STAR-MARL: LLM-based Sub-task Curricula Design

Zihao Li[*1] **, Dan Qiao**[*2]**, Muhammad Arrasy Rahman**[3]**,**
Yali Du[1]**, Stefanos Leonardos**[1]**, Stefano V. Albrecht**[4]

[1]Department of Informatics, King's College London, London, United Kingdom
[2]School of Data Science, The Chinese University of Hong Kong, Shenzhen, China
[3]Department of Computer Science, University of Texas at Austin, Texas, USA
[4]DeepFlow, London, United Kingdom
{zihao.1.li, yali.du, stefanos.leonardos}@kcl.ac.uk, danqiao@link.cuhk.edu.cn,
arrasy@cs.utexas.edu, stefano.albrecht@deepflow.com

Abstract

Sub-task curriculum learning has shown promise in cooperative multi-agent reinforcement learning (MARL), especially under sparse rewards. However, existing approaches often rely on expert-designed templates or end-to-end learning, limiting generalizability and efficiency. To address these limitations, We propose **STAR-MARL** (Sub-task Tree with Assisted Rewards), a fully automated framework that integrates large language models (LLMs) with the training dynamics of MARL agents. STAR-MARL uses Chain-of-Thought prompting and few-shot learning to generate a hierarchical, interpretable sub-task tree, with each node containing executable training scenarios and curriculum reward functions. However, a key challenge in MARL curriculum design lies in evaluating qualities of substasks, as online MARL training rollouts are computationally expensive and unstable. To address this, we introduce a retrieval-augmented generation (RAG)-based sub-curriculum evaluator that leverages MARL training trajectories to estimate potential policy improvement of reward functions without further environment interaction. Built atop a memory of historical sub-task trajectories, the evaluator enables offline curriculum evaluation and rapid curriculum refinement, making curriculum learning more sample-efficient and scalable. We apply the STAR-MARL framework to the Cooking Zoo and Google Research Football environments, generating interpretable curricula tasks of varying complexity. Our research paves the way for constructing interpretable, low-cost, and generalizable LLM-driven curricula for MARL.

Introduction

In cooperative multi-agent reinforcement learning (MARL) (Albrecht, Christianos, and Schäfer 2024), agents must explore the environment while learning to coordinate and collaborate. However, this becomes particularly difficult in complex or sparse-reward settings, where the exponential growth of the joint action space (Wong et al. 2023), the credit assignment problem (Chang, Ho, and Kaelbling 2003), and instabilities from mutual adaptation (Hernandez-Leal et al. 2017; Papoudakis et al. 2019) can significantly slow down learning or lead to suboptimal policies.

Curriculum learning (CL), particularly sub-task curricula (Ao et al. 2021; Fosong et al. 2023), is a common strategy to tackle these issues. By breaking down complex tasks into simpler multi-agent sub-tasks, CL enables agents to master individual components or skills and gradually build coordination, leading to better overall performance (Narvekar et al. 2020; Zhang et al. 2022). However, most standard CL methods rely on manually designed sub-tasks (Becht et al. 1999; Lhaksmana, Murakami, and Ishida 2018; Grote et al. 2020; Fosong et al. 2023), which are often rigid, domain-specific, or prone to bias. To address this, recent work has explored automated curriculum generation, aiming to produce flexible, goal-oriented training sequences without requiring domain knowledge. While promising, these methods often rely on end-to-end learning (Tian et al. 2023; Yang et al. 2023; You et al. 2025; Shah et al. 2025), resulting in curricula that are often hard to interpret, redundant, or poorly aligned with the target task.

Building on this direction, the emergence of large language models (LLMs) offers a unique natural opportunity to automate sub-task decomposition in MARL. Their zero-shot reasoning and task decomposition capabilities, grounded in commonsense knowledge and large-scale language patterns (Li et al. 2021), make LLMs well-suited for generating structured, task-relevant, and interpretable subtasks. LLMs have demonstrated extremely high accuracy in causal discovery and counterfactual reasoning (Kıcıman et al. 2023), with prompt engineering techniques such as chain-of-thought (CoT) (Wei et al. 2022) further boosting their reasoning capabilities. Unlike opaque end-to-end methods, LLMs can mimic human curriculum design by producing interpretable, goal-driven decompositions, addressing the rigidity and bias of manual approaches (Du et al. 2023).

Despite this promise, the integration of LLMs into multi-agent CL pipelines remains far from straightforward. It introduces significant new challenges such as the substantial prompt engineering effort required to support the decomposition, implementation, and optimization of sub-tasks within a unified framework; the difficulty of crafting prompts that

[*]These authors contributed equally.

enable a gradual increase in task complexity while effectively fostering agent cooperation; the challenge of aligning natural language outputs with the concrete skills of agents; the non-trivial problem of role assignment among agents; the absence of quality guarantees in reward functions generated by LLMs for sub-tasks, which necessitates considerable computational resources for their optimization; and the hallucination problem (Huang et al. 2025), wherein LLMs may ignore the given prompts and produce entirely unreasonable or incoherent reward functions. As a result, the few existing efforts to explore this direction either focus on single-agent tasks with limited exploration in multi-agent scenarios (Erak et al. 2024; Chi et al.) or are constrained to single-layer decompositions and require extensive manual inputs (Li et al. 2023).

Our Contributions: In this work, we propose **S**ub-task **T**ree with **A**ssisted **R**ewards for MARL (STAR-MARL), a novel framework that addresses the above limitations by introducing an automatic hierarchical sub-task decomposition mechanism guided by LLMs. The core insight behind STAR-MARL is that complex cooperative behaviors can be learned by recursively decomposing high-level multi-agent tasks into structured, skill-driven sub-tasks organized in a tree. Each node corresponds to a task that is more manageable in terms of agent interaction and reward design. This structure is based on the intuition that MARL tasks often contain compositional cooperation patterns—e.g., passing, positioning, and defending in football; and ingredient preparation and dish plating in cooking games—that can be explicitly captured and reused.

The key technical innovations of STAR-MARL lie in its recursive sub-task decomposition and reward improvement framework. STAR-MARL introduces LLM-driven recursive sub-task decomposition using Chain-of-Thought (CoT) prompting and few-shot examples to to guide the LLM in breaking down complex cooperative tasks into a hierarchical structure of semantically meaningful, interdependent sub-tasks. The LLM automatically assesses sub-task difficulty and refines the decomposition to avoid redundancy and maintain task diversity. Reward improvement is achieved via Retrieval-Augmented Generation (RAG), which prompts the LLMs with selected evaluations and improvement suggestions from an automatically generated reusable "component-evaluation" knowledge base, allowing the LLM to perform evaluation and refinement without costly reward model training.

