponent while neglecting the goal, thereby leading to failure.

### *4.3. Policy Generation and Curriculum Evolution*

Central to our curriculum evolution is the transformation of semantic feedback into executable policy code. We formalize this process as a conditional generation problem within the semantic space.

Let $\mathcal{D}_{sem}^{(k)}$ denote the semantic diagnosis derived from the gradient saliency analysis at iteration $k$. We construct a structured prompt $\mathbf{P}_k$ that encapsulates the designated role, API constraints, and the diagnosis:

$$\mathbf{P}_k = \text{Concat}(\mathcal{I}_{role}, \mathcal{I}_{API}, \mathcal{D}_{sem}^{(k)}), \tag{11}$$

where $\mathcal{I}_{role}$ defines the adversarial objective and $\mathcal{I}_{API}$ specifies the programming interface.

The LLM functions as a generator, sampling new policy code $\pi_{code}^{(k+1)}$ from its learned distribution:

$$\pi_{code}^{(k+1)} \sim P_{LLM}(\cdot \mid \mathbf{P}_k). \tag{12}$$

This generation process is conceptualized as a semantic gradient ascent. The diagnosis $\mathcal{D}_{sem}^{(k)}$ serves as a gradient direction within the semantic manifold, pointing toward the region of the strategy space where the student agent $\pi_E$ exhibits maximum vulnerability. For example, if the diagnosis indicates a susceptibility to rear attacks, the LLM generates logic specifically targeting the blind spot of the focal agent.

Finally, the curriculum evolves through the aggregation of these generated policies. The population $\mathcal{P}$ is updated cumulatively:

$$\mathcal{P}_{k+1} = \mathcal{P}_k \cup \text{Compile}(\pi_{code}^{(k+1)}). \tag{13}$$

This mechanism prevents catastrophic forgetting in the student agent $\pi_E$. To minimize regret in the subsequent inner loop, the agent is compelled to simultaneously enhance its robustness against all previously generated strategies in addition to the newly introduced adversarial policy.

### *4.4. Algorithm Summary*

The LLM-TOC algorithm, summarized in Algorithm 1, structures the learning process as a systematic, iterative cycle designed to progressively enhance the robustness of the student agent. The process commences with initialization, wherein the student agent $\pi_E$ is instantiated with random weights, and the initial population $\mathcal{P}_0$ is seeded with basic heuristic-based policies to provide foundational training signals.

 doi:10.20944/preprints202602.1147.v1



**Algorithm 1** LLM-TOC: LLM-Driven Theory-of-Mind Adversarial Curriculum

**Input**: Max outer iterations $K$, Inner steps $T_{in}$, Threshold $\epsilon_{thresh}$, Learning rate $\alpha$;
**Input**: Initialize student policy $\pi_E$ with parameters $\theta$;
**Input**: Initialize opponent population $\mathcal{P}_0 \leftarrow \{\pi_{heuristic}\}$;

**for** $k = 1, \ldots, K$ **do** ▷ **Outer Loop: Curriculum Evolution**
  $\mathcal{P}_{curr} \leftarrow \mathcal{P}_{k-1}$;
  *// Phase 1: Inner Loop (Student Optimization)*
  **repeat**
    Sample opponent batch $\pi_{-E} \sim U(\mathcal{P}_{curr})$;
    Collect trajectories $\tau = \{(s_t, a_t, r_t, s_{t+1})\}$ using $\pi_E$ and $\pi_{-E}$;
    Compute advantages $A_t$ using GAE;
    Update $\theta \leftarrow \theta + \alpha \nabla_\theta \mathbb{E}_\tau[\min(\rho_t A_t, \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)A_t)]$; ▷ PPO Update
  **until** Performance converges
  *// Phase 2: Evaluation & Diagnosis (Gradient Saliency)*
  Identify worst-case opponent: $\pi^*_{-E} \leftarrow \arg\min_{\pi \in \mathcal{P}_{curr}} \mathbb{E}_\tau[\sum \gamma^t r_t]$;
  Collect evaluation rollout $\mathcal{D}_{eval}$ against $\pi^*_{-E}$;
  Initialize diagnosis set $\mathcal{D}^{(k)}_{sem} \leftarrow \emptyset$;
  **for** each step $t$ in $\mathcal{D}_{eval}$ **do**
    Calculate Value Surprise: $\delta_t \leftarrow |r_t + \gamma V(s_{t+1}) - V(s_t)|$;
    **if** $\delta_t > \epsilon_{thresh}$ **then** ▷ Identify Critical Moment
      Compute Jacobian: $\mathcal{G}_t \leftarrow \nabla_{X_t} V(s_t)$;
      Aggregate Saliency: $M_t^{(h,w)} \leftarrow \max_c |\mathcal{G}_t^{(c,h,w)}|$;
      Compute Object Scores: $\text{Score}_{obj} \leftarrow \frac{\sum M \cdot \mathbb{I}_{obj}}{\sum M + \epsilon}$;
      Generate description $S_t$ based on $\text{Score}_{obj}$, Focus on $\pi^*_{-E}$ > Goal;
      $\mathcal{D}^{(k)}_{sem} \leftarrow \mathcal{D}^{(k)}_{sem} \cup \{S_t\}$;
    **end if**
  **end for**
  *// Phase 3: Semantic Generation (Leader Optimization)*
  Construct Prompt $\mathbf{P}_k \leftarrow \text{Concat}(\mathcal{I}_{role}, \mathcal{I}_{API}, \mathcal{D}^{(k)}_{sem})$;
  Generate Adversarial Policy: $\pi^{(k)}_{code} \sim P_{LLM}(\cdot \mid \mathbf{P}_k)$;
  Validate and Compile $\pi^{(k)}_{code}$;
  Update Population: $\mathcal{P}_k \leftarrow \mathcal{P}_{curr} \cup \{\pi^{(k)}_{code}\}$;
**end for**
**Output**: Robust Student Policy $\pi_E$

Subsequently, the algorithm enters the main outer loop, which governs curriculum evolution. Within each epoch, the system first executes the inner loop, where $\pi_E$ undergoes MARL—specifically utilizing MAPPO—against the current population $\mathcal{P}_k$. This phase persists until the performance of the student converges, ensuring mastery of the existing curriculum.

Upon convergence of the inner loop, the system proceeds to the evaluation and diagnosis phase. The student agent is evaluated against the population to identify the worst-case counterpart that induces the highest regret. Trajectories corresponding to these failure cases are recorded. The Gradient Saliency Mechanism then analyzes these trajectories, computing value surprise to identify critical moments and Jacobian saliency to attribute failure to specific visual objects. This numerical data is synthesized into a natural language description characterizing the specific vulnerabilities of the student.

Guided by this diagnosis, the semantic oracle (LLM) executes the generation phase. It synthesizes a new Python policy class specifically designed to exploit the identified weakness, such as an agent that ambushes from behind if $\pi_E$ neglects its rear. This new policy is verified for executability and subsequently added to the population, effectively expanding the curriculum ($\mathcal{P}_{k+1} \leftarrow \mathcal{P}_k \cup \{\pi_{new}\}$). This cycle repeats for a predefined number of iterations or until the LLM can no longer generate strategies that significantly reduce the return of the student, at which point the policy $\pi_E$ is deemed

 doi:10.20944/preprints202602.1147.v1



robust and returned as the final output. In Appendix C, we provide a detailed description of the prompt templates used by the LLM.

### 4.5. Theoretical Analysis

We establish the convergence properties and generalization guarantees of LLM-TOC, utilizing the PAC-Bayes framework to substantiate the claims of superior sample efficiency.

**Convergence Rate.** Modeling the bi-level optimization as a Stackelberg game, we define the regret of the student $\pi_E$ against the best-response adversary as $\mathcal{R}(\pi_E)$. Assuming the LLM functions as an $\epsilon$-approximate semantic oracle that consistently identifies descent directions in the strategy manifold, the average regret $\bar{\mathcal{R}}_K$ over $K$ outer-loop iterations is bounded by:

$$\bar{\mathcal{R}}_K \leq \frac{C}{\sqrt{K}} + \epsilon_{approx}, \tag{14}$$

where $C$ is a constant related to the strategy space diameter and $\epsilon_{approx}$ represents the oracle's approximation error. This indicates that LLM-TOC converges to a robust equilibrium at a rate of $O(1/\sqrt{K})$.

