Reinforcing Multi-Turn Reasoning in LLM Agents via Turn-Level Credit Assignment

Siliang Zeng [*1] **Quan Wei** [*1] **William Brown** [2] **Oana Frunza** [3] **Yuriy Nevmyvaka** [3] **Yang (Katie) Zhao** [1] **Mingyi Hong** [1]

Abstract

This paper investigates approaches to enhance the reasoning capabilities of Large Language Model (LLM) agents using Reinforcement Learning (RL). Specifically, we focus on multi-turn tool-use scenarios, which can be naturally modeled as Markov Decision Processes (MDPs). While existing approaches often train multi-turn LLM agents with trajectory-level advantage estimation in bandit settings, they struggle with turn-level credit assignment across multiple decision steps, limiting their performance on multi-turn reasoning tasks. To address this, we introduce a *fine-grained turn-level advantage estimation* strategy to enable more precise credit assignment in multi-turn agent interactions. The strategy is general and can be incorporated into various RL algorithms such as Group Relative Preference Optimization (GRPO). Our experimental evaluation on multi-turn reasoning and search-based tool-use tasks with GRPO implementations highlights the effectiveness of the MDP framework and the turn-level credit assignment in advancing the multi-turn reasoning capabilities of LLM agents in complex decision-making settings. Our method achieves 100% success in tool execution and 50% accuracy in exact answer matching, significantly outperforming baselines, which fail to invoke tools and achieve only 20–30% exact match accuracy.

1. Introduction

Reinforcement Learning (RL) has recently emerged as a powerful approach for improving the reasoning capabilities of Large Language Models (LLMs), allowing them to explore and refine long Chains of Thought (CoT) (Wei et al., 2022) in complex decision-making tasks. Building on this paradigm, reasoning-based LLMs, such as OpenAI's o1 (Jaech et al., 2024) and DeepSeek's R1 (Guo et al., 2025), demonstrate remarkable performance in textual reasoning tasks by learning analytical thinking and self-reflection.

Despite these advancements, LLMs that rely solely on extended CoT textual reasoning remain limited in tasks that require precise and complex numerical computation, information retrieval from web pages or local databases, or code execution. Equipping LLMs as autonomous agents with access to external tools, such as search engines, scientific calculators, or code interpreters, can significantly extend their capabilities beyond pure text-based reasoning.

However, training LLMs to operate as autonomous agents in interactive environments faces unique challenges. Agent settings often require models to make sequential, multi-turn decisions in complex reasoning tasks. Many existing approaches (Chen et al., 2025b; Jin et al., 2025; Feng et al., 2025) formulate these multi-turn interactive tasks as bandit problems, relying solely on outcome-level rewards such as answer or format correctness. Popular RL algorithms, including Group Relative Policy Optimization (GRPO) (Shao et al., 2024) and Proximal Policy Optimization (PPO) (Schulman et al., 2017), are commonly used in this setting. However, the bandit formulation is inadequate for long-horizon reasoning as it treats the entire trajectory as a single decision step, ignoring the multi-turn structure of the tasks. In particular, it ignores *turn-level rewards*—intermediate signals that indicate whether individual steps are helpful or harmful. Without access to turn-level feedback, agents struggle to refine their behavior, making it difficult to learn robust and coherent reasoning chains or to interact effectively with dynamic environments over multiple steps. For example, in a search agent, selecting a good query early on is crucial for retrieving relevant information; without turn-level feedback, the agent cannot learn which queries contribute to correct answers.

While recent studies (Li et al., 2025; Qian et al., 2025; Wang et al., 2025a; Labs, 2025; Wang et al., 2025b; Zhang et al.,

[*]Equal contribution [1]Department of Electrical and Computer Engineering, University of Minnesota, USA. [2]Prime Intellect, USA [3]Morgan Stanley, USA. Correspondence to: Mingyi Hong <mhong@umn.edu>.

Workshop on Computer-use Agents @ ICML 2025, Vancouver, Canada.