These components are not merely stitched together but are mutually reinforcing: the tree structure guides the learning process, the LLM ensures interpretability and task relevance, and the RAG-based reward evaluation ensures alignment with agent behavior without requiring dense reward functions or hand-crafted shaping. This approach addresses the challenges of task complexity and coordination in MARL by making tasks more manageable while improving agent interaction and reward design. Our main contributions are:

1. We introduce STAR-MARL, a fully automated framework for constructing hierarchical sub-task curricula using LLMs. In experiments on Cooking Zoo and Google Research Football, STAR-MARL generates diverse and highly interpretable sub-task reward generation, eliminating the need for training task-specific rsk curricula.
2. We develop a reusable "component-evaluation" dataset and integrate RAG-based methods for sub-taeward models. This approach improves sample efficiency and generalization to new tasks.
3. We apply STAR-MARL-generated curricula within the MEDoE training framework.

Unlike prior work on LLM-based curriculum generation (Li et al. 2023; Erak et al. 2024; Chi et al.), STAR-MARL introduces a structured, recursive decomposition aligned with the hierarchical nature of cooperation in MARL tasks, while avoiding manual intervention. In contrast to previous LLM-based reward design methods (Ma et al. 2023; Li et al. 2024a; Sun et al. 2024; Baek et al. 2025), STAR-MARL eliminates the need for extensive training iterations typically required when adapting to new tasks by leveraging reusable knowledge through RAG. Overall, STAR-MARL offers a principled, interpretable, and generalizable solution for structured curriculum generation in cooperative MARL.

Related Work

Curriculum Learning and Automated Curriculum Generation in MARL: Curriculum learning (CL) (Bengio et al. 2009) simulates human learning by progressing from simple to complex tasks, and has been widely applied in supervised and reinforcement learning. In MARL, CL typically focuses on agent count, environment complexity, and reward design. For instance, the AI Economist (Zheng et al. 2022) structures training with varying agent types and environments, while Zhao et al. (2023) employ distinct reward functions for different agents. While automated curriculum generation has been explored to reduce manual design, most existing methods are either limited to single-agent RL (Narvekar, Sinapov, and Stone 2017; Wang et al. 2019, 2020; Forestier et al. 2022; Parker-Holder et al. 2022), or face challenges in MARL due to task assignment and inter-agent complexity. These methods typically rely on self-play (Sukhbaatar et al. 2017; Baker et al. 2019) or end-to-end learning (Long et al. 2020; Zhang et al. 2022; Tian et al. 2023; Yang et al. 2023; You et al. 2025; Shah et al. 2025; Hill 2025).

LLMs for MARL: The exceptional reasoning and zero- or few-shot generation abilities of LLMs have led to their application in single- and multi-agent RL for tasks such as generating trajectories (Hu and Sadigh 2023; Chen et al. 2024) and reward functions (Ma et al. 2023; Li et al. 2024a; Sun et al. 2024; Baek et al. 2025). In the context of designing sub-task curricula, LLMs have been employed by Erak et al. (2024) and Chi et al. to generate sub-tasks in single-agent RL. For MARL tasks, Li et al. (2023) proposed the SAMA framework. However, unlike our current approach, SAMA decomposes tasks into independent sub-tasks for individual agents, leading to shallow hierarchies and necessitating substantial human input for language-based task manuals. Another study, introduced the L2M2 framework (Geng et al.

2025), which leverages LLMs to generate zero-shot sub-task curricula for MARL. While it provides additional reward signals for each sub-task, it differs from our approach in that it does not produce customized reward functions or scenario configurations for individual sub-tasks. For scenario-based sub-tasks, existing work has proposed the cMALC-D framework (Satheesh, Powell, and Wei 2025), which leverages LLMs to dynamically generate training curricula, thereby improving agents' generalization to unseen environmental contexts.This framework targets at reflecting incremental difficulty or diversity. In contrast, our approach designs scenario settings tailored to specific sub-task descriptions, allowing for more precise adaptation to task requirements and supporting effective agent learning.

Problem Formulation

Sub-task Curricula

In this work, the sub-task is described in terms of sub-task curricula (STCs) (Fosong et al. 2023), which encapsulate the structured use of sub-task decomposition throughout the training process. A sub-task curriculum C is a tuple, $\langle T, \{C_{\text{sub}}\}, \mathcal{A}\rangle$. The task T is a Dec-POMDP (Oliehoek, Amato et al. 2016),

$$M = \langle I, S, \{A_i\}_{i\in I}, P, \mu, \{\Omega_i\}_{i\in I}, O, R, \gamma\rangle, \qquad (1)$$

where I is the set of the agents involved in this sub-task; S is the set of states; A_i is the set of actions for agent i, and $A = \times_i A_i$ is the set of joint actions, where each joint action $a \in A$ is a tuple $a = (a_i)_{i\in I}$, with $a_i \in A_i$ for each agent $i \in I$; $P(s_{t+1} \mid s_t, a_t)$ is the state transition probability density function; $\mu(s_0)$ is the initial state distribution; Ω_i is the observation space for agent i; $O(o_t \mid s_t, a_{t-1})$ is the observation probability density function; $R : S \times A \to R$ is the reward function shared across the team; γ is the discount factor, which maintains a finite sum in the infinite-horizon case ($\gamma \in [0, 1)$).

The objective within a Dec-POMDP is to find a joint policy $\pi = (\pi_i)_{i\in I}$, where each π_i maps the local action-observation history of agent i to a distribution over its actions A_i, that maximizes the expected cumulative discounted return:

$$G = E_\pi\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right], \qquad (2)$$

where the expectation is taken over trajectories induced by the joint policy π.

$\{C_{\text{sub}}\}$ is a set of sub-task curricula, where each curriculum is recursively defined as either: (i) a base case $C_{\text{sub}} = \emptyset$, which corresponds to training from scratch on task T; or (ii) a higher-level curriculum composed of sub-curricula, $C_{\text{sub}} = \{C_1, C_2, \ldots, C_k\}$, where each $C_j \in C_{\text{sub}}$ recursively defines its own structure. This leads to a hierarchical *sub-task curricula tree*, where the leaves represent base cases, internal nodes represent sub-curricula of progressively increasing complexity, and the root node represents the curriculum for the final target task T_t.

Finally, $\mathcal{A} : \{C_{\text{sub}}\} \to 2^I$ is a function which maps from the set of sub-tasks $\{C_{\text{sub}}\}$ to a subset of agents I for task T. For brevity, we also use $\mathcal{A}$ to denote the set of agents participating in the current task in the curriculum.

Figure 1: 5-vs-5 football task decomposition into a two-level sub-task curricula tree.