**Generalization Bounds.** We compare the generalization error $\mathcal{L}(\mathcal{Q})$ of our semantic exploration against parameter-space exploration using the PAC-Bayes bound. Let $\mathcal{Q}$ be the posterior distribution of strategies and $\mathcal{P}$ the prior. With probability at least $1-\delta$:

$$\mathcal{L}(\mathcal{Q}) \leq \hat{\mathcal{L}}(\mathcal{Q}) + \sqrt{\frac{D_{KL}(\mathcal{Q}||\mathcal{P}) + \ln(1/\delta)}{2m}}. \tag{15}$$

In traditional MARL, the prior $\mathcal{P}_{param}$ is isotropic Gaussian over weights, yielding a large Kullback-Leibler divergence $D_{KL}$ when fitting complex behaviors. In LLM-TOC, the prior $\mathcal{P}_{sem}$ is induced by the LLM's pre-trained knowledge of code and logic, which aligns closely with valid strategy distributions. Consequently, $D_{KL}(\mathcal{Q}_{sem}||\mathcal{P}_{sem}) \ll D_{KL}(\mathcal{Q}_{param}||\mathcal{P}_{param})$, resulting in a strictly tighter generalization bound and enhanced zero-shot robustness. In Appendix D, we present a detailed theoretical analysis of our algorithm.

## 5. Experiments

### 5.1. Experimental Setup

#### 5.1.1. Evaluation Benchmark: Melting Pot

To rigorously evaluate zero-shot generalization in open-ended multi-agent systems, we utilize **Melting Pot 2.0** [6], a benchmark suite specifically designed to test social intelligence against unseen other agents. We select four diverse substrates that cover distinct social dilemmas:

1. `collaborative_cooking_asymmetric`: A coordination task requiring role specialization and synchronized action sequences to complete recipes.
2. `prisoners_dilemma_in_the_matrix_repeated`: A classic social dilemma testing the agent's ability to maintain cooperation against defection risks over repeated interactions.
3. `running_with_scissors_in_the_matrix_arena`: A spatially complex, cyclical resource competition game (Rock-Paper-Scissors dynamics) where agents must identify and counter opponent strategies.
4. `running_with_scissors_in_the_matrix_repeated`: A repeated version of the arena task, emphasizing long-term memory and reciprocity.

We adopt a strictly **Zero-Shot Generalization (ZSG)** evaluation protocol: agents are trained on the base substrates and evaluated on held-out test scenarios. These test scenarios introduce focal-population mismatches or specific opponent behaviors not encountered during training, serving as a robust testbed for OOD generalization. In Appendix B, we provide a detailed description of the training and evaluation environments.

 doi:10.20944/preprints202602.1147.v1



#### 5.1.2. Baselines

We benchmark LLM-TOC against three categories of methods to validate its effectiveness:

**Standard MARL (IPPO & MAPPO)** [20,21]: Independent PPO and Multi-Agent PPO trained via self-play. These represent standard reinforcement learning baselines that typically overfit to the training population and struggle with OOD partners.

**Hypothetical Minds** [16]: A state-of-the-art method that utilizes a frozen Large Language Model for run-time decision-making based on visual descriptions. This baseline represents the capability of direct LLM inference in social scenarios.

**Oracle PPO (Skyline)**: A PPO agent trained directly on the test scenarios. This serves as the theoretical performance upper bound (Skyline), indicating the maximum achievable return if the test distribution were known in advance.

#### 5.1.3. Implementation Details

**Observation Processing.** Instead of feeding raw RGB pixels directly to the policy network, we adopt a semantically structured observation space inspired by *Hypothetical Minds* [16] to enhance sample efficiency and align with the LLM's reasoning granularity. Specifically, the raw $88 \times 88$ pixel input is discretized into an $11 \times 11$ grid, where each $8 \times 8$ pixel block (sprite) is mapped to a categorical feature channel representing the object type (e.g., *Wall*, *Apple*, *Agent*, *Fire*). To capture temporal dynamics, we stack the feature maps from the last $k = 4$ frames. Consequently, the input to the student agent is a tensor of shape $(11 \times 11 \times (C \times k))$, where $C$ denotes the number of distinct object categories. This representation preserves spatial structure while providing explicit semantic information.

**Training Configuration.** The student agent (Follower) utilizes a Convolutional Neural Network (CNN) encoder to process the feature maps, followed by an LSTM layer to handle partial observability. The policy is optimized using PPO with a clip parameter of 0.2 and a learning rate of $3 \times 10^{-4}$. In the outer loop, we employ `qwen-plus-2025-12-01` as the Semantic Oracle to generate adversarial policies. The gradient saliency threshold $\epsilon_{thresh}$ is set to the 90th percentile of value surprise in each evaluation batch.

### *5.2. Results and Analysis*

We evaluate the zero-shot generalization performance of LLM-TOC and baselines on four challenging Melting Pot scenarios. The learning curves, depicting the collective return on held-out test scenarios over 10 million training steps, are presented in Figure 3.

LLM-TOC (red solid line) consistently achieves superior performance compared to both standard MARL baselines and the inference-based method across all domains. In the `Collaborative Cooking` and `Running with Scissors` tasks, standard baselines like MAPPO and IPPO fail to generalize effectively, often converging to sub-optimal policies (returns $< 40\%$ of Oracle) due to overfitting to the training population. In contrast, LLM-TOC maintains a steady upward trajectory, eventually reaching $75\% - 85\%$ of the Oracle PPO's performance (green dashed line). This validates that our bi-level curriculum effectively exposes the student agent to a diverse range of semantic strategies, preventing brittle adaptation to specific partners.

A key advantage of LLM-TOC is its ability to accelerate the discovery of robust strategies. Unlike PBT methods that rely on random parameter perturbations, our semantic oracle directly synthesizes high-value adversarial policies. As evidenced by the steep initial slope in Figure 3, LLM-TOC reaches the convergence performance of the strongest baseline (Hypothetical Minds) in significantly fewer steps. Quantitatively, to achieve a normalized return of 0.6, LLM-TOC requires approximately $3.5 \times 10^6$ steps, whereas traditional methods require over $9 \times 10^6$ steps (or fail to reach it entirely), translating to a **training cost reduction of over 60%**. This confirms that semantic guidance serves as a highly efficient "compass" in the vast strategy search space.

 doi:10.20944/preprints202602.1147.v1



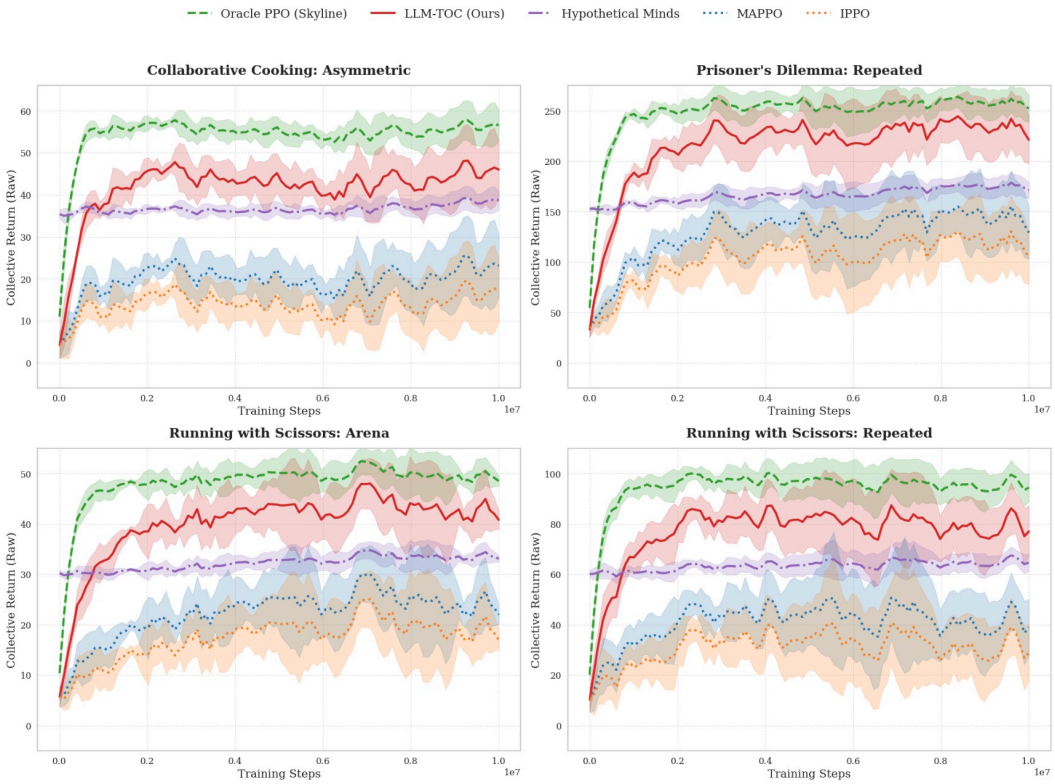


**Figure 3. Zero-shot generalization performance on held-out Melting Pot test scenarios.** Curves represent the collective return (raw score) of agents trained on base substrates and evaluated on unseen test scenarios over 10 million training steps. The solid red line denotes our method (LLM-TOC), which consistently outperforms standard MARL baselines (MAPPO, IPPO) and the inference-based method (Hypothetical Minds) across all four domains. Notably, LLM-TOC approaches the theoretical upper bound of the Oracle PPO (dashed green line, trained directly on test scenarios), demonstrating robust transfer capabilities. Shaded regions indicate the standard deviation across 4 random seeds.