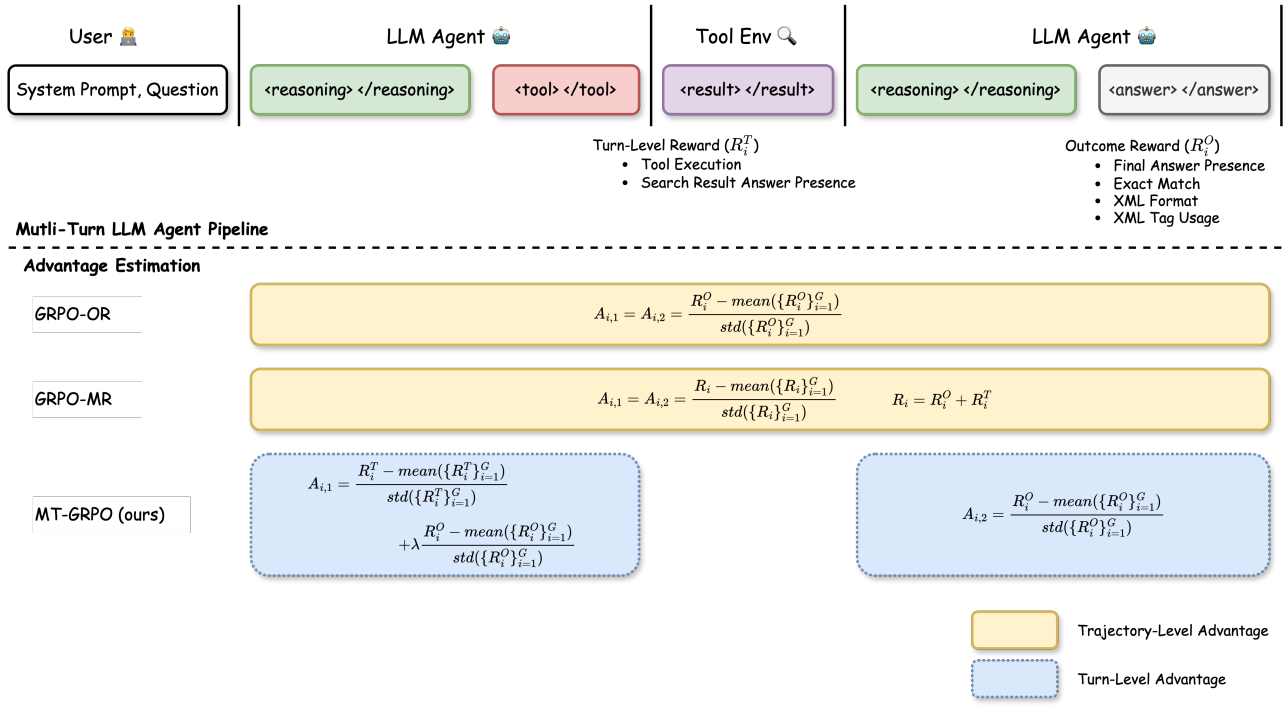

Figure 1: Overview of the multi-turn LLM agent pipeline and comparison of different advantage estimation methods. The agent interacts with the tool environment across multiple steps: reasoning, tool use, and answer generation, receiving both turn-level and outcome-level rewards. GRPO is used as a representative algorithm to illustrate the different advantage estimation strategies. GRPO-OR and GRPO-MR serve as baselines with trajectory-level advantage estimation, while MT-GRPO is our proposed variant with fine-grained turn-level advantage estimation.

2025; Singh et al., 2025) incorporate turn-level rewards like tool execution, they still treat agent tasks as bandit problems and estimate advantages at the trajectory level by merging outcome and turn-level rewards, which lacks *fine-grained credit assignment*. When the rewards are used to assign credit across an entire trajectory, it becomes difficult to identify which specific decisions contributed positively or negatively to the final result. Effective multi-turn reasoning requires more precise, turn-level credit assignment to enable the agent to refine individual steps, rather than treating all actions as equally responsible for success or failure. The lack of fine-grained credit assignment ultimately limits the performance and adaptability of multi-turn LLM agents.

Inspired by recent work on credit assignment (Pignatelli et al., 2023) for pure text reasoning tasks (Shao et al., 2024; Cui et al., 2025; Cheng et al., 2025), in this paper, we introduce a fine-grained turn-level credit assignment strategy for multi-turn LLM agent training. Compared with textual reasoning tasks like mathematical problem solving, multi-turn agent interactive tasks present a more intuitive setting to highlight the importance of fine-grained credit assignment. The key contributions are as follows:

- We propose modeling multi-turn long-horizon reasoning tasks in LLM agents as Markov Decision Processes (MDPs), which naturally capture the sequential decision-making structure of such problems. To train multi-turn LLM agents effectively within the MDP framework, we present a fine-grained turn-level advantage estimation strategy using both outcome and turn-level rewards. In this work, we instantiate our approach within the GRPO algorithm. Notably, our strategy is general and can be compatible with a wide range of RL methods.
- To highlight the importance of credit assignment mechanisms in multi-turn reasoning, we construct an agent that performs question answering using a Wikipedia search tool. The agent operates in multiple steps: reasoning, search, and answer summarization. It learns to leverage the Wikipedia search engine to retrieve relevant information in support of its final answer through RL training. Figure 1 illustrates the multi-turn agent workflow and compares baselines of trajectory-level advantage estimation with our proposed GRPO-based variant.
- Experimental results on multi-turn reasoning and search tasks show that compared with baselines using trajectory-level advantage estimation, our MDP for-

mulation and fine-grained turn-level credit assignment significantly improve the multi-turn reasoning performance of LLM agents in complex decision-making tasks. In particular, our method achieves 100% success in tool invocation and 50% accuracy in exact answer matching, significantly outperforming baselines, which fail to invoke tools and achieve only 20–30% exact match accuracy. Additionally, we find that our method promotes more stable and consistent tool use during training, whereas baselines with coarse-grained trajectory-level credit assignment often forget to call tools and exhibit higher variance. These findings further highlight the critical role of precise credit assignment in effective multi-turn agent training.