For example, for a target task of the 5-vs-5 football game illustrated in Figure 1, a sub-task curricula tree with a depth of 2 would be:

$$\begin{aligned}
C_{5v5} &= \langle T_{5v5}, \{C_{\text{defender}}, C_{\text{attacker}}\}, \mathcal{A}_{5v5}\rangle \\
C_{\text{defender}} &= \langle T_{\text{defender}}, \emptyset, \mathcal{A}_{\text{defender}}\rangle \\
C_{\text{attacker}} &= \langle T_{\text{attacker}}, \{C_{\text{passer}}, C_{\text{shooter}}\}, \mathcal{A}_{\text{attacker}}\rangle \\
C_{\text{passer}} &= \langle T_{\text{passer}}, \emptyset, \mathcal{A}_{\text{passer}}\rangle \\
C_{\text{shooter}} &= \langle T_{\text{shooter}}, \emptyset, \mathcal{A}_{\text{shooter}}\rangle
\end{aligned}$$

where each $\mathcal{A}_* \subseteq I$ denotes the subset of agents assigned to the corresponding sub-task T_*.

Objective:

Our objective is to improve cooperative MARL efficiency and stability by automatically generating hierarchical sub-task curricula. Specifically, we aim to minimize the total number of training timesteps required for agents to reach a desired level of performance in the target task.

Let $N(C)$ represent the number of training timesteps required for the current task within the curriculum C, and let $N_{\text{tot}}(C)$ represent the total number of training timesteps across the entire subtree of sub-task curricula C_{sub} rooted at the current task C. Specifically,

$$N_{\text{tot}}(C) = N(C) + \sum_{C_j \in C_{\text{sub}}} N_{\text{tot}}(C_j) \qquad (3)$$

Given a target task T_t and a target performance level L_t, our overall objective is to find an optimal sub-task curricula $C_{\text{t}}^* = \langle T_t, \ldots\rangle$, which minimizes $N(C_{\text{t}}^*)$, while ensuring that the agents achieve the expected returns $L \geq L_t$.

This optimization problem includes multiple factors: the MARL algorithm chosen at each stage of training, the stopping conditions at each stage, the design of sub-tasks at each stage, and the agents involved at each stage. In this study, we focus on the design of sub-tasks and the role assignment of agents, while the learning algorithm is based on the MEDoE (Fosong et al. 2023) method.

Sub-task Tree with Assisted Rewards (STAR) for MARL

We propose the STAR-MARL framework, which leverages the capabilities of LLMs to automatically generate hierarchical sub-task curricula for MARL tasks in complex environments.

As illustrated in Figure 2, the overall architecture of STAR-MARL consists of two core modules: a Sub-task Curricula Generator and a RAG-Based Reward Function Evaluator. Given any target task along with the game rules as input, the Sub-task Curricula Generator produces a skill-driven, tree-structured sub-task curricula. Each sub-task curriculum in the tree includes a customized training scenario and a corresponding reward function, enabling agents to progressively acquire key skills at different learning stages and transition to more challenging objectives.

The RAG-Based Reward Function Evaluator leverages the component-evaluation database, constructed during the automated preprocessing stage, to assess and iteratively refine the reward functions generated by the LLM. This process ensures the effectiveness and alignment of each reward function with its associated sub-task objective.

The final sub-task curricula tree can be integrated with any MARL algorithm for curriculum-based sub-task training. In our experiments, we adopt the MEDoE framework (Fosong et al. 2023) as a demonstration platform for evaluation. However, our method is algorithm-agnostic and can be applied universally across different MARL settings.

The technical implementation details of each module will be elaborated in the following sections.

Sub-task Curricula Generator

The motivation behind representing task decomposition as a multi-layer tree structure is that, in contrast to simple single-layer sub-task decomposition, hierarchical decomposition—by breaking down complex target tasks into smaller, skill-driven sub-tasks—facilitates more meaningful exploration of the policy at lower levels, while also promoting coordination and cooperation at higher levels. This structure accelerates the learning process and enhances performance in complex environments by encouraging both individual skill acquisition and collaborative behavior among agents (Iqbal, Costales, and Sha 2022; Li et al. 2024b).

As outlined in Algorithm 1, the process of the *Sub-task Curricula Generator* begins with the initialization of the environment game rules (E), which are described in natural language and specify the scoring criteria, initial positions of agents, and the action space available to each agent; the target task (T_t) described in natural language; the large language models ($\text{LLM}_d, \text{LLM}_s, \text{LLM}_r$), which are responsible for the generation of task decomposition, scenario setting, and reward function, respectively; and the set of agents ($\mathcal{A}_t$) involved in the target task.

Algorithm 1: Sub-task Curriculum Generation

Input: Game rules E, target task T_t, agents $\mathcal{A}_t$, $\text{LLM}_{\{d,s,r\}}$
Output: Sub-task curricula tree $C_t = \langle T_t, \{C_{\text{sub}}\}, \mathcal{A}_t \rangle$

// Stage 1: Generate sub-task decomposition tree
$\text{Prompt}_d \leftarrow \texttt{ComposeDecompositionPrompt}(E, T_t, \mathcal{A}_t)$
$G_{\text{sub}} \leftarrow \text{LLM}_d(\text{Prompt}_d)$
for each node $N_j = \langle G_j, \mathcal{A}_j \rangle$ in G_{sub} **do**
  // Stage 2: Generate scenario settings for sub-task
  $\text{Prompt}_{s,j} \leftarrow \texttt{ComposePrompt}(N_j, G_{\text{sub}})$
  $\mathcal{S}_j \leftarrow \text{LLM}_s(\text{Prompt}_{s,j})$
  // Stage 3: Generate and refine rewards for sub-task
  $\text{Prompt}_{r,j} \leftarrow \texttt{ComposePrompt}(N_j, G_{\text{sub}})$
  $R_j \leftarrow \text{LLM}_r(\text{Prompt}_{r,j})$
  $R'_j \leftarrow \texttt{RewardEvaluator}(R_j, G_j, \mathcal{A}_j)$
  $C_j \leftarrow (\mathcal{S}_j, R'_j, \mathcal{A}_j)$
  Add C_j to C_{sub}
end for
return $C_t = \langle T_t, \{C_{\text{sub}}\}, \mathcal{A}_t \rangle$ =0

The first step involves using LLM_d to generate a tree (G_{sub}) of the textual-tailored sub-task goals and involved agents based on the prompt (Prompt_d) composed of the textual description of the E, T_t and $\mathcal{A}_t$. Subsequently, the nodes of G_{sub} are traversed from bottom to top. For each node N_j, to construct $\text{Prompt}_{s,j}$ and $\text{Prompt}_{r,j}$, which are then provided as input to LLM_s and LLM_r, respectively, to generate the corresponding scenario setting$\mathcal{S}_j$ and reward function R_j code. After R_j is refined by the RAG-Based Reward Function Evaluator, the improved version $R_j^{'}$ is obtained. Each sub-task curriculum C_j is then tailored based on $\mathcal{S}_j$, $R_j^{'}$ and involved agents $\mathcal{A}_j$. Upon completion of the traversal, the full sub-task tree C_t is derived, which is subsequently used for MARL training.

