ingegneria sismica
International Journal of Earthquake Engineering
Anno XLIII - Num. 1 - 2026

A Multi-Intelligent Body Collaboration Framework for Complex Task Decomposition Based on Large Language Modeling

Houzhuo Wu[1,*]

[1] Applied Math and Statistics Department, Johns Hopkins University, Baltimore, MD, USA

SUMMARY: *With the breakthrough progress of the large language model in the field of natural language processing, the LLM-based intelligent body technology is gradually moving from theoretical research to practical application. The study first clarifies the complex task decomposition representation, takes reinforcement learning and multi-intelligent body reinforcement learning as the theoretical basis, elaborates the principle of LLaMA large language model, then introduces the PPO algorithm for strategy optimization, obtains the reward signal by interacting with the environment, and uses the dominance function and the pruning strategy to ensure the stability of the training and the convergence, realizing the application of large language model in the task decomposition. Finally, experiments are conducted in home and warehouse task scenarios for analysis. The results show that the performance of the model after fine-tuning using the PPO algorithm is significantly improved, and the average reward value during task decomposition is increased from 0.141 to 0.780, and the efficiency and stability of task decomposition are better than the original model. In the A/B test, LLaMA-PPO has a 3.46% improvement, which shows that the algorithm's training speed and final performance have been improved while the task decomposition has been automated efficiently.*

KEYWORDS: *lLaMA large language model; reinforcement learning; PPO; task decomposition*

1 Introduction

General Artificial Intelligence (AGI) is an AI system capable of thinking, learning, and performing multiple tasks like humans, and its realization requires human-like logical reasoning and autonomous decision-making capabilities [1]. In recent years, AGI has demonstrated impressive results in a variety of fields such as human-computer interaction, automated multimodal tasks, and plays an important role in applications related to natural language understanding [2, 3]. An AI intelligence is an intelligent body that can perceive its surroundings and make decisions based on these perceptions in order to achieve specific goals [4]. It is widely recognized in the current field of artificial intelligence that AI intelligences are the main research direction leading to AGI. In the early stages of AI research, researchers relied on symbolic logic and reinforcement learning to construct intelligent bodies, resulting in major achievements such as AlphaGo and DQN algorithms [5]. In recent years, large language models (LLMs) such as ChatGPT and Deepseek have demonstrated powerful effects in several fields with their rich knowledge bases and strong reasoning capabilities, and LLMs have thus been used as the basis for building intelligent bodies and have made significant progress [6].

*hwu107@alumni.jh.edu
https://doi.org/10.65102/is2026100
Published: 30 April, 2026

Multi-intelligence collaborative learning is an important research direction in the field of reinforcement learning, which involves multiple intelligences learning by cooperating with each other to achieve a common goal, and this type of learning has significant advantages in solving those complex tasks that require multiple intelligences to work together [7, 8]. Chen, W et al [9] proposed a multi-intelligence body framework called Agent Verse, which introduces the concepts of expert recruitment, collaborative decision making, etc. It contains two frameworks of task solving and simulation, the former assembles multiple intelligences into an automated multi-intelligence body system to accomplish a task, and the latter allows the user to set up customized environments for observing the interactive behaviors among multiple intelligences. The QMIX algorithm proposed by Rashid, T et al [10] further develops the idea of VDN by introducing a hybrid network to learn a global value function, and then using monotonicity constraints to ensure that the global value function is monotonically increasing with respect to the local value function of any one of the intelligentsia, thus ensuring consistency of the global optimal solution. Ding, Z [11] proposed an I2C (Individually Inferred Communication) framework that allows intelligences to infer the information that should be communicated by observing the environment and the behavior of other intelligences, which reduces the dependence on communication bandwidth while still maintaining a high level of efficiency in task execution. Qin, W et al [12] modeled the task allocation problem as a multi-intelligence stochastic game process for the dynamic task allocation problem in urban transportation systems, and proposed a dynamic task allocation method based on multi-intelligence reinforcement learning.

Earlier, rule-based construction of intelligences was a common approach to encode domain knowledge into a system in the form of logical rules by predefining a set of rules, according to which the intelligences reason and make decisions [13]. Foerster, J et al [14] proposed the Multi-Intelligent Intelligence Reinforcement Learning (DIAL) scheme, which establishes a conductive communication protocol between multiple intelligences, allowing the learning process of the entire system to be optimized end-to-end, and thus solving tasks collaboratively. Wen, M et al [15] transformed the multi-intelligent body reinforcement learning problem into a sequence modeling task processing, cleverly combining the advantages of the Transformer structure in processing sequence data and capturing long-distance dependencies, by efficiently modeling the dynamic interaction histories and decision dependencies among the intelligences, so as to enable the intelligences to make rational action decisions based on the interaction histories in a more effective way. Lowe, R et al [16] proposed the Multi-Intelligent Deep Deterministic Policy Gradient (MADDPG) method, which uses the information from other intelligences to synthesize and evaluate decisions during training, and uses their respective actor networks to make decisions during execution.The MADDPG method presents the first Centralized Training Distributed Execution (CTDE) architecture, which is also the most dominant multi-intelligent reinforcement learning architecture. Xue, X et al [17] proposed Co-evolutionary Multi-Intelligent System (CoMAS), which uses the difference in the overall rewards of the intelligences after changing their actions as a baseline, employs LLM as a refereeing mechanism to formulate these rewards and optimizes each intelligence curving paths through RL, thus achieving decentralized co-evolution.

The learning collaboration class of methods does not learn inter-intelligence communication, and at the end of training, each intelligence autonomously makes a decision to complete a collaborative task based only on its own observations, this class of methods can be divided into actor-critic based methods [18] and value decomposition based methods [19] and two categories. Among the actor-critic based methods, Iqbal, S and Sha, F [20] proposed Multi-Actor Attention Critic (MAAC), which is innovative in that it improves the learning process of the intelligences through the use of an attention mechanism that allows each intelligence to

consider the strategies and states of the other intelligences more efficiently when making decisions. The counterfactual Multi-Intelligent Actor-Critic Approach (COMA) proposed by Foerster, J et al [21], which on the other hand, focuses on the Multi-Intelligent Credit Allocation Problem by marginalizing the actions of one intelligent while keeping the actions of the others unchanged by calculating the Q-values of individual intelligences using a counterfactual baseline. Among the value decomposition based approaches, Etheve, M et al [22] proposed for the first time the use of reinforcement learning to go for branching variable selection by treating the MILP sample as an environment and solving it using intelligences, allowing the two to learn in interaction, and experimentally proved its excellent performance in complex scenarios. Mnih, V et al [23] proposed a lightweight asynchronous method for deep reinforcement learning, which is called A3C algorithm, which uses multi-core and multi-threaded CPUs to process multiple independent intelligences, and through asynchronous and parallel approach, it allows intelligences to be trained to learn by employing different strategies in different environments at the same time, which breaks empirical correlation of training and reduces the training time. CommNet, proposed by Sukhbaatar, S and Fergus, R [24], on the other hand, facilitates the exchange of information between intelligences by constructing a shared communication network, and this centralized communication architecture simplifies the process of information exchange and helps to improve the collaborative capabilities of intelligences. Schulman, J et al [25] proposed a proximal policy optimization (PPO) algorithm based on the AC framework, which uses a CLIP approach to limit the difference between the old and the new policies to a certain range, and ensures a steady improvement in the performance of the intelligences by reducing the sensitivity of the update of the policy gradient.