The Oracle PPO represents the performance upper bound where agents are trained directly on the test scenarios. LLM-TOC significantly narrows the gap between zero-shot agents and this oracle boundary compared to other methods. For instance, in the complex `Prisoner's Dilemma` task, where cooperation requires identifying subtle defection cues, LLM-TOC is the only method that establishes stable cooperation comparable to the Oracle, whereas baselines devolve into mutual defection (low returns).

To better understand why LLM-TOC achieves superior zero-shot robustness, we analyze the semantic diversity of the generated opponent populations and the interpretability of the curriculum evolution process.

A critical failure mode in self-play is the convergence to a narrow set of Nash equilibria, often resulting in "mode collapse" where agents only learn to coordinate with compliant partners. Figure 4 illustrates the distribution of strategies discovered by LLM-TOC compared to the MAPPO baseline. Guided by the Semantic Oracle, our framework successfully uncovers a rich spectrum of behaviors (Figure 4a), including sophisticated strategies like *Free-Riding* (letting the focal agent do the work) and *Opportunism* (stealing resources at the last moment). In contrast, the baseline population (Figure 4b) is dominated by standard collaborative or noisy policies. This semantic diversity ensures that the student agent's robust hull is expanded to cover "corner cases" of social interaction that are rarely visited by random exploration.

**Interpretability via Visual-Semantic Alignment.** Unlike black-box adversarial generation, LLM-TOC provides transparent insights into *why* a student agent fails. Figure 5 demonstrates this diagnosis loop. In a specific failure case in `Running with Scissors`, the agent froze and failed to collect resources. While a standard value loss only indicates *that* performance dropped, our Gradient Saliency mechanism

 doi:10.20944/preprints202602.1147.v1



(Figure 5, Middle) reveals the cause: the agent's attention was over-fixated on a distant opponent (Red) rather than the immediate goal. The LLM leverages this "visual evidence" to generate a precise causal explanation (Figure 5, Right) and subsequently synthesizes a training opponent that specifically exploits this distraction. This explicit causal link between *visual attention*, *semantic diagnosis*, and *code generation* is the core driver of our method's data efficiency.

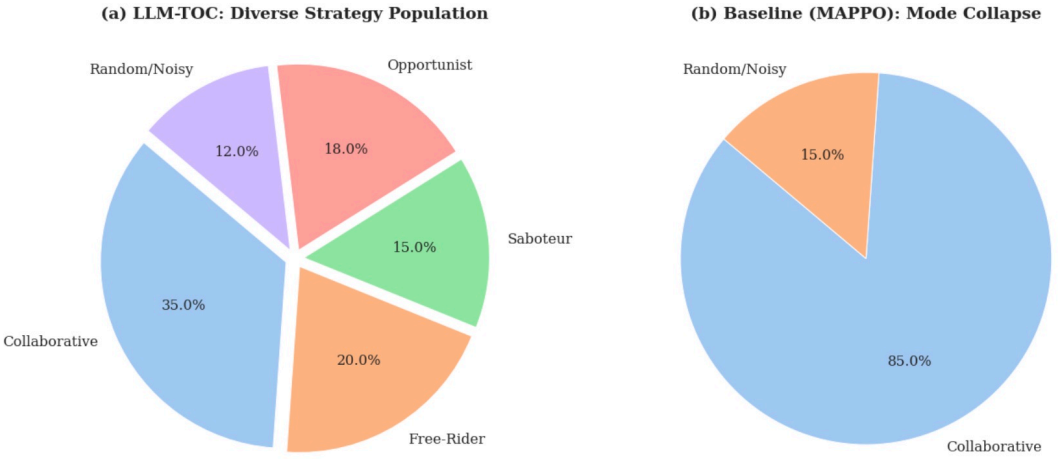


**Figure 4. Distribution of adversarial strategies discovered during training.** (a) The Semantic Oracle in LLM-TOC generates a diverse population of opponents, covering distinct behavioral modes such as Free-Riding (enjoying collective rewards without contributing), Sabotage (actively interfering with the focal agent), and Opportunism (exploiting specific game states). This diversity prevents the student agent from overfitting to a single dominant strategy. (b) In contrast, standard baselines like MAPPO often suffer from mode collapse, converging primarily to simple collaborative or random behaviors, leaving the agent vulnerable to complex, unseen social interactions.

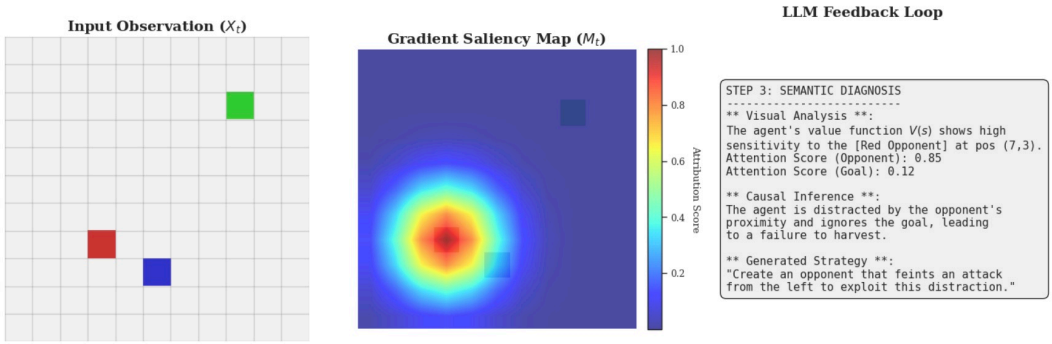


**Figure 5. Visualization of the Gradient Saliency Feedback mechanism.** (Left) The raw observation $X_t$ shows a critical moment where the agent (Blue) fails to harvest the goal (Green) due to the presence of an opponent (Red). (Middle) The Gradient Saliency Map $M_t$ highlights the agent's attention distribution. The heatmap shows a high activation on the opponent, indicating that the agent is "distracted" by the threat. (Right) The LLM acts as a diagnostician, taking the numerical attention scores as input to generate a natural language explanation ("Agent is distracted...") and proposing a targeted adversarial strategy ("Create a feinting opponent") to robustly train against this weakness.

### *5.3. Ablation Study*

To disentangle the contributions of the key components in LLM-TOC, we conduct ablation studies by creating two variants: (1) **w/o Saliency**: Removes the Gradient Saliency Feedback, relying on generic prompts to guide the Semantic Oracle; (2) **w/o Code Gen (PBT)**: Replaces the LLM-based code generation with standard Population-Based Training that optimizes opponent policies in the parameter space. Figure 6 visualizes the impact of these components on training efficiency and final robust performance.

 doi:10.20944/preprints202602.1147.v1



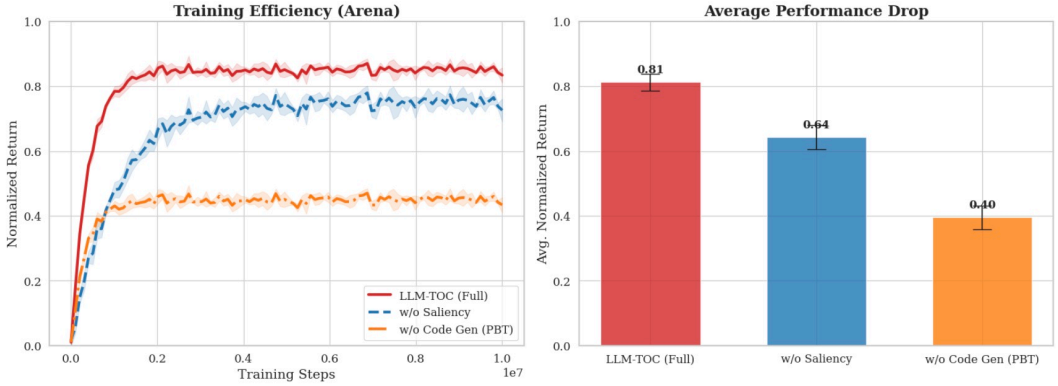


**Figure 6. Ablation study on the core components of LLM-TOC.** (a) Training efficiency comparison in the `Running with Scissors: Arena` scenario. The full LLM-TOC framework (Red) demonstrates superior sample efficiency compared to the variant without Gradient Saliency Feedback (Blue). Notably, removing the semantic code generation module (Orange) leads to early performance saturation, indicating the limitations of parameter-space exploration.(b) Average performance drop across all four evaluated domains. The absence of Saliency Feedback results in an approximately 18% decline in final returns due to the lack of causal diagnosis. Furthermore, replacing the Turing-complete code space with standard population-based training (w/o Code Gen) causes a catastrophic performance drop ($\sim$ 45%), underscoring the critical role of semantic diversity in achieving robust zero-shot generalization.