Textual Sub-Task Tree Generation Figure 3a illustrates an example workflow of generating a textual sub-task tree using LLM_d, which consists of two stages: an initial decomposition of the target task to produce first-level sub-tasks, and a subsequent further decomposition stage involving refinement to derive deeper sub-task structures. Figure 3b presents example prompts used in these two stages along with the corresponding responses from LLM_d.

Specifically, in Stage 1, we construct Prompt_d using natural language descriptions of E, T_t, and $\mathcal{A}_t$. To improve inference quality and coherence while mitigating hallucinations (Huang et al. 2025), we adopt Chain-of-Thought (CoT) prompting (Wei et al. 2022) to guide LLM_d through step-by-step reasoning. The model is also instructed to output its reasoning process for better interpretability. Additionally, we incorporate few-shot learning with manually designed decomposition examples. The output consists of first-level textual sub-tasks decomposed from the target task, each annotated with a group ID, a sub-goal, and associated agents.

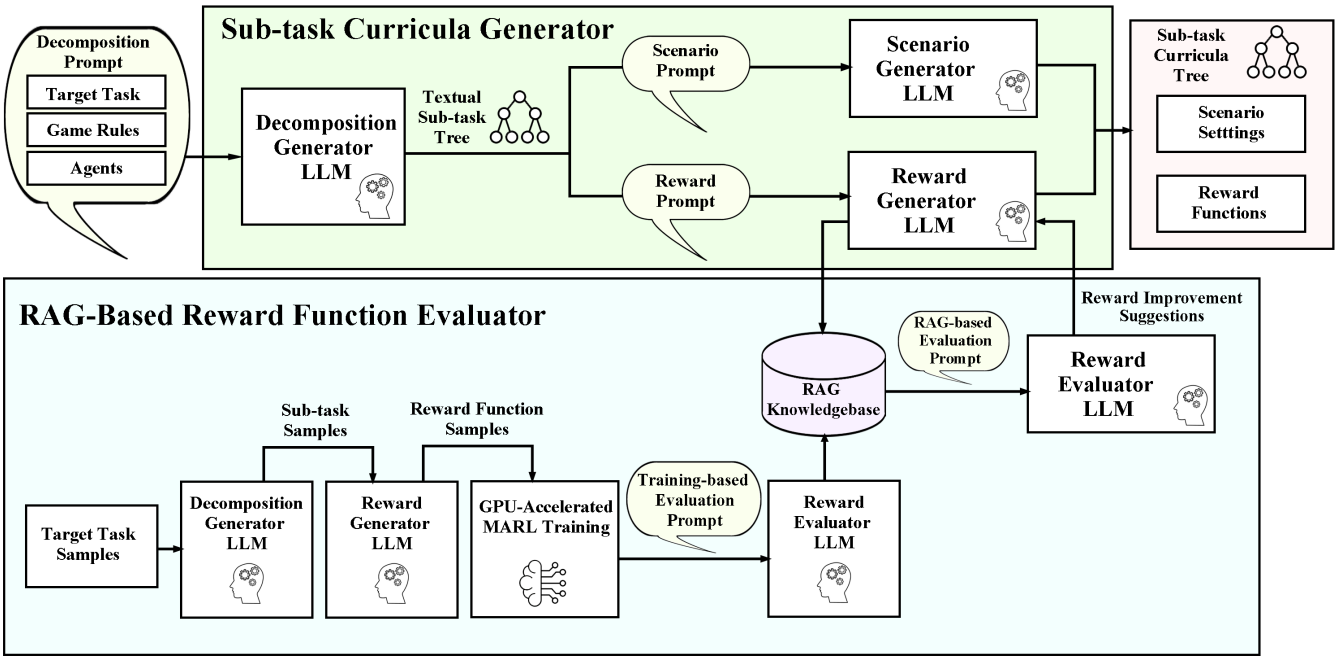


Figure 2: The STAR-MARL Framework: The target task is decomposed into a hierarchical sub-task curricula tree by the Sub-task Curricula Generator. Scenario and reward function code are generated, and the reward functions are iteratively refined via Retrieval-Augmented Generation (RAG)-Based Reward Function Evaluator—without additional training.

In Stage 2 recursively decompose sub-tasks layer by layer, starting from the first level. For each sub-task, we extract the branch from the root target task to the current node and use it as a prompt for LLM_d to assess task complexity and determine whether further decomposition is needed. If so, the model generates next-level sub-tasks in the same format as Stage 1. A central challenge is avoiding redundant learning. As shown in Figure 3a, a second-layer sub-task like "learning passing and dribbling for middle attack" may be decomposed into third-layer tasks such as "learning passing" and "learning dribbling", resulting in repetition. To mitigate this, we prompt LLM_d to revise the parent task during decomposition—for example, modifying it to "learning cooperative behaviors for middle attack" before further decomposition. This process continues until no sub-tasks require further breakdown, yielding a complete textual sub-task tree G_{sub} grounded in the target task.

Scenario Generation Extensive research on code generation with LLMs (Ma et al. 2023; Li et al. 2024a; Sun et al. 2024; Baek et al. 2025) demonstrates their strong ability to generate executable code. Thus, we apply few-shot prompting to instruct LLM_s to generate executable code for the scenario setting and reward function of each sub-task. Figure 3c shows the template for Prompt_s, which includes the sub-task goal, involved agents, sub-task tree structure, environment code relevant to the scenario, code output tips, and manually crafted examples.

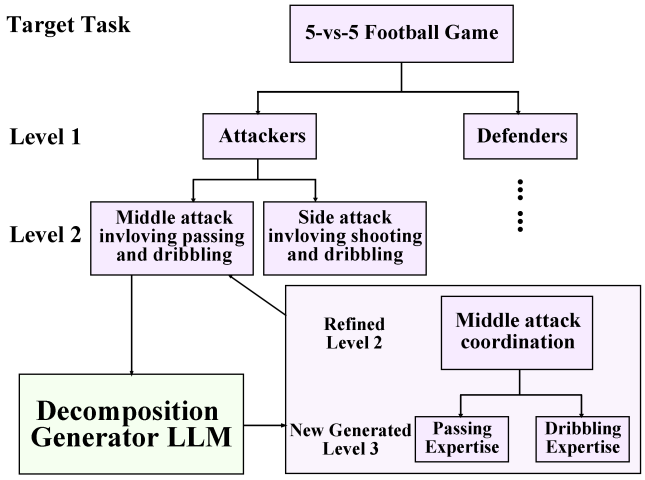
Reward Function Generation The Eureka (Ma et al. 2023) framework demonstrates strong capabilities in handling complex robotic tasks. Our reward function generation framework builds on Eureka, enhanced with CoT prompting and few-shot techniques. As shown in Figure 3d, the Prompt_r consists of the task description, involved agents, sub-task tree structure, code output tips, observation explanation, and manually crafted examples. The reward function generated by LLM_r uses the observation as input, composed of multiple reward components for easier refinement.