With the rise of LLM and the discovery of the emergent capabilities that accompany it, the ability of intelligences to handle complex reasoning tasks has been significantly improved [26]. LLM-driven intelligences are able to handle complex reasoning tasks more efficiently by utilizing the powerful language comprehension and generative capabilities of LLM in combination with various innovative reasoning mechanisms and architectures. For example, Li, X et al [27] considered LLM-based multi-intelligent body systems as a promising way to realize artificial intelligence comparable to or beyond human intelligence, for which a systematic review of LLM-based multi-intelligent body systems was conducted, pointing out its five key generalized structures, i.e., profiles, perceptions, autonomous actions, interactions, and evolutions. Hong, S et al [28] proposed an innovative metaprogramming framework (MetaGPT) based on a large population of intelligent agents in a Large Language Model (LLM), which is capable of incorporating efficient human workflows into LLM-based collaboration of multi-agent intelligences. Tran, K T et al [29] proposed a multi-intelligent body system in LLM that is capable of coordinating and working together to solve complex tasks at scale, moving away from isolated models to a collaborative-centered approach for solving complex real-world use cases. Talebirad, Y and Nadiri, A [30] proposed a new framework for enhancing the capabilities of LLM using a multi-intelligent body system, which introduces a multi-intelligent body collaborative environment by having multiple intelligent agent components with unique attributes and roles working together in order to handle complex tasks more efficiently and effectively. Pan, B et al [31] proposed a framework for visualizing the design of collaboration strategies for intelligences (Agent Coord), which establishes a structured representation to explicitly articulate entities and their relationships during collaboration, providing an interactive visual interface for designing, editing, and debugging learned collaboration strategies for LLM intelligences.

However, the single application of LLM in specific application scenarios still suffers from the problems of weak timeliness, poor adaptability, and lack of specialization. To address the limitations of a single LLM, studies have proposed a large model-driven multi-intelligence

body collaboration architecture to accomplish complex tasks, goals, or problems through the cognitive synergy of multiple autonomous intelligences. Zhang, Y et al [32] proposed a new framework for multi-intelligence collaboration in order to enhance the reasoning ability of LLMs to perform embodied tasks, which introduces Reinforcement Advantage Feedback (ReAd) to efficiently self-optimize the plan, as well as endowing LLMs with an anticipatory ability to discriminate the action through a sequential advantage function. Li, H et al [33] conducted a systematic assessment of the performance of LLM's intelligences in multi-intelligence collaborations, with a particular focus on the intelligences' abilities in a task with Theory of Mind (ToM) inference, aiming to understand whether these intelligences are able to simulate and understand the intentions, beliefs and emotions of the other intelligences, thus enabling a more complex decomposition of the task amongst them. Xu, Y et al [34] proposed a no-training required MultiAgentESC (MultiAgentESC) framework, which aims to simulate the process of providing emotional support by humans through three phases including dialog analysis, strategy deliberation, and response generation, and introduces a novel response strategy that realizes the selection of optimal responses for MultiAgentESC. Agashe, S et al [35] proposed a LLM coordination benchmark framework, which is a new benchmark for analyzing LLM in the context of a multi-intelligent body coordination setting, and experiments have shown that, unlike reinforcement learning approaches, the framework is able to remain more adaptive to unencountered collaboration partners. In addition, although autonomy and consistency criteria have a fundamental role in the architectural design of multi-intelligence collaboration frameworks, and it has been demonstrated that autonomy and consistency have a large impact on the operational outcomes of multi-intelligence collaboration based on large models. However, current research on LLM-based multi-intelligence collaboration frameworks has not yet combined autonomy and consistency metrics to analyze system performance. Therefore, there are still many limitations of existing multi-intelligent body collaboration frameworks in complex task decomposition.

The research centers around the problem of efficient decomposition of complex tasks, and proposes a multi-intelligence body task decomposition system that combines the LLaMA large language model (with the PPO algorithm. Firstly, it clarifies the representation of complex tasks, strengthens the learning with Markov decision process, value function and other elements, thus providing a collaborative framework for the multi-intelligent bodies constructed on the basis of it, and then introduces the LLaMA large language model, which enhances the quality of decision-making of the intelligent bodies through its powerful semantic understanding and generative ability, and combines the LLaMA model with the PPO algorithm for strategy optimization. Finally, experimental data analysis is carried out in smart home and warehouse scenarios to verify the effectiveness of the proposed method in task decomposition performance and scene adaptability.

2 Method

2.1 Complex task representation

The goal of the multi-robot task decomposition problem is to decompose a submitted complex task into multiple subtasks with the highest possible parallelism. For the expression of a task, the task is defined as a quintuple (ID, NA, CO, S, L), where ID is the unique encoding of the task, NA is the name of the task, CO is the constraints of the task, S denotes the structure of the task, including subtasks and dependencies between subtasks, and L denotes the type of the task, which can be divided into simple tasks and complex tasks, where simple tasks are activities that can be performed by a single robot. S represents the structure of the task, including the subtasks

that can be decomposed from the task and the dependencies between subtasks, and L represents the type of the task, which can be categorized into simple and complex tasks, where simple tasks are activities that can be completed by a single robot.

The dependencies between subtasks mainly include time dependency, resource dependency and other dependencies. Among them, time dependence is not exactly equivalent to the temporal sequential relationship, but refers to the special temporal relationship that must be satisfied between two tasks, for example, a task must be executed immediately after another task, emphasizing that it is executed immediately after rather than after it. Resource dependency refers to the fact that a task needs to rely on the resources of the previous task for its execution, and is categorized into information dependency and material dependency. Other dependencies are special requirements that arise under certain circumstances.

Each task may have multiple dependencies at the same time, and among these dependencies, resource dependency is the most direct and basic dependency in multi-robot complex tasks, and this kind of dependency usually makes the two subtasks have logical or temporal order, that is, time dependency is generally accompanied by resource dependency.

2.2 Enhanced learning

2.2.1 Markov decision-making process

Markov Decision Process (MDP) is a classical expression of sequential decision making and a mathematically ideal form of reinforcement learning. The Markov Decision Process is usually represented by the quintuple $\langle S, A, R, P, \gamma \rangle$, where S is the state space of the intelligences in the environment, and the state includes comprehensive information about the intelligences' interactions with the environment; A is the space of the intelligences' actions; and $R: S \times A \to R$ is the reward function; $P: S \times A \times S \times R[0,1]$ is the state transfer function; $\gamma \in [0,1]$ is the discount factor. Specifically, after a period of training, reinforcement learning obtains the state $s_t \in S$ of the intelligent body at time step t, on which the executable action $a_t \in A$ is selected, and at the moment of $t+1$ the intelligent body receives the numerical reward $r_{t+1} \in R$, and enters the state s_{t+1}. Where the values of r_{t+1}, s_{t+1} depend only on the previous state and action, the probability of the state and reward appearing at the moment $t+1$ is shown in (1) via the state transfer function of the environment dynamics and it follows that the sequential trajectories possessing Markovianity are $s_t, a_t, r_{t+1}, s_{t+1}...$ Note: $\doteq$ is denoted as a two-sided equality relation by definition.

$$p(s', r|s, a) \doteq \Pr\{S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a\} \tag{1}$$

2.2.2 Incentives and rewards

Reward has a crucial role in reinforcement learning methods, it is the feedback of the interaction between the intelligent body and the environment, which can assist the intelligent body to find the way forward and reach the goal. Reward is generally a numerical constant obtained by an intelligent body after performing an action in a certain state, which can be obtained by the intelligent body reaching a certain state or by the intelligent body performing a certain action. For example, in a Go game, an intelligent body is rewarded for winning or losing the game; in autonomous driving, an intelligent body performs an emergency avoidance action and is rewarded for ensuring the safety of the personnel.

One introduces the concept of reward to denote the long-term cumulative gain of an

intelligent body, and the reward for time step t is as follows:

$$G_t \doteq r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T \tag{2}$$

where T is the termination moment.

If the task is a continuous task and no termination condition is set, the long term cumulative gain will easily tend to infinity. Therefore, the discount factor γ is introduced to redefine equation (3):

$$\begin{aligned} G_t &\doteq r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^k r_{t+k+1} \\ &= r_{t+1} + \gamma\left(r_{t+2} + \gamma r_{t+3} + \cdots + \gamma^{k-1} r_{t+k+1}\right) \\ &= r_{t+1} + G_{t+1} = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \end{aligned} \tag{3}$$

From equation (3), the discount factor γ determines the multiple of the reward at future time step k to the current value. When $\gamma = 0$, the intelligent body only focuses on the current benefit and will become short-sighted; when $\gamma = 1$, the intelligent body becomes long-sighted, but the infinite sequence of rewards will cause the system to fail to converge; and when $0 < \gamma < 1$ and $\{r_k\}$ is bounded, the intelligent body will be both far-sighted and the reward will be be limited. If the reward is not properly designed, it will affect the long-term benefits and cause the intelligence to fail to complete the required task.