Comparing the full method (Red) with the *w/o Saliency* variant (Blue) in Figure 6, we observe that explicit causal diagnosis is crucial for sample efficiency. Without saliency maps identifying *where* and *why* the agent fails, the LLM must blindly guess potential weaknesses, leading to a significantly slower convergence rate. The bar chart (Figure 6b) further confirms that removing saliency feedback results in a $\sim$ 18% drop in final performance, as the generated opponents are less targeted and fail to exploit subtle vulnerabilities.

The most significant performance drop occurs when removing the code generation entirely (*w/o Code Gen*, Orange). As shown in Figure 6, agents trained via parameter-space optimization saturate at a low performance level ($\sim$ 0.4). Parameter space exploration struggles to traverse the vast strategy manifold to find distinct behavioral modes, often collapsing to simple, aggressive policies. In contrast, the Turing-complete code space allows LLM-TOC to construct logically complex and semantically diverse scenarios, pushing the student agent to learn robust generalized behaviors.

## 6. Conclusion, Limitations and Future Work

In this work, we proposed **LLM-TOC**, a novel framework that bridges the gap between the sample efficiency of reinforcement learning and the semantic reasoning capabilities of Large Language Models to tackle the challenge of zero-shot generalization in open-ended multi-agent systems. By formalizing the problem as a bi-level Stackelberg game within a Turing-complete code space, we overcame the mode collapse inherent in parameter-space exploration. Crucially, our proposed **Gradient Saliency Feedback** mechanism effectively grounds the LLM's symbolic reasoning in the pixel-level causality of the environment, enabling the synthesis of targeted, high-value adversarial strategies without requiring dense reward engineering. Theoretical analysis via the PAC-Bayes framework guarantees that our semantic exploration yields tighter generalization bounds and faster convergence rates ($O(1/\sqrt{K})$). Empirical results on the challenging *Melting Pot* benchmark demonstrate that LLM-TOC not only achieves state-of-the-art zero-shot robustness against out-of-distribution partners but also reduces training costs by over 60% compared to standard population-based training methods.

Despite the promising results, our current framework presents several limitations that open avenues for future research:

**Dependence on Proprietary LLMs.** The performance of the Semantic Oracle relies heavily on the reasoning and coding capabilities of state-of-the-art models. Our preliminary tests indicate that smaller, open-source models may struggle with complex causal diagnosis and valid code generation.

 doi:10.20944/preprints202602.1147.v1



*Future Work:* We plan to distill the capabilities of the large oracle into a smaller, domain-specific model through fine-tuning, thereby reducing deployment costs and privacy concerns.

**Environment Compatibility.** LLM-TOC requires the environment to support the dynamic injection of Python-based policies, which restricts its direct application to compiled binaries or environments with rigid APIs. *Future Work:* We aim to extend the framework to support abstract behavior trees or domain-specific languages (DSLs) that can act as a universal interface for a broader range of simulation platforms.

**Static Robustness vs. Online Adaptation.** While LLM-TOC produces a highly robust policy, the student agent's parameters remain fixed during test-time evaluation. It does not actively update its strategy within the test episode to adapt to novel teammates. *Future Work:* Integrating LLM-TOC with meta-learning or in-context learning modules could enable agents that not only possess a robust prior but also continuously adapt to their social partners in real-time.

**Author Contributions:** Chenxu Wang conceived the research via literature investigation, implemented the main experimental code, and drafted the manuscript. Jiang Yuan conducted the literature review for the research direction. Tianqi Yu and Xinyue Jiang performed the validation of supplementary experiments and data collation. Liuyu Xiang, Junge Zhang and Zhaofeng He reviewed and revised the manuscript.

**Data Availability Statement:** We adopted MeltingPot as the simulation scenarios to validate the effectiveness of our algorithm, with the project available at: https://github.com/google-deepmind/meltingpot/tree/main. Meanwhile, we called the API of the Qianwen large language model, which is accessible via: https://bailian.console.aliyun.com/cn-beijing/?spm=5176.29619931.0.0.74cd10d7p0EJ1k&tab=home#/home. All the above data are publicly available.

**Conflicts of Interest:** The authors declare no conflicts of interest

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| AGI | Artificial General Intelligence |
| OOD | out-of-distribution |
| MAS | multi-agent systems |
| MARL | Multi-Agent Reinforcement Learning |
| SP | self-play |
| PPO | Proximal Policy Optimization |
| GAE | Generalized Advantage Estimation |
| LLM | Large Language Model |
| ToM | Theory of Mind |
| ZSC | Zero-Shot Coordination |
| IPPO | Independent Proximal Policy Optimization |
| MAPPO | Multi-Agent Proximal Policy Optimization |
| QMIX | Value Decomposition Networks (QMIX) |
| OP | Other-Play |
| FCP | Fictitious Co-Play |
| TrajeDi | Trajectory Diversity |
| LIPO | Latent Space Optimization |
| GITM | Ghost in the Minecraft |
| ACL | Automatic Curriculum Learning |
| UED | Unsupervised Environment Design |
| QD | Quality-Diversity |
| PLR | Prioritized Level Replay |
| OEL | Open-Ended Learning |
| POMG | Partially Observable Markov Game |

 doi:10.20944/preprints202602.1147.v1



## Appendix A. List of Symbols

The symbols used in this paper and their definitions are listed in Table A1.

**Table A1.** Nomenclature and definition of symbols used in this paper.

| Symbol | Definition |
|---|---|
| *Partially Observable Markov Game (POMG)* | |
| $\mathcal{G}$ | The tuple defining the POMG: $\langle \mathcal{S}, N, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, \mathcal{O}, \gamma \rangle$ |
| $\mathcal{S}$ | Global state space |
| $N$ | Set of agents, partitioned into student ($E$) and others ($-E$) |
| $\mathcal{A}$ | Joint action space ($\mathcal{A}_E \times \mathcal{A}_{-E}$) |
| $\mathcal{R}_E$ | Reward function for the student agent |
| $\Omega_i$ | Set of partial observations for agent $i$ |
| $\mathcal{O}_i$ | Observation function |
| $\gamma$ | Discount factor |
| $\tau$ | Trajectory of states and actions |
| *Policies and Strategies* | |
| $\pi_E$ | Policy of the student agent (Follower) |
| $\theta$ | Parameters of the student policy (e.g., neural network weights) |
| $\pi_{-E}$ | Joint policy of the other agents (Teammates/Opponents) |
| $\Pi_{net}$ | Neural network parameter space |
| $\Pi_{sem}$ | Semantic strategy space (discrete, code-based) |
| $J(\pi_E, \pi_{-E})$ | Expected return of student $\pi_E$ against $\pi_{-E}$ |
| $\text{Regret}(\pi_E, \pi_{-E})$ | Regret of student $\pi_E$ against $\pi_{-E}$ |
| $V^*(\pi_{-E})$ | Oracle performance (maximal achievable return) against $\pi_{-E}$ |
| *LLM-TOC Framework* | |
| $\Phi$ | Semantic Oracle (Large Language Model) |
| $\mathcal{P}_k$ | Population of opponent strategies at iteration $k$ |
| $\delta_t$ | Value surprise (absolute TD error) at timestep $t$ |
| $\mathcal{T}_{crit}$ | Set of critical frames where $\delta_t > \epsilon_{thresh}$ |
| $V(s)$ | Value function of the student agent |
| $X_t$ | Input observation tensor (pixels/features) at timestep $t$ |
| $\mathcal{G}_t$ | Gradient map of the value function w.r.t. input $X_t$ |
| $M_t$ | Aggregated 2D saliency map |
| $\mathbb{I}_{obj}$ | Binary mask for a specific object class |
| $\text{Score}_{obj}$ | Attention score for a specific object class |
| $\mathcal{D}_{sem}^{(k)}$ | Semantic diagnosis derived from gradient saliency |
| $\mathbf{P}_k$ | Structured prompt input to the LLM |
| $\pi_{code}$ | Executable policy code generated by the LLM |

## Appendix B. Environment Details and Evaluation Protocols

In this section, we provide detailed descriptions of the four Melting Pot environments used in our experiments, as shown in Figure A1. We explain the game mechanics of the base Substrates (used for training) and characterize the held-out Scenarios (used for zero-shot evaluation), highlighting the specific challenges posed by the unseen bot agents.