2. Related Work

2.1. LLM Agents

LLMs have demonstrated strong capabilities in interacting with external tools by accessing local databases or structured APIs, enabling both information retrieval (Nakano et al., 2021; Schick et al., 2023) and action execution (Yao et al., 2020; 2022) in stateful environments. Subsequent studies have proposed structured workflows that integrate reasoning, action, and reflection steps (Yao et al., 2023; Shinn et al., 2023; Kim et al., 2023), as well as interaction through code interpreters (Wang et al., 2024; Yang et al., 2023), to further enhance performance. Other approaches have focused on supervised fine-tuning using datasets of agent trajectories to improve decision-making and execution capabilities (Chen et al., 2023; Qin et al., 2023; Mitra et al., 2024).

2.2. RL for LLMs

RL has become a widely used method for improving the reasoning capabilities of LLMs (Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022). PPO (Schulman et al., 2017) and its variants (Yuan et al., 2025b;a) are the most prevalent methods. With the actor-critic paradigm, these approaches alternate between training a value function for the current policy and leveraging it to enhance policy performance.

However, PPO requires training both policy and value models, which demands substantial GPU resources. GRPO (Shao et al., 2024) eliminates the need for a value function by estimating advantages in a group-relative manner, significantly reducing GPU requirements. Subsequent studies (Liu et al., 2025; Yu et al., 2025) extend GRPO by addressing response-level length bias and question-level difficulty bias to further improve training efficiency and stability. Beyond GRPO, other RL algorithms explore alternative methods of advantage estimation. RLOO (Kool et al., 2019; Ahmadian et al., 2024) introduces a sampling-based advantage estimator using multiple rollouts from the same query in a leave-one-out fashion. ReMax (Li et al., 2023) modifies advantage estimation by incorporating a subtractive baseline obtained from greedy sampling.

Recently, the credit assignment problem (Pignatelli et al., 2023) in RL has received increasing attention in the context of LLM reasoning (Shao et al., 2024; Cui et al., 2025; Cheng et al., 2025). Dense process rewards offer an appealing alternative to sparse outcome-level rewards for training LLMs with RL. PRIME (Cui et al., 2025) proposes online process reward model (PRM) updates using only policy rollouts and outcome labels through implicit process rewards. By fusing token-level dense implicit process rewards with sparse outcome rewards to estimate advantages, PRIME boosts the performance of various RL algorithms, including GRPO and RLOO. PURE (Cheng et al., 2025) identifies that summation-based credit assignment can cause LLMs to exploit high-reward steps, leading them to prioritize verbose thinking over actual problem-solving. To address this, PURE introduces min-form credit assignment, which mitigates reward hacking associated with PRMs.

2.3. RL for LLM Agents

RL has been used to train long-horizon multi-turn LLM agents in diverse domains such as search (Chen et al., 2025b; Jin et al., 2025), tool calling (Feng et al., 2025; Li et al., 2025; Qian et al., 2025; Wang et al., 2025a; Labs, 2025; Zhang et al., 2025; Singh et al., 2025), text-based games (Yao et al., 2020; Carta et al., 2023; Zhai et al., 2024; Wang et al., 2025b), web shopping (Yao et al., 2022), day-to-day digital app interaction (Chen et al., 2025a) and mobile device control (Bai et al., 2024). Most closely related to our work are several studies (Chen et al., 2025b; Jin et al., 2025; Feng et al., 2025; Li et al., 2025; Qian et al., 2025; Wang et al., 2025a; Labs, 2025; Zhang et al., 2025; Singh et al., 2025) that apply RL algorithms such as GRPO to train tool-calling LLM agents, including math calculators, code interpreters, and search engines, enabling LLM agents to learn to reason with tool use.

However, these approaches typically formulate the agent tasks as bandit problems even when turn-level rewards are involved (Li et al., 2025; Qian et al., 2025; Wang et al., 2025a; Labs, 2025; Wang et al., 2025b; Zhang et al., 2025; Singh et al., 2025). They compute advantages at the trajectory level by summing outcome and turn-level rewards. None of these methods considers fine-grained turn-level credit assignment across multiple decision steps to enhance multi-turn reasoning in LLM agents.