2.2.3 Strategies

A strategy is a mapping of the probability of an intelligent body choosing an executable action in a state, and this mapping is generally denoted by $\pi(\cdot)$. When the intelligent body chooses the strategy π at time step t there is:

$$\pi(a|s) = \Pr\{A_t = a | S_t = s\} \tag{4}$$

The meaning of Eq. (4) is that the $A_t = a$ probability when the state $S_t = s$ of the intelligent body, i.e., in the state $S_t = s$, the action of the intelligent body is selective. The strategy π defines a probability distribution of actions for all states in the state space S, and this strategy is generally known as the randomness strategy.

When the strategy is deterministic, the action is unique in the same state, as in equation (5):

$$a = \mu(s) \tag{5}$$

The learning of data by reinforcement learning algorithms affects the strategies of the intelligences, and the merit of the strategies directly determines the behavioral capabilities of the intelligences, whose merit is generally represented by a value function.

2.2.4 Value functions

Reinforcement learning focuses on the strategic process by which an intelligent body maximizes the cumulative reward. In this process, two types of value function equations are

defined to represent the strategy utility of an intelligent body, one is the state value function that evaluates the strategy utility in terms of the expected reward value of the state s, as in equation (6):

$$V_\pi(s) \doteq \mathbb{E}_\pi\left[G_t \middle| S_t = s\right] \tag{6}$$

The other type is the action value function that evaluates the utility of the strategy in terms of the expected payoff value of choosing action a in state s (also known as the state-action value function), as in equation (7):

$$Q_\pi(s,a) \doteq \mathbb{E}_\pi\left[G_t \middle| S_t = s, A_t = a\right] \tag{7}$$

where G_t of Eqs. (6) and (7) are consistent with Eq. (3), and thus the state value function can be rewritten as:

$$\begin{aligned} V_\pi(s) &\doteq \mathbb{E}_\pi\left[G_t \middle| S_t = s\right] = \mathbb{E}_\pi\left[r_{t+1} + G_{t+1} \middle| S_t = s\right] \\ &= \sum_{a\in A}\gamma(a\,|\,s)\sum_{s'\in S, r\in R} p\left(s', r \middle| s, a\right)\left[r_{t+1} + \gamma\mathbb{E}_\pi\left[G_{t+1} \middle| S_{t+1}\right]\right] \\ &= \sum_{a\in A}\gamma\left(a \middle| s\right)\sum_{s'\in S, r\in R} p\left(s', r \middle| s, a\right)\left[r_{t+1} + \gamma V_\pi(s')\right] \end{aligned} \tag{8}$$

The rewritten Eq. (8) expresses the recursive linkage of the state value function at neighboring moments, and the form is said to be the Bellman equation for the state value function under the strategy π.

Similarly, the action value function in the form of Eq. (7) is rewritten:

$$\begin{aligned} Q_\pi\left(s,a\right) &\doteq \mathbb{E}_\pi\left[G_t \middle| S_t = s, A_t = a\right] = \mathbb{E}_\pi\left[r_{t+1} + G_{t+1} \middle| S_t = s, A_t = a\right] \\ &= \sum_{s'\in S, r\in R} p\left(s', r \middle| s, a\right)\left[r_{t+1} + \gamma\sum_{a'\in A}\gamma\left(a' \middle| s'\right)Q_\pi\left(s', a'\right)\right] \end{aligned} \tag{9}$$

The Bellman optimality equation is obtained when the value of each state must be equal to the expected return of the optimal action in this state:

$$\begin{aligned} V_{\pi^*}\left(s\right) &= \max_{a\in\mathcal{A}(s)} Q_{\pi^*}\left(s,a\right) \\ &= \sum_{s_{t+1}, r} p\left(s_{t+1}, r \middle| s_t, a_t\right)\left[r + \gamma\max_{a\in\mathcal{A}(s)} Q_{\pi^*}\left(s_{t+1}, a_{t+1}\right)\right] \end{aligned} \tag{10}$$

From Eq. (10), it can be seen that solving Bellman's optimality equation is one of the ways for an intelligent body to find an optimal strategy.

2.3 Multi-intelligent Body Reinforcement Learning

2.3.1 Multi-Intelligence Markov Decision Processes

The interaction process between intelligences and the environment in MARL can be modeled using MDP or game processes as well. But the multi-intelligent body environment is again very different from the single-intelligent body environment. Therefore, the Markov decision process

needs to be redefined before studying multi-intelligence reinforcement learning.

In a multi-intelligent body environment, the intelligences can only observe the local environment. Therefore, MDP is extended to Partially Observable Markov Decision Process (POMDP). The task of multi-intelligent body reinforcement learning is to find a set of intelligent body strategies $\pi = \pi_1 \times \pi_2 \times \pi_4 \times \pi_4 \times \cdots \times \pi_N$, such that all the intelligences get the maximum overall cumulative discount under the guidance of the set of strategies Reward $E\left[R(t)\middle|\pi\right]$ or $E\left[\sum_{t=0}^{N} R_i R_i(t)\middle|\pi\right]$ in an independent environment. For intelligences other than intelligence i can be denoted as i^-. At this point the state value function is updated as:

$$V_i^{\pi}(s) = \sum_{a \in A} \pi(s,a) \sum_{s' \in S} P\left(s'\middle|s, a_i, a_{i^-}\right)\left[R_i\left(s, a_i, a_{i^-}\right) + \gamma V_i^{\pi}(s')\right] \tag{11}$$

Then, the optimal policy can be expressed as:

$$\pi_i^*\left(s, a_i, \pi_{i^-}\right) = \arg\max_{\pi_i} V_i^{\left(\pi_i, \pi_{i^-}\right)} \tag{12}$$

However it is very difficult to reach the optimal strategy for all intelligences at the same time. Therefore Nash equilibrium is introduced as a compromise. Although the Nash equilibrium is not necessarily the global optimum, the joint policy at this point is the easiest to obtain and the algorithm is easy to converge. A strategy that satisfies the Nash equilibrium can be expressed as follows: it is not possible to obtain a higher reward by updating again for any one intelligence:

$$V^i_{\pi_i^*, \pi_{i^-}} \geq V^i_{\pi_i, \pi_{i^-}^*}, \forall i \in N \tag{13}$$

2.3.2 Independent versus centralized learning

In multi-intelligent body reinforcement learning algorithms, the algorithms can be classified into independent learning algorithms and centralized learning algorithms according to the setting of the environment and the relationship between the intelligences and the learning process. The difference between the two lies in whether the intelligences are independent of each other, whether the reward functions are independent or joint, and so on. The MARL algorithm has different advantages and disadvantages in different settings. Therefore, before applying the algorithm and improving it, one needs to consider the specifics of the scenarios and environments to which it is applied.

(1) In stand-alone learning, each intelligence is independent of each other. There is no access to and acquisition of information from other intelligences, while each intelligence has its own reward function, and the independent multi-intelligence reinforcement learning process is shown in Figure 1. Each intelligent body optimizes its strategy independently of each other. In this setup, other intelligences are considered as part of the environment. In a sense, when the other intelligences are ignored, the problem is transformed into a single-intelligence task. The intelligences will only consider maximizing their own rewards when making decisions, without caring about the other intelligences.

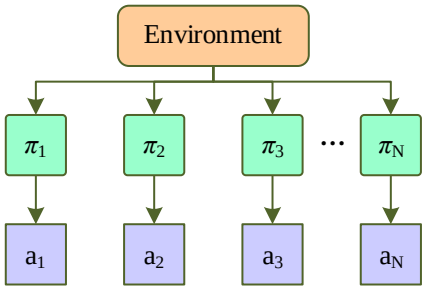


Figure 1: Schematic diagram of independent multi-agent reinforcement learning

This setup makes the problem very simple, but it also creates many problems. Independent learning ignores the fact that the migration of environmental states is a result of the joint actions of all the intelligences acting together. Attributing environmental rewards to one's own actions makes it difficult for an intelligent's strategy to converge, which is detrimental to the training of the algorithm.