During testing, embedding the complete environment code in the prompt, as done in Eureka, often led to LLM hallucinations due to an excessive number of tokens, resulting in incorrect or misinterpreted reward functions. To mitigate this, we removed the environment code from the prompt and provided a textual explanation of the observation, with annotated examples of the agent's observation space. This approach effectively reduced hallucinations.

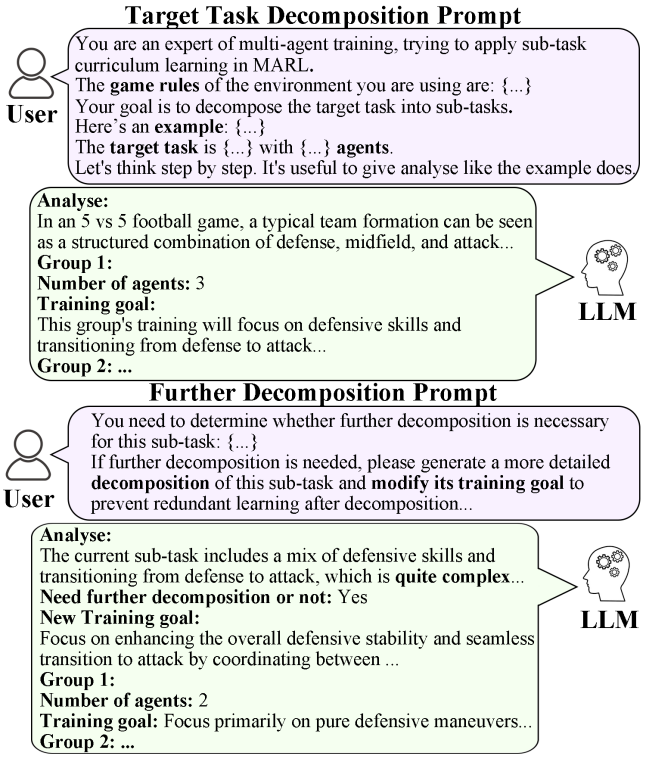
Retrieval- Augmented Generation (RAG)-Based Reward Function Evaluator

The Eureka framework employs iterative training and evolutionary optimization for reward function optimisation, which is effective in simple single-agent RL tasks but incurs substantial computational cost in complex multi-agent settings such as Google Research Football (GRF) (Kurach et al. 2020). To mitigate this, we propose a RAG-based Reward Function Evaluator that uses a pre-trained knowledge base to enable efficient evaluation without additional training.

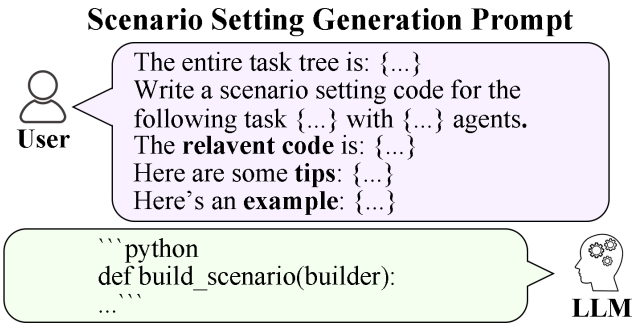
This approach is motivated by our observation that LLM-generated reward components exhibit structural consistency across similar tasks, and that sub-task decompositions ex-



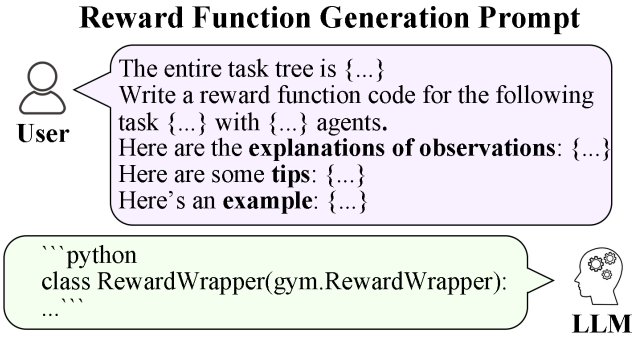
(a) 3-level sub-task decomposition example for the target task of a 5-vs-5 football game



(b) Example decomposition prompt followed by LLM-generated textual sub-task decomposition



(c) Example scenario prompt followed by LLM-generated scenario setting code



(d) Example reward prompt followed by LLM-generated reward function code

Figure 3: The Sub-task Curricula Generator comprises three key processes: (1) constructing a textual representation of the sub-task tree; and for each sub-task, generating the corresponding (2) scenario setting code and (3) reward function code.

hibit diverse yet bounded patterns. Leveraging this, we construct a "component-evaluation" knowledge base and apply RAG (Lewis et al. 2020) to assess and refine reward functions, aiming for task-effective rather than globally optimal solutions.

The "component-evaluation" knowledge base was constructed as follows. For complex target tasks, a decomposition generator LLM_d was employed to generate sub-task goals, covering various agent configurations and coordination complexities. For each sub-task, distinct reward function samples were generated using the reward function generator LLM_r, and then trained with a standard MARL algorithm under predefined conditions. The training trajectories of each reward function were tracked. Since outcome-based metrics were not applicable in the early stages of training in some complex MARL environments, such as goal-scoring in GRF, we monitored the values of reward function components at multiple checkpoints to assess their effectiveness. Additionally, to prevent reward hacking (Skalse et al. 2022), we tracked the frequency of key actions performed by agents.

For evaluation, the sub-task description, agent configura-

Algorithm 2: RAG-Based Reward Function Evaluation

Input: Reward function R_j; Goal G_j; Agents $\mathcal{A}_j$; $\text{LLM}_{\{\text{emb},e\}}$; RAG knowledge base $\mathcal{K}$
Output: Refined reward function R'_j

$\text{EMB}_j \leftarrow \text{LLM}_{\text{emb}}(R_j, G_j, \mathcal{A}_j)$
for N iterations **do**
  $\mathcal{R}_{10} \leftarrow \underset{r_i \in \mathcal{K}}{\text{Top-10}} \left\{ \frac{\text{cosine\_similarity}(\text{EMB}_j, r_i)}{\|\text{EMB}_j\| \|r_i\|} \right\}$
  $\text{Prompt}_{e,j} \leftarrow \texttt{ComposePrompt}(R_j, G_j, \mathcal{A}_j, \mathcal{R}_{10})$
  $R'_j \leftarrow \text{LLM}_e(\text{Prompt}_{e,j})$
  $\text{EMB}_j \leftarrow \text{LLM}_{\text{emb}}(R'_j, G_j, \mathcal{A}_j)$
end for
return R'_j =0

tion, reward function code, tracked metrics, and an evaluation guide were provided as input to the evaluator LLM_e. CoT prompting was used to enhance reasoning. The output for each reward function component consisted of: (1) a binary judgment indicating whether the component facilitated agent learning, and (2) suggestions for improving the component. The final "component-evaluation" knowledge base included the specific components, sub-task descriptions, agent configurations, full reward function code, and evaluation results from LLM_e.