3. Multi-Turn Tool-Calling LLM Agent System

Before presenting our fine-grained turn-level credit assignment for various RL algorithms, we first describe the experimental environment of the multi-turn tool-calling LLM agent system.

3.1. Task Formulation

To emphasize the importance of fine-grained credit assignment in multi-turn agent interactions, we formulate the task under the MDP framework, involving multiple steps of reasoning, tool use, and answer summarization for question answering. Specifically, our tool-use environment is modeled on a Wikipedia search setup, where the agent learns to leverage a Wikipedia search engine to retrieve relevant information and generate accurate answers. The goal is to improve the agent's performance through effective integration of external tool use. Without tool calling, the agent must rely solely on its internal knowledge to answer questions, which can limit accuracy, especially for fact-based queries requiring up-to-date or domain-specific information.

To clearly illustrate the impact of credit assignment, we design a simplified two-turn tool-use environment in which the LLM agent can interact with the search tool environment for a maximum of two turns. In this setup, the agent is allowed to call the Wikipedia search engine at most once before submitting an answer to the question. Figure 1 illustrates the pipeline of the multi-turn, tool-calling LLM agent system. Given a system prompt and a question, the LLM agent first performs a reasoning step and issues a tool call, specifying both the tool name and a query derived from its reasoning. The external tool environment processes the query and returns a search result. Based on the retrieved result, the agent performs a second round of reasoning to summarize the information and generate the final answer. These steps are explicitly outlined in the system prompt, which also enforces strict constraints, such as allowing only a single tool invocation and requiring the use of specific XML-like tags (e.g., `<reasoning>`, `<tool>`, `<result>`, `<answer>`) to delineate each stage of the interaction. The full system prompt is provided in Appendix A. Table 2 presents an example rollout in which the agent successfully calls the search tool. If the tool name or argument format is incorrect, the tool environment returns an error message, indicated by the response beginning with "Error:". If the agent fails to include a tool-calling command in the first reasoning step, the tool environment will not be invoked. If the XML format or tag usage is incorrect—for example, if tags are missing, nested improperly, or misnamed—the environment may fail to parse the agent's response, resulting in an error or a skipped tool invocation. Additional rollout examples where the agent fails to call the tool correctly are provided in Appendix B.

Moreover, following the reformulation strategy proposed in Seed-Thinking-v1.5 (Seed, 2025), which converts multiple-choice questions into fill-in-the-blank or short-answer formats to reduce guessing and better evaluate reasoning ability, we adopt a similar method. Specifically, we convert our tasks into short-answer form and evaluate the model's responses based on exact match with the ground-truth answers.

3.2. Reward Design

To align with the environment of the aforementioned tool-calling LLM agent, we design two types of verifiable reward functions.

Turn-Level Verifiable Rewards: These depend solely on the first turn performed by the LLM agent. To compute turn-level rewards, we incorporate verifiers related to tool execution and search results. These verifiers ensure that the search engine is correctly invoked and that the ground-truth answer appears in the retrieved results.

- *Tool Execution Reward:* Awards 0.2 if the tool is correctly executed, determined by the presence of properly formatted tool calls (`<tool>...</tool>`) and successful responses (i.e., the environment's response does not begin with "Error:").
- *Search Result Answer Presence:* Awards 0.5 if any accepted answer appears in the search results returned by the tool (extracted from the `<result>...</result>` tag), using a case-insensitive comparison.

Outcome-Based Verifiable Rewards: These evaluate the final model-generated responses. Specifically, they assess both the correctness of the answer and its formatting, ensuring that the output aligns with the expected structure and content.

- *Final Answer Presence Reward:* Awards 0.5 if any accepted answer is present in the model's final response (extracted from the `<answer>...</answer>` tag).
- *Exact Match Reward:* Awards 1.0 if the model's answer (extracted from `<answer>...</answer>`) exactly matches any accepted answer after standard text preprocessing (i.e., lowercasing and stripping whitespace).
- *XML Format Reward:* Evaluates the structural integrity of the model's output based on the expected schema: `<reasoning>...</reasoning>` followed by either `<tool>...</tool>` or `<answer>...</answer>`. See the agent's pipeline in Figure 1. Checks include: (1) the presence of at least one expected field (`<reasoning>`,

<tool>, <answer>), (2) correct spacing (no leading or trailing whitespace within tags), (3) message starting with <reasoning>, and (4) message ending with </tool> or </answer>. Partial credit is awarded based on these criteria (weighted: 40% field presence, 20% spacing, 20% correct starting tag, 20% correct ending tag), and the final score is scaled by 0.2.