Some stand-alone algorithms such as Independent Q Learning (IQL), Independent AC (IAC) and Independent Proximal Policy Optimization Algorithm (IPPO) are direct extensions of classical single-intelligent body reinforcement learning algorithms in MAS. These algorithms were used as tentative algorithms in the early stages of the development of multi-intelligent body reinforcement learning to look for possible flaws in specific applications.

(2) In centralized learning, the environment feeds back a uniform reward based on the joint actions of the intelligences. The most basic centralized learning algorithm will directly construct a joint policy. The strategy receives all the intelligences' observations of the environment and then outputs each intelligence's actions, thus realizing collaborative decision-making, and the centralized multi-intelligence mention reinforcement learning process is shown in Fig. 2. The method considers all the intelligences as a whole and jointly optimizes the joint policy to achieve the overall optimum. Such a setup is very suitable for solving teamwork and strong collaboration tasks.

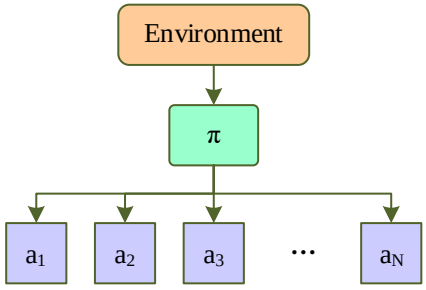


Figure 2: Schematic diagram of centralized multi-agent reinforcement learning

The classical centralized learning algorithm Value Decomposition Network (VDN) investigates this problem.VDN is improved for action value functions. The algorithm argues that the joint action value can be viewed as the sum of the action values of each intelligent in a centralized learning setting:

$$Q\left(\left(s_1, s_2, \cdots, s_n\right), \left(a_1, a_2, \cdots, a_n\right)\right) \approx \sum_{i=1}^{N} \tilde{Q}_i\left(s_i, a_i\right) \tag{14}$$

In other words, the joint Q value is decomposed to each intelligent body. In this way, each

intelligence is guided to optimize its strategy, avoiding the creation of “lazy intelligences”.

2.4 Multi-Intelligent Body Task Decomposition Incorporating LLaMA Large Models

2.4.1 LLaMA 2 model

LLM (Large Language Model) is a revolutionary technological breakthrough in the field of deep learning. Take OpenAI's ChatGPT as an example, it has shown excellent performance in understanding and generating natural language.Llama 2, like most of the mainstream LLM processing models nowadays, is built based on Transformer.The generative task of LLM is to predict the next word or token based on the contextual information of a given sequence of input text, so LLM models usually only need to use the Transformer Decoder part. Therefore, LLM models usually only need to use the Transformer Decoder part, and the so-called Decoder versus Encoder is the introduction of the Mask in the computation of $Q*K$ to ensure that the current position can only focus on what has been generated in front of the current position.The reason why LLM are mainly all using Decoder-only architecture, in addition to the training efficiency and In addition to the advantages in training efficiency and engineering implementation, theoretically, it is because Encoder's bidirectional attention Q will suffer from the low-rank problem, which may weaken the model expressive ability, and there is no substantial benefit of introducing bidirectional attention in terms of generative tasks. And the Encoder-Decoder architecture can perform better in some scenarios probably only because it has twice as many parameters. Therefore, with the same number of parameters and the same inference cost, the Decoder-only architecture is the optimal choice.

The model structure of Llama 2 is basically the same as the standard Transformer Decoder structure, which mainly consists of 32 Transformer Blocks, with differences in the use of a front-loaded RMSNorm layer; Q is positionally encoded using RoPE before being multiplied by K; and KVCache is used for inference performance optimization with Group Query Attention (GQA).

The Normalization layer in Transformer generally uses LayerNorm to normalize the Tensor, and RMSNorm is a variant of LayerNorm, RMSNorm omits the process of finding the mean and the bias β, i.e.:

$$y = \frac{x}{\sqrt{Mean\left(x^2\right)+\epsilon}} \times \gamma \tag{15}$$

$$Mean\left(x^2\right) = \frac{1}{N}\sum_{i=1}^{N} x_i^2 \tag{16}$$

where γ is a learnable parameter.

In the self-attention mechanism, the dot product of the query $\left(Q\right)$ and the key $\left(K\right)$ is a key step.Llama 2 optimizes on this by encoding the positional information by using Rotary Positional Encoding (RoPE) before the dot product is computed. This not only captures the absolute positional information of the words, but also takes into account the relative positional information, which enhances the model's understanding of the text structure. Let q be the original query vector and p be the positional encoding vector, the query vector q' after applying RoPE can be expressed as:

$$q' = q \odot \cos(p) + \sin(p) \tag{17}$$

where $\odot$ denotes per-element multiplication, and $\cos(p)$ and $\sin(p)$ are cosine and sine functions applied element by element to the position-encoded vector p.

In traditional attention mechanisms, each query vector is processed separately.Llama 2 optimizes this by grouping multiple queries and processing them in a single operation. If Q is the query matrix, K is the key matrix, and V is the value matrix, then the attention output A can be computed using grouped query attention:

$$A = Soft\max\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{18}$$

where d_k is the dimension of the key vector. The $K-V$ cache is used to store the previous K and V matrices to avoid redundant computations and thus improve computational efficiency.

2.4.2 Efficient fine-tuning of parameters

Parameter Efficient Fine-Tuning (PEFT) is a class of techniques aimed at reducing the need for parameter updates during the fine-tuning of large language models. This class of methods includes cued fine-tuning, prefix fine-tuning, and low-rank adaptation. These methods effectively reduce computational overhead and resource requirements by avoiding large-scale updating of the parameters of the entire model by tuning specific parts of the input or hidden layers, or by introducing low-rank matrix approximation for weight updating.

Since LLaMA is a large-scale model with a large number of parameters, directly training the whole model is not only computationally expensive, but also prone to overfitting problems. For this reason, we adopt LoRA for fine-tuning.The basic principle of LoRA is that in the fine-tuning process of the large-scale model, the weight matrix $W \in \mathbb{R}^{d\times k}$ is decomposed into two low-rank matrices $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times k}$, where $r << \min(d,k)$, so the weight update takes the form:

$$\Delta W = AB \tag{19}$$

By using a low-rank matrix instead of the full weight matrix, LoRA is able to reduce the number of parameters and computational resources required for fine-tuning, while preserving the performance of the model.

2.4.3 Overview of complex task decomposition methods

1. Initial Policy Generation

Since pre-trained language models have strong semantic understanding and text generation capabilities, we can directly use them for initial policy generation. However, due to environmental constraints, the actions directly generated by a large language model may not be applicable to the current environment. For example, the model may predict the action “pick up the milk”, but if the milk does not exist in the environment, the action will not be executed, resulting in a misalignment between the action and the environment information.

To solve this problem, we construct a set of executable actions $A = \{a1, a2 \cdots ak\}$ based on

the objects and their states in the current environment, where k represents the size of the set of actions, and all the actions to be executed are selected from this set. Subsequently, we input the environment observation information S together with the set of actions A into the large language model, so that the model selects the optimal actions for execution based on the token probabilities corresponding to each action in A. This approach ensures that the generated actions always conform to the environmental constraints, which improves the feasibility and stability of task planning.

For each action a_i, the model calculates the joint probability $P\left(a_i \middle| S\right)$ of its token sequence, where S is the environmental cue, i.e., the observation information received by the model. The joint probability can be expressed by the following equation:

$$P\left(a_i \middle| S\right) = \prod_{t=1}^{k} P\left(w_t^i \middle| w_1^i, w_2^i, \cdots, w_{t-1}^i, S\right) \tag{20}$$

where w_t^i denotes the t th token in the action a_i, and k is the total number of tokens contained in the action, the formula represents the probability that the model predicts the current token based on the previous tokens and the environmental cue S.