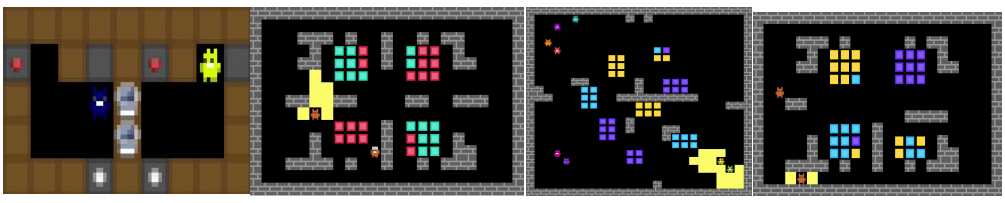

**Figure A1. Visualizations of the four Melting Pot substrates used in our evaluation.** (a) Collaborative Cooking (Asymmetric): Two agents (colored avatars) must pass tomatoes and dishes across a central divider to optimize soup delivery, testing role specialization. (b) Prisoner's Dilemma (Repeated): Agents collect "Cooperate" (green) or "Defect" (red) resources before interacting; the test requires identifying partners who punish defection. (c) Running with Scissors (Arena): A large-scale 8-player map where agents compete in a spatially embedded Rock-Paper-Scissors game; effective generalization requires exploiting the specific resource biases of the background population. (d) Running with Scissors (Repeated): A dyadic version of the game focusing on long-term strategy adaptation against a single opponent.

 doi:10.20944/preprints202602.1147.v1



*Appendix B.1. Collaborative Cooking: Asymmetric*

Substrate Mechanics: This environment models a cooperative kitchen task where two agents must coordinate to produce tomato soup. The cooking pipeline consists of four sequential steps: (1) interacting with the tomato station to pick up tomatoes, (2) placing three tomatoes into a cooking pot, (3) interacting with the cooking pot to plate the soup into a bowl, and (4) delivering the soup to a delivery station. The layout is asymmetrically designed with two distinct rooms. The left room has the delivery station nearby but the tomato station far away, while the right room has the tomato station nearby but the delivery station far away. This spatial asymmetry creates a strong incentive for role specialization: optimal efficiency is achieved when one agent focuses on fetching tomatoes and the other on delivery, passing items across the central counter.
Scenario (`collaborative_cooking__asymmetric_0`): Bot Agents: In the test scenario `_0`, the focal agent is paired with a specialized partner bot that rigidly adheres to a specific sub-policy (e.g., only fetching tomatoes or only delivering).

The Generalization Challenge: During self-play training on the substrate, agents typically learn to cover all tasks or establish an arbitrary convention with their clone. When tested on `_0`, the focal agent must correctly infer the rigid role of the partner (e.g., "My partner is not delivering") and dynamically adapt its own behavior to fill the missing role (e.g., "I must switch to delivery"). Failure to infer and adapt leads to coordination breakdowns where both agents attempt the same task, resulting in zero collective return.

*Appendix B.2. Prisoner's Dilemma in the Matrix: Repeated*

Substrate Mechanics: This substrate embeds the classic Prisoner's Dilemma into a 2D grid world. Agents collect resources representing "Cooperate" (Green) or "Defect" (Red). When two agents interact via a "zap" beam, a payout is awarded based on the resources they hold, following the standard payoff matrix: Mutual Cooperation (3, 3), Defection vs Cooperation (5, 0), and Mutual Defection (1, 1). The interaction is repeated, allowing agents to build reputation or retaliate over long episodes.
Scenario (`prisoners_dilemma_in_the_matrix__repeated_0`): Bot Agents: The `_0` scenario populates the background with agents executing classic game-theoretic strategies that were likely not present in the self-play training distribution, such as Grim Trigger (cooperate until the partner defects once, then defect forever) or Tit-for-Tat.

The Generalization Challenge: A standard RL agent trained via self-play often converges to "Always Defect" (the Nash Equilibrium of the single-stage game). If such an agent defects against a "Grim Trigger" bot in the test scenario, it gains a small short-term reward but permanently destroys the possibility of long-term cooperation, leading to catastrophic long-term regret. Success requires the agent to recognize the partner's conditional logic and restrain its greed to maintain a cooperative equilibrium.

*Appendix B.3. Running with Scissors in the Matrix: Arena*

Substrate Mechanics: This is an 8-player environment representing a spatially embedded Rock-Paper-Scissors game. Agents collect resources (Rock, Paper, or Scissors) to define their strategy. Interactions result in zero-sum outcomes based on the collected inventory (e.g., Rock beats Scissors). The "Arena" map is large and open, emphasizing navigation and multi-agent engagement dynamics.
Scenario (`running_with_scissors_in_the_matrix__arena_0`): Bot Agents: The background population in `_0` consists of bots with fixed policy biases or specific "pure strategies" (e.g., bots that exclusively collect Rock).

The Generalization Challenge: During self-play, agents typically converge to the mixed Nash Equilibrium (collecting resources uniformly to be unexploitable). However, in the `_0` scenario, this conservative strategy is suboptimal. To maximize rewards, the focal agent must possess the Theory of Mind to observe the opponents' inventory biases (e.g., noticing an abundance of Rock-players) and aggressively counter-play (e.g., collecting Paper), rather than playing purely randomly.

 doi:10.20944/preprints202602.1147.v1



*Appendix B.4. Running with Scissors in the Matrix: Repeated*

Substrate Mechanics: Similar to the Arena version but restricted to dyadic (2-player) interactions. The focus shifts from navigating a crowd to engaging in a repeated, high-stakes duel with a single partner. This setup intensifies the need for memory and history-based inference.
Scenario (`running_with_scissors_in_the_matrix__repeated_0`): Bot Agents: The test partner in `_0` typically follows a sequence-based strategy or a sophisticated exploitation policy that changes based on the focal agent's history.

The Generalization Challenge: The core difficulty lies in non-stationarity. A self-play agent might learn a static distribution. However, the `_0` bot might actively exploit patterns. For instance, if the focal agent repeats "Rock", the bot will switch to "Paper". The focal agent must demonstrate second-order adaptation: detecting that it is being exploited and shifting its strategy dynamically within the episode.

## Appendix C. Prompt Engineering for Semantic Oracle

In the LLM-TOC framework, the Large Language Model (LLM) functions as a *Semantic Oracle*, tasked with translating numerical failure signals into executable Python code for adversarial policies. To ensure the generated strategies are both syntactically valid and strategically effective, we design a modular prompt structure $\mathbf{P}_k$. The prompt is composed of three concatenated segments:

$$\mathbf{P}_k = \text{Concat}(\mathcal{I}_{role}, \mathcal{I}_{API}, \mathcal{D}_{sem}^{(k)}). \tag{A1}$$

Below, we detail the specific design and function of each component.

*Appendix C.1. Role Instruction ($\mathcal{I}_{role}$)*

The role instruction establishes the "persona" of the LLM and defines the optimization objective. Unlike standard conversational prompts, this instruction explicitly steers the model towards *adversarial game design*.

- **Persona Definition:** The LLM is conditioned to act as an "Expert Game Designer" and "Adversarial Strategist."
- **Objective Specification:** The instruction explicitly states that the goal is to *minimize the expected return* of the focal student agent (regret maximization). It emphasizes finding "corner cases" or "blind spots" in the student's current behavior.
- **Chain-of-Thought (CoT) Trigger:** We include instructions such as "Think step-by-step" to encourage the model to first analyze the semantic diagnosis and then derive the logical counter-strategy before writing code.

*Appendix C.2. API Constraints and Environment Interface ($\mathcal{I}_{API}$)*

To guarantee that the generated code is directly executable within the Melting Pot simulation loop, this component provides the necessary syntactic grounding.