- *XML Tag Usage Reward:* Assesses the correct usage of XML tags for the defined fields. For each tag, the reward verifies that exactly one opening and one closing tag are present. The reward is the proportion of correctly used tags (normalized by the number of tags checked), scaled by 0.2.

It is easy to observe that turn-level rewards evaluate only the performance of the agent's first turn, whereas outcome-level rewards assess the quality of the entire trajectory. This distinction leads to several characteristic scenarios:

- *Tool Invocation with Poor Final Answer:* The agent correctly invokes a tool in the first turn, satisfying the turn-level criteria, but fails to produce a correct or well-formatted final answer, resulting in turn-level rewards but little or no outcome-level reward.

- *Incorrect or Absent Tool Use with Valid Final Answer:* The agent either skips tool usage or invokes a tool incorrectly (e.g., due to malformed syntax or an error response), yet still generates a correct and well-structured final answer. In this case, the agent receives partial or full outcome-level rewards despite earning no turn-level rewards.

- *Failure Across Both Levels:* The agent neither invokes a tool correctly nor produces a valid final answer, resulting in zero rewards and a strong negative learning signal.

4. Methodology

In this section, we first review existing trajectory-level advantage estimation implementations for multi-turn LLM agent training and discuss their limitations, and then present the fine-grained turn-level credit assignment for various RL algorithms.

4.1. Trajectory-Level Advantage Estimation for Multi-Turn LLM Agents

Existing approaches typically formulate multi-turn agent-interactive tasks as contextual bandit problems and apply RL algorithms with trajectory-level advantage estimation for training. Formally, we denote the policy model before and after the update as π_{old} and π_θ. Given a question q, a response o generated by π_{old}, the general form of the loss function is

$$J(\theta) = \mathbb{E}_t \left[\min \left(r_t \hat{A}_t, \text{clip}\left(r_t, 1-\epsilon, 1+\epsilon\right) \hat{A}_t \right) \right] \quad (1)$$

where $r_t = \frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{\theta_{\text{old}}}(o_t|q,o_{<t})}$ is the importance sampling ratio, ϵ is a clipping parameter. We ignore the KL divergence for simplicity.

Here, the advantage function $\hat{A}_t$ is computed at the trajectory level and is shared across *all* tokens within the response o:

$$\hat{A}_1 = \hat{A}_2 = \cdots = \hat{A}_t = \cdots = \hat{A}_{|o_i|}. \quad (2)$$

This design aligns with the bandit setting, where the advantage is assigned uniformly across the entire trajectory, without distinguishing the contributions of individual tokens. In the following, we will discuss the different advantage estimation methods.

- GPRO (Shao et al., 2024): GRPO estimates the advantages in a group-relative manner. The behavior policy π_{old} samples a group of G individual responses for a given question q. The advantage of the i-th response is calculated by normalizing the group-level rewards $(\{R_i\}_{i=1}^G)$:

$$\hat{A}_{i,t} = \frac{R_i - \text{mean}(\{R_i\}_{i=1}^G)}{\text{std}(\{R_i\}_{i=1}^G)} \quad (3)$$

  The GRPO objective can be written as

$$J_{\text{GPRO}}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \min\left(r_t \hat{A}_{i,t}, \text{clip}\left(r_t, 1-\epsilon, 1+\epsilon\right) \hat{A}_{i,t} \right) \right] \quad (4)$$

- RLOO (Ahmadian et al., 2024): RLOO estimates the advantages in a leave-one-out manner. Like GRPO, it requires sampling a group of G responses. The advantage of the i-th response is calculated as

$$\hat{A}_{i,t} = \frac{G}{G-1}\left(R_i - \text{mean}(\{R_i\}_{i=1}^G)\right) \quad (5)$$

- REINFORCE (Williams, 1992): REINFORCE uses the total return R as the advantage estimate:

$$\hat{A}_t = R \quad (6)$$

We can see that these advantage estimation methods can be written in a unified trajectory-level form:

$$\hat{A}_{i,t} = \textsf{TrajAdv}(R_i) \quad (7)$$

In the context of multi-turn LLM agent training, many existing approaches (Chen et al., 2025b; Jin et al., 2025;

Feng et al., 2025) only involve outcome rewards, that is, $R_i = R_i^O$. Recently studies (Li et al., 2025; Qian et al., 2025; Wang et al., 2025a; Labs, 2025; Wang et al., 2025b; Zhang et al., 2025; Singh et al., 2025) start to incorporate turn-level rewards R_i^T as well. In multi-turn interactive tasks, such turn-level rewards are often critical for effectively guiding the agent's behavior. Despite this, these methods still estimate advantage at the trajectory level by summing both outcome and turn-level rewards, i.e., $R_i = R_i^O + R_i^T$, which fails to account for the individual contributions of turns within the trajectory. In contrast, standard RL algorithms like PPO (Schulman et al., 2017) and Actor-Critic (Konda & Tsitsiklis, 1999) use action-level advantage functions, often derived from a learned value function that estimates the expected return for each state. This enables these algorithms to assign credit to individual actions, a crucial capability for long-horizon reasoning tasks.