The model calculates the joint probability of each action a_i in the action table and selects the action with the largest joint probability $P\left(a_i \middle| S\right)$ as the next action to be executed:

$$a^* = \arg\max_{a_i \in A} P\left(a_i \middle| S\right) \tag{21}$$

where a^* denotes the optimal action, i.e., the action finally selected by the model. $\arg\max_{a_i \in A}$ denotes the action a_i that maximizes the joint probability $P\left(a_i \middle| S\right)$ from the action set A.

In this way, the model is able to generate a good initial policy by selecting the most contextualized action from the action table A under the current environmental constraints.

2. Policy Optimization Process of PPO Algorithm for Fine-tuning LLaMA Model

When fine-tuning the LLaMA model, the LLaMA model generates text or action sequences (e.g., natural language action descriptions) that can be regarded as strategies π_θ, and each time the generated action sequences are sampled from the action space according to the current strategy. The strategies are optimized by PPO in each generation step to make their generated sequences more consistent with the task goals.

a.Model Output and Action Generation

In the LLaMA model, each action or token generated corresponds to a logits value, which can be converted to a probability distribution by Softmax. Given the current input sequence s_t, the action a_t generated by the LLaMA model is sampled from these rate distributions:

$$\pi_\theta\left(a_t \middle| s_t\right) = soft\max\left(f_\theta\left(s_t\right)\right) \tag{22}$$

where $f_\theta\left(s_t\right)$ is the logits of the LLaMA model output.

b.Calculation of dominance function

In the PPO algorithm, strategy updating is used to optimize the decision-making ability of the model through constant interaction and feedback, specifically, by calculating the dominance function and strategy gradient to guide the model to update its parameters. The dominance

function $\hat{A}_t$ is a measure of the relative superiority of an action in a given state. The dominance function is usually estimated by the value function $V(s_t)$.

c. Calculation of Policy Gradient

The PPO algorithm uses the strategy gradient method to update the model parameters. The goal of policy gradient is to maximize the desired reward, which is usually optimized by gradient ascent. For the strategy π_θ, the goal is to optimize the following objective function:

$$J(\theta) = E_{s_t \sim p^\pi}\left[\sum_{t=0}^{T} \hat{A}_t \log \pi_\theta\left(a_t \middle| s_t\right)\right] \tag{23}$$

where p^π denotes the policy-dependent representation of the distribution of states in the environment, i.e., the probability distribution of the intelligences visiting each state under a certain policy. T denotes the maximum time step. $\hat{A}_t$ is the dominance function that represents the dominance of taking action a_t from state s_t. $\pi_\theta\left(a_t \middle| s_t\right)$ is the output of the strategy network, which denotes the probability of taking action a_t from state s_t.

By optimizing the above objective function with gradient ascent, PPO achieves the improvement of the strategy in each time step.

d. Policy Update and Objective Optimization

After each generated action, we update the parameters of the LLaMA model through the policy gradient optimization algorithm. Specifically, the generative strategy π_θ of the model is updated by minimizing the PPO loss function $L^{PPO}(\theta)$. The gradient during each update can be expressed as:

$$\nabla_\theta L^{PPO}(\theta) = E_t\left[\nabla_\theta \min\left(r_t(\theta)\hat{A}_t, clip\left(r_t(\theta), 1 \pm \epsilon\right)\hat{A}_t\right)\right] \tag{24}$$

where $r_t(\theta)$ is the ratio between the old and new strategies: $r_t(\theta) = \dfrac{\pi_\theta\left(a_t \middle| s_t\right)}{\pi_{\theta_{old}}\left(a_t \middle| s_t\right)}$, $\pi_\theta\left(a_t \middle| s_t\right)$ is the probability of generating action a_t under the new strategy, $\pi_{\theta_{old}}\left(a_t \middle| s_t\right)$ is the probability of generating action a_t under the old strategy, $\hat{A}_t$ is the dominance function, and ϵ is the hyperparameter used to limit the strategy update.By introducing the CLIP operation, the PPO restricts the magnitude of the change in the strategy to avoid the strategy update from being too large, so as to maintain the stability of the training.

3 Results and Discussion

3.1 Home environment task decomposition experiment

3.1.1 Experimental environment

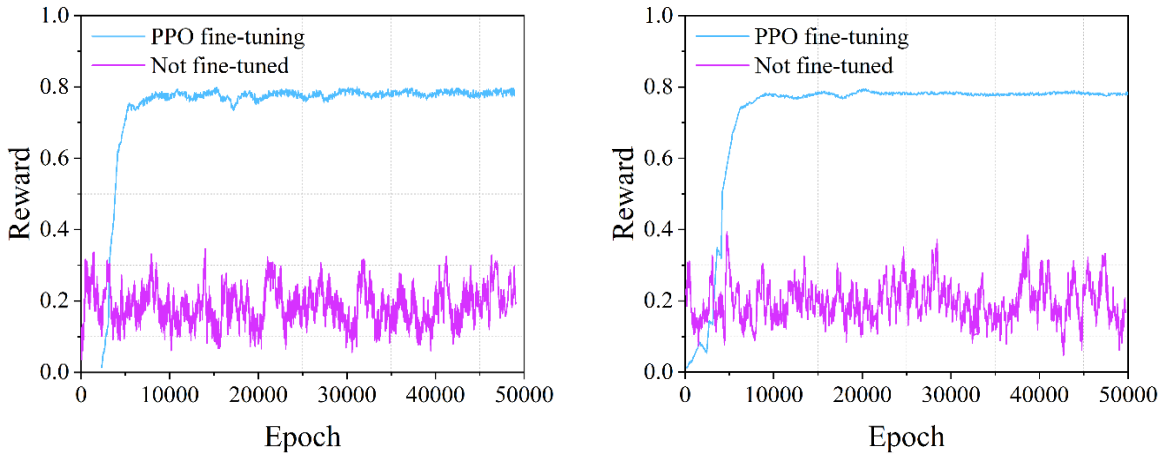
The performance of the model after fine-tuning using the PPO algorithm was evaluated during the experiment, and the specific evaluation environment was derived from the VirtualHome simulation environment, on which appropriate modifications were made to suit the needs of this experiment. In the VirtualHome environment, the intelligent body is in a home environment

with a full range of items.

3.1.2 Comparison of model performance before and after fine-tuning the PPO algorithm

In this experiment, the cumulative reward return for each task executed during the training process and the length of action steps required for each task were recorded and visualized. The visualization result of the cumulative reward of the task by the PPO algorithm is shown in Figure 3, and the visualization result of the impact of the PPO algorithm on the number of steps required for the task is shown in Figure 4. (a) and (b) respectively represent the task execution of heating salmon and pouring water. The blue curve indicates the result optimized by the PPO algorithm, while the purple curve represents the result without the PPO algorithm optimization.

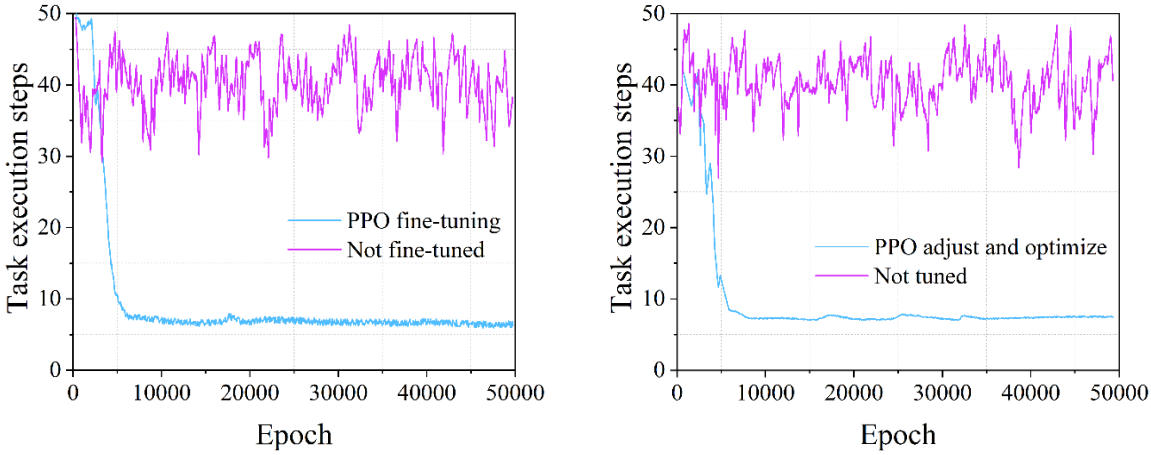
Analysis of the data in Fig. 3 shows that the performance of task planning after fine-tuning the LLaMA model using PPO is significantly better than task planning using the LLaMA model directly. The cumulative reward of the fine-tuned model using the PPO algorithm can quickly converge and stabilize around 0.79 during the training process, showing the efficiency and stability of the model. On the other hand, the model without fine-tuning using the PPO algorithm shows larger fluctuations, and its cumulative reward stays between 0-0.3 for a long time.