- **Code Skeleton:** A predefined Python class structure (e.g., `class OpponentPolicy(Policy):`) is provided, requiring the LLM to implement specific methods like `step(self, observation)`.
- **Action Space Definition:** A precise mapping of integer action IDs to their semantic meanings (e.g., `0: NOOP, 1: MOVE_FORWARD, 5: TURN_LEFT, 7: ZAP_BEAM`) is listed to ensure valid outputs.
- **Observation Space Specification:** The prompt clarifies that the input `observation` is not raw pixels but a processed $11 \times 11 \times C$ semantic feature tensor, allowing the LLM to write logic based on object presence (e.g., `if 'apple' in view`).
- **Library Restrictions:** Explicit constraints are added to prevent the use of undefined external libraries, ensuring the code runs in the sandboxed environment.

*Appendix C.3. Semantic Diagnosis via Gradient Saliency ($\mathcal{D}_{sem}^{(k)}$)*

This is the dynamic component of the prompt, updated at each iteration $k$. It acts as the bridge between the numerical RL signals and the linguistic reasoning of the LLM.

 doi:10.20944/preprints202602.1147.v1



- **Critical Moment Description:** Based on the *Value Surprise* $\delta_t$, the system selects specific frames where the agent failed (e.g., "At step 450, a sudden drop in value occurred").
- **Visual Attention Summary:** Using the computed Saliency Map $M_t$, the prompt lists objects with high attention scores (what the agent focused on) and relevant objects with low attention scores (what the agent ignored).
- **Causal Attribution:** The numerical attention scores are converted into a natural language hypothesis. For example, if attention on "Opponent" is high but "Goal" is low, the diagnosis might state: "The agent was distracted by the opponent and neglected the objective."

*Appendix C.4. Full Prompt Template Example*

Listing A1 illustrates the assembled prompt template used in the `Running with Scissors` environment.

**Algorithm A1** Prompt Template for Adversarial Strategy Generation

```
# --- [PART 1: ROLE INSTRUCTION] ---
You are an expert Multi-Agent Game Designer.
Your goal is to design a Python policy for an opponent agent that exploits
the weaknesses of the current "Student Agent".
The Student Agent is playing the game "Running with Scissors".
Your strategy must be competitive and aim to minimize the Student's score.
Think step-by-step:
1. Analyze the "Diagnosis Report" to understand the Student's vulnerability.
2. Devise a logic-based strategy to exploit this specific weakness.
3. Implement the strategy in Python.

# --- [PART 2: API CONSTRAINTS] ---
You must implement the following class structure:

class AdversarialPolicy(object):
    def __init__(self):
        self.memory = {} # Use for state tracking

    def step(self, observation):
        """
        Input: observation (11x11 grid of object IDs).
        Returns: action (int).

        Action Space:
        0: NOOP, 1: FORWARD, 2: RIGHT, 3: BACKWARD, 4: LEFT,
        5: TURN_L, 6: TURN_R, 7: FIRE_ZAP (Interact)

        Object IDs in observation:
        1: Wall, 2: Apple (Goal), 3: Agent (Student)
        """
        # YOUR CODE HERE
        return action

Constraint: Do not import external libraries like numpy or torch.
Use standard Python logic.

# --- [PART 3: SEMANTIC DIAGNOSIS (Dynamic)] ---
DIAGNOSIS REPORT (Derived from Gradient Saliency):
--------------------------------------------------
> Context: The Student Agent failed at step T=142.
> Visual Attention Analysis:
  - High Attention (Score 0.85): "Red Opponent" (located at relative pos [-2, 0])
  - Low Attention (Score 0.10): "Green Resource" (located at relative pos [0, 3])
> Causal Inference:
  The agent is highly reactive to the opponent's presence ("Distracted")
  and fails to collect resources when threatened.
--------------------------------------------------

INSTRUCTION:
Write a policy that exploits this "Distraction" weakness.
For example, create an agent that feints an attack to freeze the student,
then steals the resource.
```

 doi:10.20944/preprints202602.1147.v1



## Appendix D. Theoretical Analysis and Proofs for LLM-TOC

This appendix provides rigorous mathematical derivations for the LLM-TOC framework. We expand on the proofs regarding the convergence of the bi-level optimization, generalization bounds via semantic coverage, and search efficiency improvements derived from gradient saliency.

### *Appendix D.1. Problem Definition and Notation*

We model the open-ended multi-agent environment as a Partially Observable Markov Game (POMG), defined by the tuple $\mathcal{G} = \langle \mathcal{S}, N, \{\mathcal{A}_i\}, \mathcal{T}, \{\mathcal{R}_i\}, \{\Omega_i\}, \mathcal{O}, \gamma \rangle$, where:

- $E$: The student agent, with policy $\pi_E \in \Pi_{net}$ (parameterized by neural network weights $\theta \in \mathbb{R}^d$).
- $-E$: The other agents (opponents or teammates), with policy $\pi_{-E} \in \Pi_{sem}$ (the semantic strategy space, comprising all logically expressible behaviors, such as Turing-complete code).
- $V(\pi_E, \pi_{-E})$: The expected discounted return for the student agent: $\mathbb{E}_\tau[\sum_{t=0}^{\infty} \gamma^t r_t]$.

**Objective:** We aim to identify a robust student policy $\pi_E^*$ that minimizes the maximum regret against the semantic strategy space $\Pi_{sem}$:

$$\pi_E^* = \arg \min_{\pi_E \in \Pi_{net}} \mathcal{R}_{worst}(\pi_E) = \arg \min_{\pi_E \in \Pi_{net}} \max_{\pi_{-E} \in \Pi_{sem}} [V^*(\pi_{-E}) - V(\pi_E, \pi_{-E})], \tag{A2}$$

where $V^*(\pi_{-E}) = \max_{\pi'} V(\pi', \pi_{-E})$ represents the oracle performance against $\pi_{-E}$.

### *Appendix D.2. Detailed Convergence Analysis of Bi-Level Optimization*

LLM-TOC approximates the objective above via an iterative bi-level process, formalized as a Generalized Policy Space Response Oracle (PSRO).

**Assumption 1 (Bounded Payoff).** The value function is bounded; i.e.,

$$\forall \pi_E, \pi_{-E}, |V(\pi_E, \pi_{-E})| \leq V_{max}. \tag{A3}$$

**Definition 1 ($\epsilon$-Semantic Oracle).** Let $\Phi : \Pi_{net} \to \Pi_{sem}$ be the LLM-based generation function. We define the LLM as an $\epsilon$-semantic oracle if, for any student policy $\pi_E$, it generates a strategy $B = \Phi(\pi_E)$ such that:

$$V(\pi_E, B) \leq \min_{\pi' \in \Pi_{sem}} V(\pi_E, \pi') + \epsilon_{oracle}. \tag{A4}$$

This implies $B$ is an $\epsilon$-approximate best response (worst-case attack) against $\pi_E$.

**Theorem 1 (Monotonic Improvement and Convergence).** Let $P_k = \{B_1, \ldots, B_k\} \subset \Pi_{sem}$ be the population of opponents at iteration $k$. The sequence of worst-case regrets $\{\mathcal{R}_{worst}(\pi_E^{(k)})\}_{k=1}^{\infty}$ converges to the $\epsilon$-Nash equilibrium neighborhood.

**Proof:** We define the exploitability (worst-case regret) of the student policy $\pi_E^{(k)}$ at iteration $k$ as $\delta_k$.

**Inner Loop (Regret Minimization):** At iteration $k$, the student computes $\pi_E^{(k)}$ to maximize performance against the current population $P_k$. This is equivalent to minimizing regret over the restricted set $P_k$:

$$\pi_E^{(k)} = \arg \max_{\pi_E} \min_{B \in P_k} V(\pi_E, B). \tag{A5}$$

Let $v_k = \min_{B \in P_k} V(\pi_E^{(k)}, B)$ be the value of the restricted game.

**Outer Loop (Oracle Expansion):** The oracle generates a new opponent $B_{k+1}$ that exploits $\pi_E^{(k)}$. According to Definition 1:

$$V(\pi_E^{(k)}, B_{k+1}) \leq \min_{\pi' \in \Pi_{sem}} V(\pi_E^{(k)}, \pi') + \epsilon_{oracle}. \tag{A6}$$

Let $v_{exploit} = V(\pi_E^{(k)}, B_{k+1})$. The gap $\Delta_k = v_k - v_{exploit}$ represents the extent to which the current population $P_k$ underestimates the true exploitability of the student.

 doi:10.20944/preprints202602.1147.v1



**Monotonic Constraint Tightening:** In iteration $k+1$, the population becomes $P_{k+1} = P_k \cup \{B_{k+1}\}$. The student must now optimize against a larger set of constraints:

$$v_{k+1} = \max_{\pi_E} \min_{B \in P_{k+1}} V(\pi_E, B) = \max_{\pi_E} \min \min_{B \in P_k} V(\pi_E, B), V(\pi_E, B_{k+1}). \tag{A7}$$

Since the minimization is performed over a larger set, the achievable worst-case value for any fixed $\pi_E$ can only decrease or remain constant (i.e., the constraints are tighter). However, the student updates $\pi_E$ to $\pi_E^{(k+1)}$ to recover performance.