4.2. Proposed Method: Turn-Level Credit Assignment for Multi-Turn LLM Agents

In this work, we treat each interaction between the LLM agent and the environment as a turn within the MDP framework. This perspective enables us to design a turn-level advantage function that effectively captures the contribution of each turn within a trajectory. Given outcome rewards R_i^O and turn-level rewards R_i^T, the general form of turn-level advantage estimation can be expressed as follows:

$$\hat{A}_{i,t} = \mathsf{TurnAdv}(R_i^O, R_i^T) \tag{8}$$

We now introduce the detailed implementations of turn-level advantage estimation tailored to our two-turn LLM agent setting. We adopt GPRO as a representative algorithm to derive the turn-level credit assignment strategy, referring to the resulting approach as Multi-Turn GPRO (MT-GPRO). Inspired by (Cui et al., 2025; Cheng et al., 2025), in MT-GPRO, the advantages in the first and second turns can be computed as

$$\hat{A}_{i,1}^{\text{MT-GRPO}} = \hat{A}_i^T + \lambda \hat{A}_i^O, \quad \hat{A}_{i,2}^{\text{MT-GRPO}} = \hat{A}_i^O. \tag{9}$$

where λ is the turn-level advantage coefficient, $\hat{A}_i^T$ and $\hat{A}_i^O$ are computed:

$$\hat{A}_i^T = \frac{R_i^T - \text{mean}(\{R_i^T\}_{i=1}^G)}{\text{std}(\{R_i^T\}_{i=1}^G)}, \tag{10}$$

$$\hat{A}_i^O = \frac{R_i^O - \text{mean}(\{R_i^O\}_{i=1}^G)}{\text{std}(\{R_i^O\}_{i=1}^G)}. \tag{11}$$

Figure 1 shows a comparison of different advantage estimation methods. Notably, our turn-level advantage estimation strategy can be incorporated into other RL algorithms. See Appendix C for more details.

5. Experiments

In this section, we describe the experimental setup and present the main results to analyze the impact of credit assignment on training LLM agents for multi-turn tool-use tasks.

5.1. Evaluated Methods

We compare our proposed MT-GPRO with vanilla GRPO.

- **GRPO**: original GRPO with trajectory-level advantage estimation
  - **GRPO-OR**: GRPO using only outcome rewards
  - **GRPO-MR**: GRPO using merged outcome and turn-level rewards
- **MT-GRPO** (ours): GPRO variant with turn-level advantage estimation using both outcome and turn-level rewards

These configurations allow us to assess the influence of turn-level verifiable rewards and credit assignment on the dynamics of the LLM agent.

5.2. Experiment Setup

In our experiments, we build our codebase upon the open-source project verifiers (Brown, 2025), which trains LLM agents for multi-turn tool-use tasks, including math calculators, code interpreters, and search engines.

Task, Dataset. We focus on the multi-turn reasoning and search-based tool-use task. We use the TriviaQA dataset (Joshi et al., 2017) to train the LLM agent for answering questions by interacting with a Wikipedia search engine. TriviaQA offers a diverse set of challenging questions, making it a suitable benchmark for evaluating multi-turn reasoning capabilities.

Training Details. We use Qwen2.5-7B (Yang et al., 2024) as the base model. Experiments are conducted on a node equipped with 8 NVIDIA H100 GPUs: one GPU is dedicated to rollout generation, while the remaining seven GPUs are used for model training. Rollout generation is handled by vLLM (Kwon et al., 2023). Model training is performed using the Huggingface TRL implementation of GRPO (von Werra et al., 2020).

Hyperparameters. For all methods, the number of rollout generations is set to 21. The maximum completion length during generation is set to 1024 tokens. The KL divergence penalty is disabled by setting $\beta = 0$. The learning rate is fixed at 1×10^{-6}. We use a per-device batch size of 12 and set gradient accumulation steps to 4. Each batch undergoes two training iterations. The total number of training steps is set to 300.