(a) Heating Salmon (b) Pouring water

Figure 3: The effect of PPO algorithm on task cumulative reward

Analysis of the data in Figure 4 shows that the model fine-tuned with the PPO algorithm performs significantly better than the unfine-tuned model during training. The model fine-tuned with the PPO algorithm rapidly reduced the number of steps in the task decomposition at the beginning and eventually stabilized at around 8 steps. The model without PPO fine-tuning, on the other hand, showed large fluctuations during training, with the number of task decomposition steps remaining between 30 and 50 for a long time.



(a) Heating Salmon (b) Pouring water

Figure 4: The effect of PPO algorithm on the number of steps required for a task

In summary, PPO fine-tuning improves the convergence speed and task planning efficiency of the model and ensures the stability and efficiency of the model in performing complex tasks.

3.1.3 Task generalization performance tests

Thanks to the open-vocabulary nature of the large language model, the trained model is able to transfer the acquired skills to a variety of different tasks, which is difficult with traditional reinforcement learning. The generalization ability of the fine-tuned model as well as the non-fine-tuned model on new unknown tasks was then evaluated, and the results are shown in Table 1. Five tasks were redefined, and for the tasks of “pouring coffee”, “heating milk”, and “washing clothes”, the water and salmon in the original tasks of pouring water and heating salmon were replaced with the corresponding food and items, respectively. The fourth and fifth tasks are cross-experiments, i.e., the model trained in one of the task scenarios is used to reason about the success rate of the other task. The fine-tuned model consistently achieved a high success rate of 100% on the tasks “pour coffee”, “heat milk”, and “do laundry”. In contrast, the model that was not fine-tuned performed significantly worse, with a success rate of only between 0.53 and 0.67 on these tasks. For the two tasks where cross-experimentation was performed, the model still showed a high level of accuracy. In the task of pouring water, the performance of the model declined but the fine-tuned model still maintained a success rate of 89%, whereas the model that was not fine-tuned performed poorly with a success rate of only 9%.

The results show that fine-tuning using the PPO algorithm significantly improves the model's ability to generalize to new tasks and achieves near-perfect performance in most cases. The difference in performance between the fine-tuned model and the model without fine-tuning highlights the importance of the PPO algorithm in improving the model's ability to maintain generalization in unknown environments.

Table 1: Test results of the model's task generalization ability

Assignment	PPO fine-tuning	Not fine-tuned
Pour coffee	1	0.53
Heated milk	1	0.64
Do the laundry	1	0.67
Heated salmon	1	0.58
Pouring water	0.89	0.09

3.2 Warehouse Task Decomposition Experiment

3.2.1 Offline warehouse testing

Warehouse is a large-scale realistic and complex environment, on the one hand, it means that the environment of the warehouse is not stable, which is a challenge to the robustness of the algorithm, on the other hand, it means that the warehouse algorithm on-line has a great cost of trial and error, and the bad effect of the algorithm will directly affect the warehouse's income. So although the reinforcement learning algorithm itself is used for global optimization, but because of the complexity of the environment, as well as the reinforcement learning part of the transformation of the part is too much, it is necessary to test the stability of the global optimization method, i.e., the reinforcement learning part of the test alone. Two order pools from May 2024 as well as mid-February 2025 were selected, and the reward functions had the R network learned by LLaMA, as well as the old heuristic rules. Then the optimization process of picking order generation strategy is tested under these four configurations, and the optimization results are shown in Fig. 5, where the purple line is vanill's TD method, the green and blue lines are the TD using target network, and the red line is the TD using slow update. As can be seen in the figure, the algorithm is able to obtain some positive optimization effect of 2.5% 5.1% for all four different configurations.

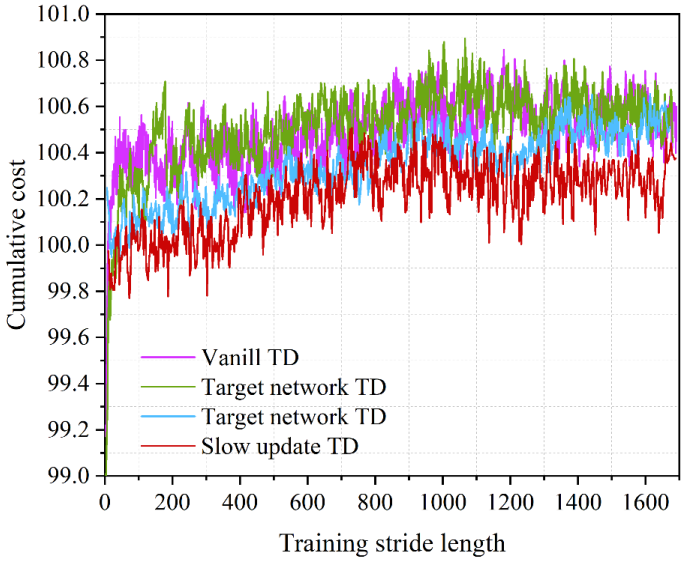


Figure 5: LLaMA score curve in environment

3.2.2 Comparative testing of the practical effectiveness of TD methods

As can be seen in Fig. 5, all TD methods have a rising cumulative cost of picking from the initial value, i.e., “the more they learn, the worse they get”. Moreover, it is noted that since the network parameters obtained from MC training are used as the initial values for TD training, and the fact that the TD methods get worse as they learn means that the TD methods are unable to learn a better result than the MC methods.

Further, the output mean of the V network trained by the TD method is tracked in real time and the tracking results are shown in Fig. 6. In the figure, the V values obtained by all the TD methods are constantly shifted towards the coordinate axis, in which the methods such as target network and slow update can slow down the shifting speed of the V values, but cannot cure the overfitting problem of the V values.

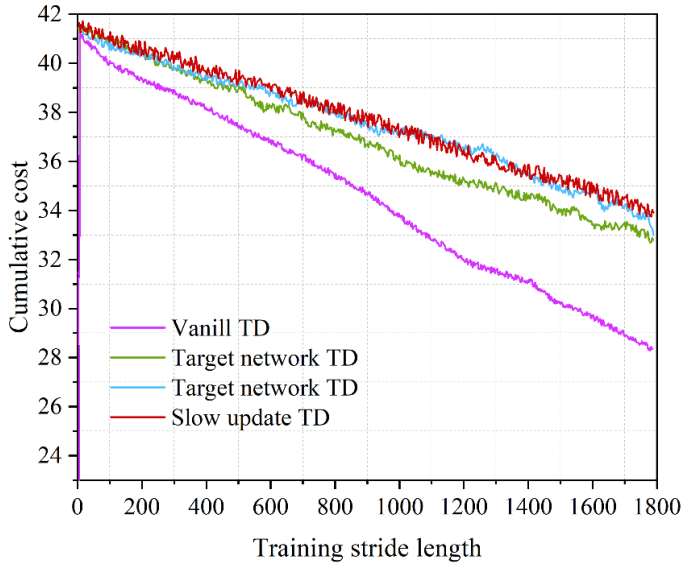


Figure 6: The mean curve of v values fitted by TD

3.2.3 Virtual warehouse optimization applications

The virtual warehouse is now applied to the order combination problem. The optimization method uses an improved version of a local search, a simple and widely used method on combinatorial optimization. In contrast to the improved optimization method, we use the virtual warehouse to reconstruct the objective function, and the next results will include both offline and online experiments.