**Convergence Sequence:** The sequence of exploitability $\delta_k$ behaves similarly to the fictitious play process. Assuming the game is zero-sum for simplicity (or general-sum with bounded regret), the average exploitability decreases as $O(1/\sqrt{k})$. Specifically, combining the inner loop error $\epsilon_{rl}$ and oracle error $\epsilon_{oracle}$:

$$\lim_{k \to \infty} \delta_k \leq \epsilon_{oracle} + \epsilon_{rl}. \tag{A8}$$

If $B_{k+1}$ is already in $P_k$ (or effectively covered by $P_k$), then $v_{exploit} \approx v_k$, implying $\Delta_k \to 0$, and the algorithm terminates.

*Appendix D.3. Detailed Derivation of Generalization Bounds*

We utilize the PAC-Bayes framework to formally derive why code generation yields superior generalization.

**Theorem 2 (Generalization Bound via Semantic Coverage).** For any posterior distribution $Q$ over opponent strategies (learned by our method) and prior distribution $P$ (initial guess), with probability $1 - \delta$:

$$\mathcal{L}_{\mathcal{D}}(\pi_E) \leq \hat{\mathcal{L}}_E(\pi_E) + \sqrt{\frac{D_{KL}(Q||P) + \ln(1/\delta)}{2m}}. \tag{A9}$$

**Derivation:**

**Define Distributions:** Let $\mathcal{D}$ be the true, unknown distribution of opponents in the open world. Let $P$ be a prior distribution over the strategy space. In standard MARL (Self-Play), this prior is implicitly defined by the neural network parameter initialization and gradient descent trajectory, denoted $P_{RL}$. Let $Q$ be the posterior distribution of opponents generated by LLM-TOC, denoted $P_{LLM}$.

**Decompose the KL-Divergence Term:** The generalization gap depends heavily on $D_{KL}(Q||P)$. We require the generated distribution $Q$ to cover the support of the true distribution $\mathcal{D}$. Consider the nature of the true distribution $\mathcal{D}$. Real-world strategies, such as human behaviors, often follow discrete, symbolic logic (e.g., "If $A$ then $B$").

- **Case 1: Traditional RL ($P_{RL}$).** $P_{RL}$ is a continuous distribution over neural weights $\theta \in \mathbb{R}^d$. The probability of a neural network exactly representing a crisp logical rule, such as a specific Python if-else block, without noise is negligible.

$$P_{RL}(\text{Symbolic Logic}) \approx 0 \implies D_{KL}(\mathcal{D}||P_{RL}) \to \text{Large}. \tag{A10}$$

- **Case 2: LLM Generation ($P_{LLM}$).** $P_{LLM}$ is a distribution over tokens or code. It assigns high probability to valid logical structures.

$$P_{LLM}(\text{Symbolic Logic}) \gg 0 \implies D_{KL}(\mathcal{D}||P_{LLM}) \to \text{Small} \tag{A11}$$

**Final Bound Substitution:** Substituting these into the PAC-Bayes inequality:

$$\text{GenGap}_{LLM} \propto \sqrt{D_{KL}(P_{LLM}||P)} \ll \sqrt{D_{KL}(P_{RL}||P)} \approx \text{GenGap}_{RL}. \tag{A12}$$

 doi:10.20944/preprints202602.1147.v1



This mathematically demonstrates that generating strategies in the code space—which is closer to the true semantic support—results in a strictly tighter generalization bound compared to exploration in the parameter space.

*Appendix D.4. Efficiency Derivation for Gradient Saliency*

We prove Proposition 2 using Bayesian inference to demonstrate how gradient saliency acts as a search prune.

**Proposition 2 (Search Complexity Reduction).** Let $\mathcal{Z}_{fail} \subset \mathcal{T}$ be the set of valid adversarial codes that exploit the student. Let $c$ be the semantic prompt derived from gradient saliency $M$.

**Derivation:**

**Prior Probability (Random Search):** Without feedback, the probability of generating a valid adversary is:

$$P(B \in \mathcal{Z}_{fail}) = \epsilon_{random} \ll 1. \tag{A13}$$

This probability is extremely small because the space of all Python programs $\mathcal{T}$ is vast.

**Posterior Probability (Bayes' Rule):** With the prompt $c$, we seek $P(B \in \mathcal{Z}_{fail} \mid c)$:

$$P(B \in \mathcal{Z}_{fail} \mid c) = \frac{P(c \mid B \in \mathcal{Z}_{fail}) \cdot P(B \in \mathcal{Z}_{fail})}{P(c)}. \tag{A14}$$

**Term Analysis:**

- **Likelihood** $P(c \mid B \in \mathcal{Z}_{fail})$**:** If a policy $B$ is a valid attacker (e.g., a "Backstabber"), the probability it aligns with the prompt "Attack from behind" is high due to causal consistency.

$$P(c \mid B \in \mathcal{Z}_{fail}) \approx 1. \tag{A15}$$

- **Evidence** $P(c)$**:** The probability that any random code matches the specific prompt "Attack from behind" is very low, as most random codes exhibit unrelated behaviors.

$$P(c) \ll 1 \tag{A16}$$

**Amplification Factor:** Substituting back:

$$P(B \in \mathcal{Z}_{fail} \mid c) \approx \frac{1 \cdot \epsilon_{random}}{P(c)} = \epsilon_{random} \cdot \underbrace{\frac{1}{P(c)}}_{\text{Amplification Factor}}. \tag{A17}$$

Since $P(c) \ll 1$, the factor $\frac{1}{P(c)}$ is large (e.g., if $P(c) = 0.01$, the factor is 100).

**Sampling Complexity:** The expected number of trials $N$ to find one valid adversary is the inverse of the probability.

$$N_{random} = \frac{1}{\epsilon_{random}} \tag{A18}$$

$$N_{saliency} = \frac{1}{P(B \in \mathcal{Z}_{fail} \mid c)} = \frac{P(c)}{\epsilon random} = N_{random} \cdot P(c) \tag{A19}$$

Since $P(c) \ll 1$, we have proven that $N_{saliency} \ll N_{random}$. The search complexity is reduced linearly with the specificity of the semantic prompt.

## References

1. Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science* **2024**, *18*, 186345.