Algorithm 2 outlines the process of optimizing reward functions using the knowledge base. Upon generating the initial reward function code R_j for a new sub-task, we embed its textual goal G_j, agent count $\mathcal{A}_j$ together with R_j using Monarch Mixer-BERT (Fu et al. 2023) as LLM_{emb}. Subsequently, we retrieve the top-10 most semantically relevant reward function components $\mathcal{R}_{10}$ from the knowledge base $\mathcal{K}$ via cosine similarity. These retrieved components, along with their corresponding evaluation outcomes and refinement suggestions, are incorporated into the prompt $\text{Prompt}_{e,j}$ provided to the evaluator LLM_e. The evaluator then evaluates R_j and provides a refined reward function. It iteratively refines the reward function for N predefined iterations. As the value of N increases, the final refined reward function progressively converges toward the representations encoded within the knowledge base.

Experiments

In this section, we evaluate STAR-MARL on several standard multi-agent reinforcement learning suites to investigate the question: Can LLMs design better curricula for MARL learning than human experts or end-to-end learning?

We adopt the Modulating Exploration and Training via Domain of Expertise (MEDoE) framework proposed by Fosong et al. (2023) to train MARL tasks. This framework is specifically designed for sub-task curricula learning in MARL with given sub-tasks and ensures that agents retain the specialized skills acquired in sub-tasks when transitioning to the main task.

Settings

Environment We use two environments:

CookingZoo (Fosong et al. 2023), a flexible cooking environment that provides a variety of cooking tools and ingredients.

Google Research Football (GRF) (Kurach et al. 2020), a complex MARL environment with sparse rewards, where agents are trained to play football in a physics-based 3D simulator.

For the target task's scenario selection, we use:

`coop_test`,a CookingZoo setting in which two agents cooperate to complete two recipes: "Tomato Lettuce Salad" and "Carrot Banana".

`academy_3_vs_1_with_keeper`, a GRF setting in which three of our players try to score from the edge of the box: one on each side, and the other at the center. Initially, the player at the center has the ball and is facing the defender. Each side also has a built-in AI-controlled goalkeeper who does not participate in the training.

Baselines For the experiments in CookingZoo, we chose the standard MAPPO (Yu et al. 2022) algorithm as the baseline. For the experiments in GRF, we select LDSA (Yang et al. 2022) and the standard IPPO (De Witt et al. 2020) as baselines to respectively compare STAR-MARL with sub-task curricula generated by end-to-end learning methods and traditional MARL algorithms trained from scratch.

LLM resources We use GPT-4 for the decomposition generator and the reward evaluator, Claude-4 for the scenario generator and the reward generator, and Monarch Mixer-BERT (Fu et al. 2023) for embedding generation.

RAG database For the CookingZoo experiments, the reward functions are relatively simple, so we did not utilize the RAG database for reward function enhancement. For the GRF experiments, We use 60 sample tasks, each paired with 30 reward functions generated by GPT-4. Each reward function is trained using the standard IPPO algorithm for 500,000 steps within the 5-vs-5 standard scenario. The training outcomes are then evaluated by GPT-4, resulting in a "component-evaluation" knowledge base comprising 3,062 entries. This knowledge base is utilized in STAR-MARL for RAG-based reward function evaluation. This knowledge base covers multiple tasks across various football scenarios and can be shared and reused across all sub-tasks.

Training Settings In the CookingZoo experiments, we trained each sub-task separately for 5 million steps, followed by 20 million steps of training on the target task. In the GRF experiments, to prevent the reward function from overly converging to the data in the knowledge base, we set N, the number of refinement iterations for the sub-task reward function to 1. We trained each sub-task separately for 500,000 steps, followed by 10 million steps of training on the target task. In the presented experimental results, STAR-MARL and IPPO each use a single random seed, while LDSA uses 5 random seeds.

Results on Cooking Zoo

As shown in Figure 4, STAR-MARL exhibits faster performance improvement during training on the main task and ultimately achieves higher mean episodic returns compared to the standard MAPPO algorithm.

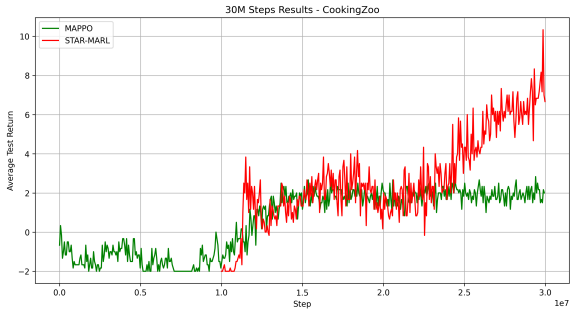


Figure 4: Results on the CookingZoo `coop_test` scenario using a single seed.

Results on Google Research Football

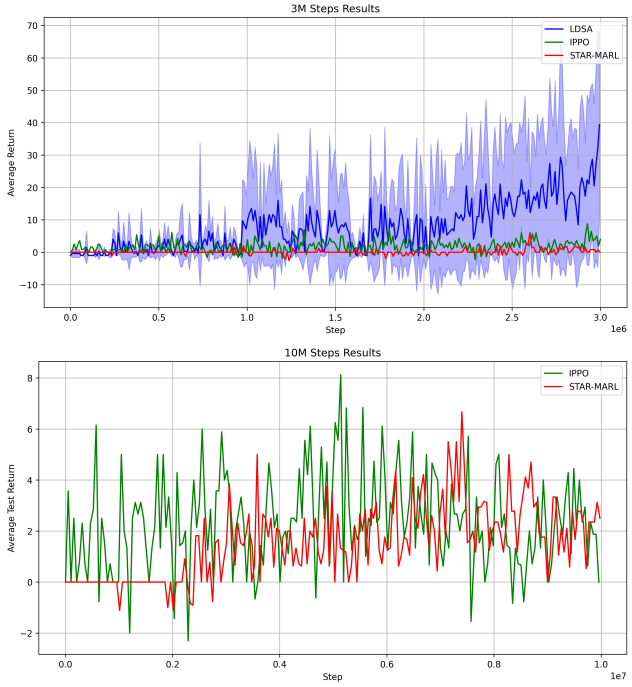


Figure 5: Results on GRF `academy_3_vs_1_with_keeper` scenario. LDSA includes 5 seeds, while STAR-MARL and IPPO each use a single seed.

As shown in Figure 5, compared to the end-to-end learning-based LDSA method and the train-from-scratch approach using the basic IPPO algorithm, STAR-MARL does not demonstrate an improvement in training efficiency; Throughout the entire training process, the training results of STAR-MARL also do not converge to a policy better than that of the basic IPPO algorithm.