1. Offline experiments

We use $s(p)$ to denote the number of different sku types included in the picking order p in total, $u(p)$ to denote the number of different zones included in the picking order (the whole warehouse is divided into 10 large zones), and $v(p)$ to denote the number of different aisles. The objective function based on expert knowledge is defined as follows: $f_1(p) = \alpha \cdot s(p) + \beta \cdot u(p) + \gamma \cdot v(p)$, i.e., we want to combine the most similar orders together. The α, β, γ in Eq. are hyperparameters. We used the following objective function as a comparison between the offline and online experiments:

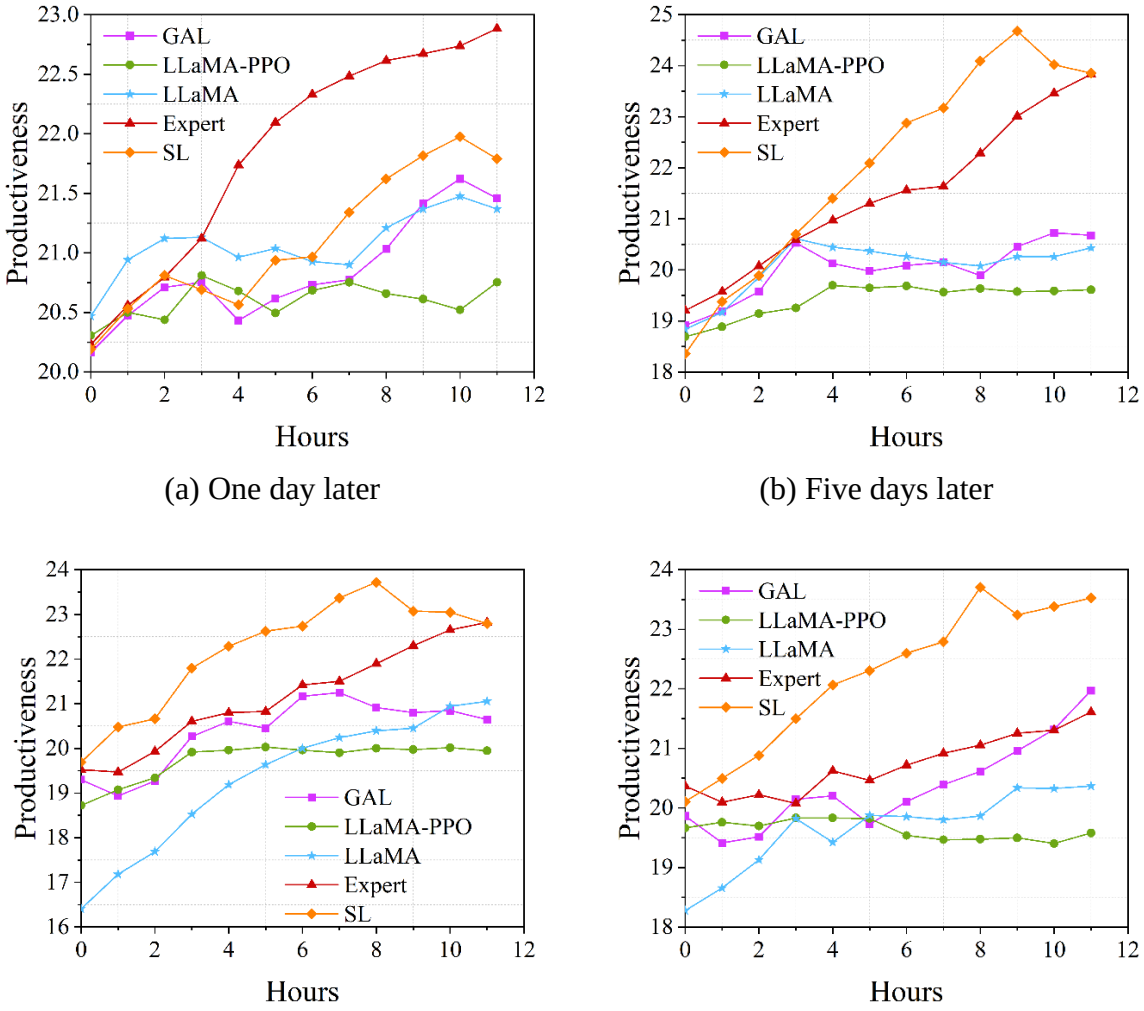
$$f_1(p) = 0.033s(p) + 0.2v(p) + 2u(p) \tag{25}$$

The above hyperparameters are manually tuned by experts based on historical long-term performance.

According to the virtual warehouse, the objective function can be directly transformed into the average time spent on picking each sku: $f_2(p) = t/n$, where t denotes the estimated picking time of a picking order simulated by the virtual warehouse, and n denotes the total number of sku's in the picking order. For simplicity, all commercial restrictions in the offline experiments were removed, leaving only one restriction, i.e., the number of orders contained in each picking order, fixed at 18. We trained the virtual warehouse using different methods, all of which were trained with 5 days of data. We simultaneously trained another virtual warehouse for testing, using data after 1, 5, 10, and 15 days. Different objective functions were applied in these four test environments to show the average picking time per sku under long running time, and the results are shown in Fig. 7.

When the virtual and real warehouses are close enough, we are able to achieve a large

improvement in picking efficiency. At the beginning, virtual warehouses trained with different methods achieve good simulation results and some improvement in picking efficiency. The simulation results become worse as time increases, because the environment also changes over time. In the figure it can be seen that the LLaMA and LLaMA-PPO methods show the best generalization and achieve the best improvement in picking efficiency. Also the smoother curve of LLaMA-PPO demonstrates its ability to act as a global optimization.



(a) One day later

(b) Five days later

(c) Ten days later

(d) 15 days later

Figure 7: Comparison of Different Objective Functions on the Hourly Dimension

2. Online Experiments

Next, a virtual warehouse was applied in a real warehouse and A/B tests were conducted to test whether the LLaMA-PPO approach can find a better solution for the order pooling problem. First, we introduce the A/B test scenario: in a real warehouse, the order pool continuously receives consumer orders from the Taobao platform. The order combination engine performs the parsing work and then generates a picking order every few minutes. The optimization algorithm used to solve the order combining problem is an adaptive local search algorithm that works to minimize the current objective function. The picking worker then picks the generated picking order to completion.

In the A/B test, because of resource constraints, two objective functions were tested: LLaMA, LLaMA-PPO, respectively. At each triggering of the order combination engine, the

objective function will randomly select one from LLaMA and LLaMA-PPO with a probability of 50% each. The picking orders are divided into two categories according to the selected objective function, and these two categories of picking orders are compared in terms of real picking efficiency.Both LLaMA and LLaMA-PPO are trained using one month's data, and the comparison is mainly to check the generalization of the RL policies trained from the simulator, and the training results are shown in Table 2.Both LLaMA and LLaMA-PPO are trained using one month's data for training, and their comparison is mainly to test the generalizability of RL strategies trained from the simulator, and the results show that LLaMA-PPO has a 3.46% improvement in picking efficiency compared to LLaMA, implying that the simulator constructed using LLaMA-PPO is still accurate enough to train an RL strategy that has practical value in the real world.

Table 2: Comparison of average picking time of LLaMA and LLaMA-PPO in days

	Ten days later	11 days later	12 days later	13 days later	14 days later	15 days later	16 days later	17 days later	Average
LLaMA-PPO	29.34	26.42	30.39	26.88	27.39	26.71	27.29	27.75	27.75
LLaMA	28.15	24.55	28.63	26.73	26.62	26.21	26.33	26.93	26.79
Improvement (%)	4.06%	7.08%	5.79%	0.56%	2.81%	1.87%	3.52%	2.95%	3.46%

4 Conclusion

In order to further improve the efficiency and success rate of complex task decomposition, through the parameter efficient fine-tuning technique, the LLaMA model is able to generate the initial policy efficiently with limited computational resources, and a task decomposition system incorporating the LLaMA model (LLaMA-PPO) is proposed in combination with the PPO algorithm. The experimental part uses home and warehouse environments to verify the decomposition performance of LLaMA-PPO. The experimental results show that the LLaMA model, when combined with the PPO method, has a task success rate of as much as 1, which significantly improves the correctness and execution efficiency of its task decomposition, and effectively solves the dynamic decision-making problem faced by large models in complex environments. And the LLaMA-PPO model has a 3.46% improvement in picking efficiency compared to the traditional LLaMA model, which proves that the method in this paper also has a significant performance improvement strategy in the real environment.