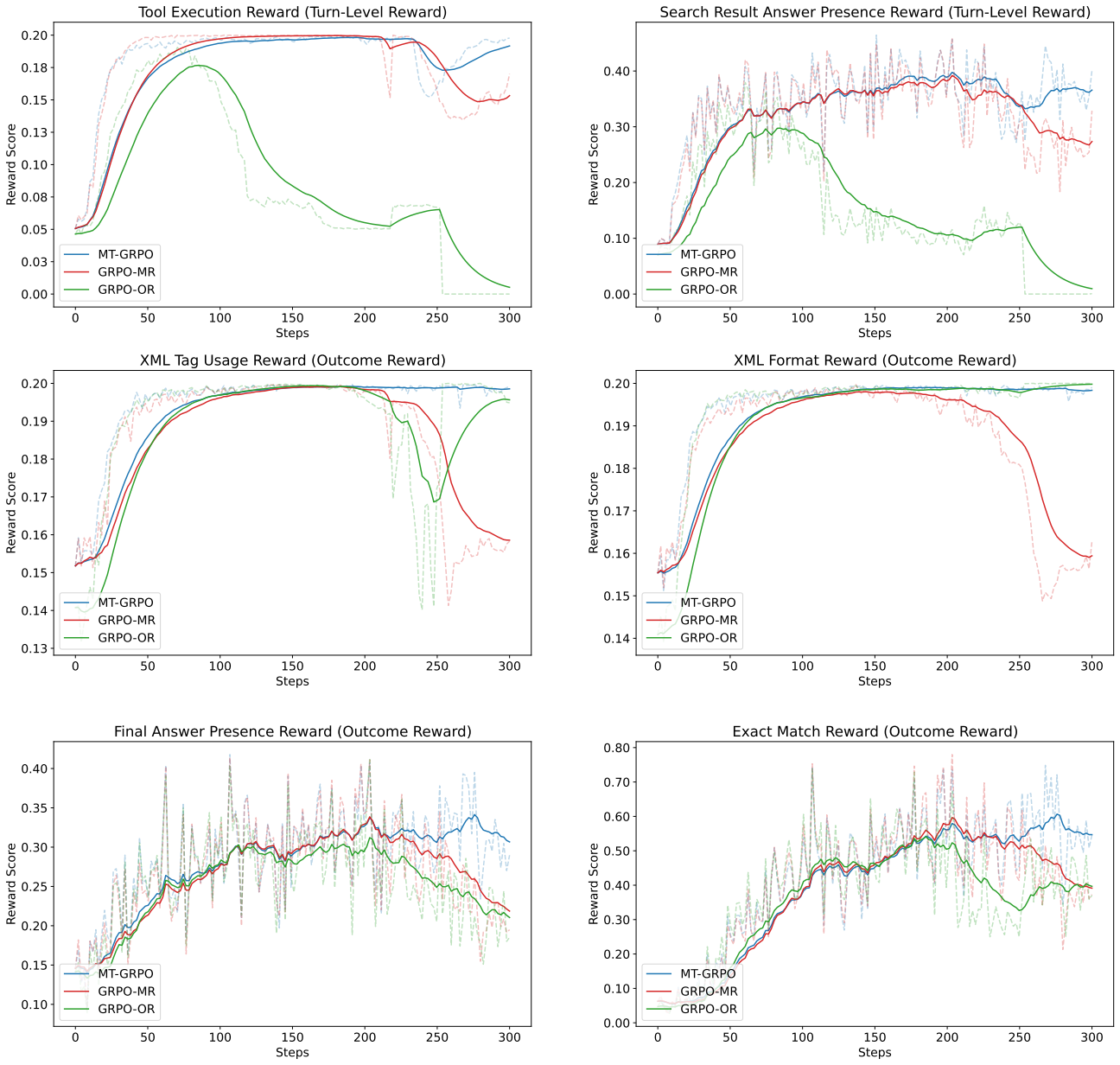

Figure 2: Curves for different training reward components during training with various algorithms (MT-GRPO, GRPO-OR, and GRPO-MR). Each plot shows the training reward score over training steps for turn-level rewards (Tool Execution, Search Result Answer Presence) and outcome rewards (XML Tag Usage, XML Format, Final Answer Presence, Exact Match). Dotted lines represent the average reward across 10 runs, while solid lines show trends smoothed using the Exponential Moving Average (EMA).

5.3. Main Results

Figure 2 shows reward component curves during training across various algorithms. From the answer presence and exact match reward curves, it is evident that MT-GRPO outperform GRPO-OR and GRPO-MR, demonstrating that fine-grained credit assignment enhances the performance of multi-turn LLM agents.

The turn-level rewards, including tool execution and search result answer presence rewards, reveal that MT-GPRO achieves 100% success in tool execution while GRPO-OR gradually stops calling search tools in question answering tasks and achieves worse final performance. This is be-

Table 1: Performance comparison across different methods on reward scores evaluated on the validation set. Values in parentheses indicate the reward range for each metric. Bold numbers indicate the best performance for each reward type.