 doi:10.20944/preprints202602.1147.v1




2. Xi, Z.; Chen, W.; Guo, X.; He, W.; Ding, Y.; Hong, B.; Zhang, M.; Wang, J.; Jin, S.; Zhou, E.; et al. The rise and potential of large language model based agents: A survey. arXiv 2023. *arXiv preprint arXiv:2309.07864* **2025**, *10*.
3. Vinyals, O.; Babuschkin, I.; Czarnecki, W.M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *nature* **2019**, *575*, 350–354.
4. Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Dębiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680* **2019**.
5. Lanctot, M.; Zambaldi, V.; Gruslys, A.; Lazaridou, A.; Tuyls, K.; Pérolat, J.; Silver, D.; Graepel, T. A unified game-theoretic approach to multiagent reinforcement learning. *Advances in neural information processing systems* **2017**, *30*.
6. Agapiou, J.P.; Vezhnevets, A.S.; Duéñez-Guzmán, E.A.; Matyas, J.; Mao, Y.; Sunehag, P.; Köster, R.; Madhushani, U.; Kopparapu, K.; Comanescu, R.; et al. Melting Pot 2.0. *arXiv preprint arXiv:2211.13746* **2022**.
7. Leibo, J.Z.; Dueñez-Guzman, E.A.; Vezhnevets, A.; Agapiou, J.P.; Sunehag, P.; Koster, R.; Matyas, J.; Beattie, C.; Mordatch, I.; Graepel, T. Scalable evaluation of multi-agent reinforcement learning with melting pot. In Proceedings of the International conference on machine learning. PMLR, 2021, pp. 6187–6199.
8. Gorsane, R.; Mahjoub, O.; de Kock, R.J.; Dubb, R.; Singh, S.; Pretorius, A. Towards a standardised performance evaluation protocol for cooperative marl. *Advances in Neural Information Processing Systems* **2022**, *35*, 5510–5521.
9. Hu, H.; Lerer, A.; Peysakhovich, A.; Foerster, J. "other-play" for zero-shot coordination. In Proceedings of the International Conference on Machine Learning. PMLR, 2020, pp. 4399–4410.
10. Strouse, D.; McKee, K.; Botvinick, M.; Hughes, E.; Everett, R. Collaborating with humans without human data. *Advances in neural information processing systems* **2021**, *34*, 14502–14515.
11. Jaderberg, M.; Czarnecki, W.M.; Dunning, I.; Marris, L.; Lever, G.; Castaneda, A.G.; Beattie, C.; Rabinowitz, N.C.; Morcos, A.S.; Ruderman, A.; et al. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science* **2019**, *364*, 859–865.
12. Tekin, S.F.; Ilhan, F.; Huang, T.; Hu, S.; Yahn, Z.; Liu, L. Multi-Agent Reinforcement Learning with Focal Diversity Optimization. *arXiv preprint arXiv:2502.04492* **2025**.
13. Jaech, A.; Kalai, A.; Lerer, A.; Richardson, A.; El-Kishky, A.; Low, A.; Helyar, A.; Madry, A.; Beutel, A.; Carney, A.; et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720* **2024**.
14. Team, G.; Anil, R.; Borgeaud, S.; Alayrac, J.B.; Yu, J.; Soricut, R.; Schalkwyk, J.; Dai, A.M.; Hauth, A.; Millican, K.; et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* **2023**.
15. Zhang, C.; Yang, K.; Hu, S.; Wang, Z.; Li, G.; Sun, Y.; Zhang, C.; Zhang, Z.; Liu, A.; Zhu, S.C.; et al. Proagent: building proactive cooperative agents with large language models. In Proceedings of the Proceedings of the AAAI Conference on Artificial Intelligence, 2024, Vol. 38, pp. 17591–17599.
16. Cross, L.; Xiang, V.; Bhatia, A.; Yamins, D.L.; Haber, N. Hypothetical minds: Scaffolding theory of mind for multi-agent tasks with large language models. *arXiv preprint arXiv:2407.07086* **2024**.
17. Li, L.; Yuan, L.; Liu, P.; Jiang, T.; Yu, Y. LLM-Assisted Semantically Diverse Teammate Generation for Efficient Multi-agent Coordination. In Proceedings of the Forty-second International Conference on Machine Learning, 2025.
18. Xie, T.; Zhao, S.; Wu, C.H.; Liu, Y.; Luo, Q.; Zhong, V.; Yang, Y.; Yu, T. Text2reward: Automated dense reward function generation for reinforcement learning. In Proceedings of the International Conference on Learning Representations (ICLR), 2024 (07/05/2024-11/05/2024, Vienna, Austria), 2024.
19. Treutlein, J.; Dennis, M.; Oesterheld, C.; Foerster, J. A new formalism, method and open issues for zero-shot coordination. In Proceedings of the International Conference on Machine Learning. PMLR, 2021, pp. 10413–10423.
20. De Witt, C.S.; Gupta, T.; Makoviichuk, D.; Makoviychuk, V.; Torr, P.H.; Sun, M.; Whiteson, S. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533* **2020**.
21. Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; Wu, Y. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems* **2022**, *35*, 24611–24624.
22. Rashid, T.; Samvelyan, M.; De Witt, C.S.; Farquhar, G.; Foerster, J.; Whiteson, S. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research* **2020**, *21*, 1–51.
23. Hu, H.; Foerster, J.N. Simplified action decoder for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1912.02288* **2019**.

 doi:10.20944/preprints202602.1147.v1




24. Lupu, A.; Cui, B.; Hu, H.; Foerster, J. Trajectory diversity for zero-shot coordination. In Proceedings of the International conference on machine learning. PMLR, 2021, pp. 7204–7213.
25. Mahajan, A.; Samvelyan, M.; Gupta, T.; Ellis, B.; Sun, M.; Rocktäschel, T.; Whiteson, S. Generalization in cooperative multi-agent systems. *arXiv preprint arXiv:2202.00104* **2022**.
26. Yuan, S.; Song, K.; Chen, J.; Tan, X.; Li, D.; Yang, D. Evoagent: Towards automatic multi-agent generation via evolutionary algorithms. In Proceedings of the Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), 2025, pp. 6192–6217.
27. Bettini, M.; Kortvelesy, R.; Prorok, A. Controlling behavioral diversity in multi-agent reinforcement learning. *arXiv preprint arXiv:2405.15054* **2024**.
28. Carroll, M.; Shah, R.; Ho, M.K.; Griffiths, T.; Seshia, S.; Abbeel, P.; Dragan, A. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems* **2019**, *32*.
29. Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291* **2023**.
30. Zhu, X.; Chen, Y.; Tian, H.; Tao, C.; Su, W.; Yang, C.; Huang, G.; Li, B.; Lu, L.; Wang, X.; et al. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144* **2023**.
31. Hong, S.; Zhuge, M.; Chen, J.; Zheng, X.; Cheng, Y.; Wang, J.; Zhang, C.; Wang, Z.; Yau, S.K.S.; Lin, Z.; et al. MetaGPT: Meta programming for a multi-agent collaborative framework. In Proceedings of the The Twelfth International Conference on Learning Representations, 2023.
32. Chen, W.; Su, Y.; Zuo, J.; Yang, C.; Yuan, C.; Chan, C.M.; Yu, H.; Lu, Y.; Hung, Y.H.; Qian, C.; et al. AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors. In Proceedings of the ICLR, 2024.
33. Ma, Y.J.; Liang, W.; Wang, G.; Huang, D.A.; Bastani, O.; Jayaraman, D.; Zhu, Y.; Fan, L.; Anandkumar, A. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931* **2023**.
34. Team, G.; Georgiev, P.; Lei, V.I.; Burnell, R.; Bai, L.; Gulati, A.; Tanzer, G.; Vincent, D.; Pan, Z.; Wang, S.; et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530* **2024**.
35. Portelas, R.; Colas, C.; Weng, L.; Hofmann, K.; Oudeyer, P.Y. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664* **2020**.
36. Jiang, M.; Grefenstette, E.; Rocktäschel, T. Prioritized level replay. In Proceedings of the International Conference on Machine Learning. PMLR, 2021, pp. 4940–4950.
37. Dennis, M.; Jaques, N.; Vinitsky, E.; Bayen, A.; Russell, S.; Critch, A.; Levine, S. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems* **2020**, *33*, 13049–13061.
38. Parker-Holder, J.; Jiang, M.; Dennis, M.; Samvelyan, M.; Foerster, J.; Grefenstette, E.; Rocktäschel, T. Evolving curricula with regret-based environment design. In Proceedings of the International Conference on Machine Learning. PMLR, 2022, pp. 17473–17498.
39. Mouret, J.B.; Clune, J. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909* **2015**.
40. Jiang, M.; Dennis, M.; Parker-Holder, J.; Foerster, J.; Grefenstette, E.; Rocktäschel, T. Replay-guided adversarial environment design. *Advances in Neural Information Processing Systems* **2021**, *34*, 1884–1897.
41. Team, O.E.L.; Stooke, A.; Mahajan, A.; Barros, C.; Deck, C.; Bauer, J.; Sygnowski, J.; Trebacz, M.; Jaderberg, M.; Mathieu, M.; et al. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808* **2021**.
42. Fan, L.; Wang, G.; Jiang, Y.; Mandlekar, A.; Yang, Y.; Zhu, H.; Tang, A.; Huang, D.A.; Zhu, Y.; Anandkumar, A. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems* **2022**, *35*, 18343–18362.
43. Simonyan, K.; Vedaldi, A.; Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* **2013**.
44. Atrey, A.; Clary, K.; Jensen, D. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. *arXiv preprint arXiv:1912.05743* **2019**.
45. Narvekar, S.; Peng, B.; Leonetti, M.; Sinapov, J.; Taylor, M.E.; Stone, P. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research* **2020**, *21*, 1–50.

 doi:10.20944/preprints202602.1147.v1



46. Li, C.; Chen, Y.H.; Zhao, H.; Sun, H. Stackelberg game theory-based optimization of high-order robust control for fuzzy dynamical systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **2020**, *52*, 1254–1265.
47. Yang, C.; Wang, X.; Lu, Y.; Liu, H.; Le, Q.V.; Zhou, D.; Chen, X. Large language models as optimizers. In Proceedings of the The Twelfth International Conference on Learning Representations, 2023.
48. Littman, M.L. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*; Elsevier, 1994; pp. 157–163.
49. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems* **2017**, *30*.