We analyze that there are three main reasons for the unsatisfactory results: first, the complexity of the tasks in Google Research Football makes training challenging; second, the quality of the RAG knowledge base has not been manually verified, leading to instability in the optimization of the reward function; third, the design of different scenarios for the subtasks causes the Domain of Experts (DoE) classifier used in the MEDoE method to make inaccurate judgments. Specifically, the DoE classifier relies on the intuition that "agents who have mastered skills in sub-tasks will encounter similar states in the target task," and thus their exploration should be reduced. However, this intuition assumes that the same scenario is used in both the sub-tasks and the target task. Otherwise, similar states may not align in meaning across different scenarios. For example, in one sub-task, enemy lazy players are placed at fixed positions, and agents focusing on movement skills deliberately avoid these fixed positions; however, in the target task, these positions do not have enemy players.

Conclusion and Future Work

In this work, we propose STAR-MARL, an automatic hierarchical sub-task decomposition with reward refinement framework for MARL driven by LLMs. STAR-MARL recursively breaks down the complex cooperative task into a structured sub-task tree, mitigating the limitations of manual curriculum design. The reward functions generated by LLMs are dynamically refined by leveraging a pre-trained RAG knowledge base, eliminating the need for additional training on newly generated reward functions for new tasks, thereby improving sample efficiency. Empirical results show that although the current STAR-MARL-generated curricula do not perform well in improving cooperative learning efficiency and performance in complex multi-agent environments, the reasons for this are analyzable. Overall, STAR-MARL offers a principled, interpretable, and scalable approach to curriculum generation in MARL, highlighting the potential of integrating LLMs with sub-task curriculum design for complex cooperative tasks.

One future work direction involves optimizing the sub-task decomposition process. currently, the determination of decomposition depth is solely governed by the LLM. An alternative approach involves leveraging information capacity (Furuta et al. 2021) as a quantitative metric to evaluate the complexity of sub-tasks, thereby providing an objective criterion to guide the decision on whether further hierarchical decomposition is warranted. Additionally, we aim to incorporate agent-specific role assignment to distinguish roles within the same task, complementing existing task-based role assignments.

Another potential avenue for future work is to utilize thought cloning (Hu and Clune 2024) in place of the DoE classifier in the MEDoE algorithm used in the experiments. This method eliminates the requirement to train a separate classifier for each high-level sub-task to determine whether the corresponding sub-task has acquired expert policies. Instead, during training, the agent generates an explicit explanation of its policy along with a continuous representation, which the large language model (LLM) can use to assess whether the sub-task has achieved its goal. In future experiments, we plan to introduce various environments—from complex ones like StarCraft 2 to simpler ones like Chainball (Fosong et al. 2023).

References

Albrecht, S. V.; Christianos, F.; and Schäfer, L. 2024. *Multi-agent reinforcement learning: Foundations and modern approaches*. MIT Press.

Ao, S.; Zhou, T.; Long, G.; Lu, Q.; Zhu, L.; and Jiang, J. 2021. CO-PILOT: Collaborative planning and reinforcement learning on sub-task curriculum. *Advances in Neural Information Processing Systems*, 34: 10444–10456.

Baek, I.-C.; Kim, S.-H.; Earle, S.; Jiang, Z.; Jin-Ha, N.; Togelius, J.; and Kim, K.-J. 2025. Pcgrllm: Large language model-driven reward design for procedural content generation reinforcement learning. *arXiv preprint arXiv:2502.10906*.

Baker, B.; Kanitscheider, I.; Markov, T.; Wu, Y.; Powell, G.; McGrew, B.; and Mordatch, I. 2019. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*.

Becht, M.; Gurzki, T.; Klarmann, J.; and Muscholl, M. 1999. ROPE: Role oriented programming environment for multi-agent systems. In *Proceedings Fourth IFCIS International Conference on Cooperative Information Systems. CoopIS 99 (Cat. No. PR00384)*, 325–333. IEEE.

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, 41–48.

Chang, Y.-H.; Ho, T.; and Kaelbling, L. 2003. All learning is local: Multi-agent learning in global reward games. *Advances in neural information processing systems*, 16.

Chen, X.-H.; Wang, Z.; Du, Y.; Jiang, S.; Fang, M.; Yu, Y.; and Wang, J. 2024. Policy learning from tutorial books via understanding, rehearsing and introspecting. *Advances in Neural Information Processing Systems*, 37: 18940–18987.

Chi, H.; Zhao, S.; Tsang, I.; Ong, Y.-S.; Chen, H.; Chang, Y.; and Yin, H. ???? LLMV-AgE: verifying LLM-guided planning for agentic exploration in open-world RL. In *International Conference on Learning Representations 2025 Workshop: VerifAI: AI Verification in the Wild*.

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

Du, Y.; Watkins, O.; Wang, Z.; Colas, C.; Darrell, T.; Abbeel, P.; Gupta, A.; and Andreas, J. 2023. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*, 8657–8677. PMLR.

Erak, O.; Alhussein, O.; Naser, S.; Alabbasi, N.; Mi, D.; and Muhaidat, S. 2024. Large Language Model-Driven Curriculum Design for Mobile Networks. *arXiv preprint arXiv:2405.18039*.

Forestier, S.; Portelas, R.; Mollard, Y.; and Oudeyer, P.-Y. 2022. Intrinsically motivated goal exploration processes with automatic curriculum learning. *Journal of Machine Learning Research*, 23(152): 1–41.

Fosong, E.; Rahman, A.; Carlucho, I.; and Albrecht, S. V. 2023. Learning complex teamwork tasks using a given sub-task decomposition. *arXiv:2302.04944*.

Fu, D.; Arora, S.; Grogan, J.; Johnson, I.; Eyuboglu, E. S.; Thomas, A.; Spector, B.; Poli, M.; Rudra, A.; and Ré, C. 2023. Monarch mixer: A simple sub-quadratic gemm-based architecture. *Advances in Neural Information Processing Systems*, 36: 77546–77603.

Furuta, H.; Matsushima, T.; Kozuno, T.; Matsuo, Y.; Levine, S.; Nachum, O.; and Gu, S. S. 2021. Policy information capacity: Information-theoretic measure for task complexity in deep reinforcement learning. In *International Conference on Machine Learning*, 3541–3552. PMLR.

Geng, M.; Pateria, S.; Subagdja, B.; Li, L.; Zhao, X.; and Tan, A.-H. 2025. L2m2: A hierarchical framework integrating large language model and multi-agent reinforcement learning. In *International Joint Conference on Artificial Intelligence*.

Grote, E.-M.; Pfeifer, S. A.; Röltgen, D.; Kühn, A.; and Dumitrescu, R. 2020. Towards defining role models in advanced systems engineering. In *2020 IEEE International Symposium on Systems Engineering (ISSE)*, 1–7. IEEE.