About the Author

Houzhuo Wu, a graduated student from Johns Hopkins University was born in Harbin, Heilongjiang, in 1999. He obtained a master's degree from Johns Hopkins University and a bachelor degree from University of California, Irvine. His main research interets are LLM, Multiagent and reinforcement learning.

References

[1] Obaid, O. I. (2023). From machine learning to artificial general intelligence: A roadmap and implications. Mesopotamian Journal of Big Data, 2023, 81-91.

[2] Xu, W., Dainoff, M. J., Ge, L., & Gao, Z. (2021). From human-computer interaction to human-AI Interaction: new challenges and opportunities for enabling human-centered AI. arXiv preprint arXiv:2105.05424, 5(10.1080), 10447318-2022.

[3] Bryndin, E. (2025). Formation of Motivated Adaptive Erudite AGI Twin with Reflexive Multimodal Ontology by Ensembles of Intelligent Agents. Research on Intelligent Manufacturing and Assembly, 4(2), 272-282.

[4] Marozau, R. (2025). Operational AGI: A Language-Based Approach for Adaptive Multi-Agent Systems. Authorea Preprints.

[5] Schaat, S., Wendt, A., Jakubec, M., Gelbard, F., Herret, L., & Dietrich, D. (2014, August). ARS: an AGI agent architecture. In International Conference on Artificial General Intelligence (pp. 155-164). Cham: Springer International Publishing.

[6] Phogat, R., Arora, D., Mehra, P. S., Sharma, J., & Chawla, D. (2025, March). A Comparative Study of Large Language Models: ChatGPT, DeepSeek, Claude and Qwen. In 2025 3rd International Conference on Device Intelligence, Computing and Communication Technologies (DICCT) (pp. 609-613). IEEE.

[7] Gronauer, S., & Diepold, K. (2022). Multi-agent deep reinforcement learning: a survey. Artificial Intelligence Review, 55(2), 895-943.

[8] Feng, J., Li, H., Huang, M., Liu, S., Ou, W., Wang, Z., & Zhu, X. (2018, April). Learning to collaborate: Multi-scenario ranking via multi-agent reinforcement learning. In Proceedings of the 2018 World Wide Web Conference (pp. 1939-1948).

[9] Chen, W., Su, Y., Zuo, J., Yang, C., Yuan, C., Chan, C. M., ... & Zhou, J. (2023, October). Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In The Twelfth International Conference on Learning Representations.

[10] Rashid, T., Samvelyan, M., De Witt, C. S., Farquhar, G., Foerster, J., & Whiteson, S. (2020). Monotonic value function factorisation for deep multi-agent reinforcement learning. Journal of Machine Learning Research, 21(178), 1-51.

[11] Ding, Z., Huang, T., & Lu, Z. (2020). Learning individually inferred communication for multi-agent cooperation. Advances in neural information processing systems, 33, 22069-22079.

[12] Qin, W., Sun, Y. N., Zhuang, Z. L., Lu, Z. Y., & Zhou, Y. M. (2021). Multi-agent reinforcement learning-based dynamic task assignment for vehicles in urban transportation system. International Journal of Production Economics, 240, 108251.

[13] Calegari, R., Ciatto, G., Mascardi, V., & Omicini, A. (2021). Logic-based technologies for multi-agent systems: a systematic literature review. Autonomous Agents and Multi-Agent Systems, 35(1), 1.

[14] Foerster, J., Assael, I. A., De Freitas, N., & Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. Advances in neural information processing systems, 29.

[15] Wen, M., Kuba, J., Lin, R., Zhang, W., Wen, Y., Wang, J., & Yang, Y. (2022). Multi-agent reinforcement learning is a sequence modeling problem. Advances in Neural Information Processing Systems, 35, 16509-16521.

[16] Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Pieter Abbeel, O., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. Advances in neural information processing systems, 30.

[17] Xue, X., Zhou, Y., Zhang, G., Zhang, Z., Li, Y., Zhang, C., ... & Bai, L. (2025). CoMAS: Co-Evolving Multi-Agent Systems via Interaction Rewards. arXiv preprint arXiv:2510.08529.

[18] Estornell, A., Ton, J. F., Yao, Y., & Liu, Y. (2024). ACC-collab: An actor-critic approach to multi-agent LLM collaboration. arXiv preprint arXiv:2411.00053.

[19] Wang, S., Zhou, Y., Zhao, Z., Zhang, R., Shao, J., Chen, W., & Cheng, Y. (2025). Heterogeneous Value Decomposition Policy Fusion for Multi-Agent Cooperation. arXiv preprint arXiv:2502.02875.

[20] Iqbal, S., & Sha, F. (2019, May). Actor-attention-critic for multi-agent reinforcement learning. In International conference on machine learning (pp. 2961-2970). PMLR.

[21] Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., & Whiteson, S. (2018, April). Counterfactual multi-agent policy gradients. In Proceedings of the AAAI conference on artificial intelligence (Vol. 32, No. 1).

[22] Etheve, M., Alès, Z., Bissuel, C., Juan, O., & Kedad-Sidhoum, S. (2020, September). Reinforcement learning for variable selection in a branch and bound algorithm. In International conference on integration of constraint programming, artificial intelligence, and operations research (pp. 176-185). Cham: Springer International Publishing.

[23] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ... & Kavukcuoglu, K. (2016, June). Asynchronous methods for deep reinforcement learning. In International conference on machine learning (pp. 1928-1937). PmLR.

[24] Sukhbaatar, S., & Fergus, R. (2016). Learning multiagent communication with backpropagation. Advances in neural information processing systems, 29.

[25] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

[26] Chen, R., & He, C. (2025). Fostering collective intelligence in CPSS: an LLM-driven multi-agent cooperative tuning framework. Frontiers in Physics, 13, 1613499.

[27] Li, X., Wang, S., Zeng, S., Wu, Y., & Yang, Y. (2024). A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges. Vicinagearth, 1(1), 9.

[28] Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Wang, J., ... & Schmidhuber, J. (2023, August). MetaGPT: Meta programming for a multi-agent collaborative framework. In The Twelfth International Conference on Learning Representations.

[29] Tran, K. T., Dao, D., Nguyen, M. D., Pham, Q. V., O'Sullivan, B., & Nguyen, H. D. (2025). Multi-agent collaboration mechanisms: A survey of llms. arXiv preprint arXiv:2501.06322.

[30] Talebirad, Y., & Nadiri, A. (2023). Multi-agent collaboration: Harnessing the power of intelligent llm agents. arXiv preprint arXiv:2306.03314.

[31] Pan, B., Lu, J., Wang, K., Zheng, L., Wen, Z., Feng, Y., ... & Chen, W. (2025). Agentcoord: Visually exploring coordination strategy for llm-based multi-agent collaboration. Computers & graphics, 104338.

[32] Zhang, Y., Yang, S., Bai, C., Wu, F., Li, X., Wang, Z., & Li, X. (2025, July). Towards efficient llm grounding for embodied multi-agent collaboration. In Findings of the Association for Computational Linguistics: ACL 2025 (pp. 1663-1699).

[33] Li, H., Chong, Y., Stepputtis, S., Campbell, J. P., Hughes, D., Lewis, C., & Sycara, K. (2023, December). Theory of mind for multi-agent collaboration via large language models. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (pp. 180-192).

[34] Xu, Y., Hu, J., Zhao, Z., Duan, Z., Sun, X., & Yang, X. (2025, November). MultiAgentESC: A LLM-based Multi-Agent Collaboration Framework for Emotional Support Conversation. In Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (pp. 4665-4681).

[35] Agashe, S., Fan, Y., Reyna, A., & Wang, X. E. (2025, April). Llm-coordination: evaluating and analyzing multi-agent coordination abilities in large language models. In Findings of the Association for Computational Linguistics: NAACL 2025 (pp. 8038-8057).