Hernandez-Leal, P.; Kaisers, M.; Baarslag, T.; and De Cote, E. M. 2017. A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*.

Hill, B. 2025. Co-Evolving Complexity: An Adversarial Framework for Automatic MARL Curricula. *arXiv preprint arXiv:2509.03771*.

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

Hu, S.; and Clune, J. 2024. Thought cloning: Learning to think while acting by imitating human thinking. *Advances in Neural Information Processing Systems*, 36.

Huang, L.; Yu, W.; Ma, W.; Zhong, W.; Feng, Z.; Wang, H.; Chen, Q.; Peng, W.; Feng, X.; Qin, B.; et al. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2): 1–55.

Iqbal, S.; Costales, R.; and Sha, F. 2022. Alma: Hierarchical learning for composite multi-agent tasks. *Advances in neural information processing systems*, 35: 7155–7166.

Kıcıman, E.; Ness, R.; Sharma, A.; and Tan, C. 2023. Causal reasoning and large language models: Opening a new frontier for causality. *arXiv preprint arXiv:2305.00050*.

Kurach, K.; Raichuk, A.; Stańczyk, P.; Zajac, M.; Bachem, O.; Espeholt, L.; Riquelme, C.; Vincent, D.; Michalski, M.; Bousquet, O.; et al. 2020. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 4501–4510.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474.

Lhaksmana, K. M.; Murakami, Y.; and Ishida, T. 2018. Role-based modeling for designing agent behavior in self-organizing multi-agent systems. *International Journal of*

Software Engineering and Knowledge Engineering, 28(01): 79–96.

Li, H.; Yang, X.; Wang, Z.; Zhu, X.; Zhou, J.; Qiao, Y.; Wang, X.; Li, H.; Lu, L.; and Dai, J. 2024a. 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*, 16426–16435.

Li, W.; Qiao, D.; Wang, B.; Wang, X.; Jin, B.; and Zha, H. 2023. Semantically aligned task decomposition in multi-agent reinforcement learning. *arXiv preprint arXiv:2305.10865*.

Li, X.; Li, Y.; Zhang, J.; Xu, X.; and Liu, D. 2024b. A hierarchical multi-agent allocation-action learning framework for multi-subtask games. *Complex & Intelligent Systems*, 10(2): 1985–1995.

Li, X. L.; Kuncoro, A.; Hoffmann, J.; d'Autume, C. d. M.; Blunsom, P.; and Nematzadeh, A. 2021. A systematic investigation of commonsense knowledge in large language models. *arXiv preprint arXiv:2111.00607*.

Long, Q.; Zhou, Z.; Gupta, A.; Fang, F.; Wu, Y.; and Wang, X. 2020. Evolutionary population curriculum for scaling multi-agent reinforcement learning. *arXiv preprint arXiv:2003.10423*.

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

Narvekar, S.; Peng, B.; Leonetti, M.; Sinapov, J.; Taylor, M. E.; and Stone, P. 2020. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181): 1–50.

Narvekar, S.; Sinapov, J.; and Stone, P. 2017. Autonomous task sequencing for customized curriculum design in reinforcement learning. In *International Joint Conference on Artificial Intelligence*, 2536–2542.

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

Papoudakis, G.; Christianos, F.; Rahman, A.; and Albrecht, S. V. 2019. Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1906.04737*.

Parker-Holder, J.; Jiang, M.; Dennis, M.; Samvelyan, M.; Foerster, J.; Grefenstette, E.; and Rocktäschel, T. 2022. Evolving curricula with regret-based environment design. In *International Conference on Machine Learning*, 17473–17498. PMLR.

Satheesh, A.; Powell, K.; and Wei, H. 2025. cMALC-D: Contextual Multi-Agent LLM-Guided Curriculum Learning with Diversity-Based Context Blending. *arXiv preprint arXiv:2508.20818*.

Shah, A.; Lauffer, N.; Chen, T.; Pitta, N.; and Seshia, S. A. 2025. Learning symbolic task decompositions for multi-agent teams. *arXiv preprint arXiv:2502.13376*.

Skalse, J.; Howe, N.; Krasheninnikov, D.; and Krueger, D. 2022. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35: 9460–9471.

Sukhbaatar, S.; Lin, Z.; Kostrikov, I.; Synnaeve, G.; Szlam, A.; and Fergus, R. 2017. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*.

Sun, S.; Liu, R.; Lyu, J.; Yang, J.-W.; Zhang, L.; and Li, X. 2024. A large language model-driven reward design framework via dynamic feedback for reinforcement learning. *arXiv preprint arXiv:2410.14660*.

Tian, Z.; Chen, R.; Hu, X.; Li, L.; Zhang, R.; Wu, F.; Peng, S.; Guo, J.; Du, Z.; Guo, Q.; et al. 2023. Decompose a task into generalizable subtasks in multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 36: 78514–78532.

Wang, R.; Lehman, J.; Clune, J.; and Stanley, K. O. 2019. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*.

Wang, R.; Lehman, J.; Rawal, A.; Zhi, J.; Li, Y.; Clune, J.; and Stanley, K. 2020. Enhanced poet: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In *International conference on machine learning*, 9940–9951. PMLR.

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

Wong, A.; Bäck, T.; Kononova, A. V.; and Plaat, A. 2023. Deep multiagent reinforcement learning: Challenges and directions. *Artificial Intelligence Review*, 56(6): 5023–5056.

Yang, M.; Yang, Y.; Lu, Z.; Zhou, W.; and Li, H. 2023. Hierarchical multi-agent skill discovery. *Advances in Neural Information Processing Systems*, 36: 61759–61776.

Yang, M.; Zhao, J.; Hu, X.; Zhou, W.; Zhu, J.; and Li, H. 2022. Ldsa: Learning dynamic subtask assignment in cooperative multi-agent reinforcement learning. *Advances in neural information processing systems*, 35: 1698–1710.

You, C.; Wu, Y.; Cai, J.; Luo, Q.; and Zhou, Y. 2025. Dynamic subtask representation and assignment in cooperative multi-agent tasks. *Neurocomputing*, 129535.

Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; and Wu, Y. 2022. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems*, 35: 24611–24624.

Zhang, T.; Liu, Z.; Pu, Z.; and Yi, J. 2022. Automatic curriculum learning for large-scale cooperative multiagent systems. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 7(3): 912–930.

Zhao, C.; Zhuang, L.; Huang, Y.; and Liu, H. 2023. Curriculum learning-based multi-agent path finding for complex environments. In *2023 International Joint Conference on Neural Networks*, 1–8. IEEE.

Zheng, S.; Trott, A.; Srinivasa, S.; Parkes, D. C.; and Socher, R. 2022. The AI Economist: Taxation policy design via two-level deep multiagent reinforcement learning. *Science advances*, 8(18): eabk2607.