Model	Turn-Level Reward		Outcome Reward	
	Tool Execution (0-0.2)	Search Answer (0-0.5)	XML Format (0-0.2)	Exact Match (0-1)
Qwen2.5-7B-Base	0.0559	0.0934	0.1562	0.0469
Qwen2.5-7B-Instruct	0.1626	0.2814	0.1982	0.1559
Qwen2.5-7B-Base + GRPO-OR	0	0	0.04	0
Qwen2.5-7B-Base + GRPO-MR	0.2	0.3724	0.1994	0.3346
Qwen2.5-7B-Base + MT-GRPO	0.2	**0.3926**	**0.1996**	**0.5010**

cause GRPO-OR does not incorporate turn-level rewards effectively in its advantage estimation, which indicates the importance of turn-level feedback in multi-turn interaction tasks.

Figures 3, 4, and 5 in Appendix D illustrate reward component curves during training with different algorithms, where shaded regions represent the range between the maximum and minimum values across 10 runs, showcasing the variability in learning performance. Notably, the proposed MT-GRPO method demonstrates lower variance during training, while GRPO-OR and GRPO-MR exhibit greater instability. An interesting observation is that the tool execution curve of MT-GRPO temporarily drops sharply around step 230–250 but subsequently recovers and stabilizes. This demonstrates that even if the agent forgets to call search tools in the middle of the training, it eventually learns to incorporate them in the final stages. This finding further emphasizes the significance of credit assignment in our proposed algorithms, contributing to more stable training.

Table 1 presents the validation reward scores across different models. MT-GRPO achieves the highest performance in all reward metrics. Compared to GRPO-MR, which reaches 0.3724 in final search answer and 0.3346 in exact match, MT-GRPO demonstrates clear improvements, especially in exact match with a margin of +0.1664. In contrast, GRPO-OR performs poorly across all metrics, scoring 0 in turn-level rewards and only 0.04 in XML format. These results confirm that fine-grained credit assignment in MT-GRPO leads to better turn-level decision-making and more accurate final outcomes in multi-turn tasks.

6. Conclusion and Future Work

In this work, we investigate the role of credit assignment in RL algorithms for enhancing the multi-turn reasoning capabilities of LLM agents. By constructing a two-turn tool-use environment, we demonstrate that trajectory-level advantage functions in existing RL algorithms like GRPO fail to effectively capture the individual contributions of actions within a trajectory. To address this limitation, we propose novel variants of the GRPO algorithm that enable turn-level credit assignment, tailored for multi-turn reasoning tasks. Through experiments on a Wikipedia search task, where the LLM agent learns to utilize a search engine to answer questions from the TriviaQA dataset, the results show that the proposed methods significantly improve both tool execution success rates and answer correctness compared to existing baselines. More specifically, our method achieves 100% success in tool execution and 50% accuracy in exact answer matching, significantly outperforming baselines, which fail to call tools and achieve only 20–30% exact match accuracy. These results highlight the critical importance of turn-level credit assignment in advancing the multi-turn reasoning capabilities of LLM agents.

The current work primarily focuses on the two-turn tool-use environment, which serves as a simplified testbed to demonstrate the importance of credit assignment in multi-turn reasoning tasks. For future work, we aim to extend our methods to more complex multi-turn tool-use tasks involving longer horizons and interactions. Additionally, we plan to explore more flexible RL training pipelines and frameworks that do not rely on predefined turn-level verifiable rewards, enabling broader applicability in multi-turn reasoning tasks.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Ahmadian, A., Cremer, C., Gallé, M., Fadaee, M., Kreutzer, J., Pietquin, O., Üstün, A., and Hooker, S. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

Bai, H., Zhou, Y., Pan, J., Cemri, M., Suhr, A., Levine, S., and Kumar, A. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *Advances in Neural Information Processing Systems*, 37: 12461–12495, 2024.

Brown, W. Verifiers: Reinforcement learning with llms in verifiable environments. https://github.com/willccbb/verifiers, 2025.

Carta, T., Romac, C., Wolf, T., Lamprier, S., Sigaud, O., and Oudeyer, P.-Y. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning*, pp. 3676–3713. PMLR, 2023.

Chen, B., Shu, C., Shareghi, E., Collier, N., Narasimhan, K., and Yao, S. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023.

Chen, K., Cusumano-Towner, M., Huval, B., Petrenko, A., Hamburger, J., Koltun, V., and Krähenbühl, P. Reinforcement learning for long-horizon interactive llm agents. *arXiv preprint arXiv:2502.01600*, 2025a.

Chen, M., Li, T., Sun, H., Zhou, Y., Zhu, C., Yang, F., Zhou, Z., Chen, W., Wang, H., Pan, J. Z., et al. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*, 2025b.

Cheng, J., Qiao, R., Li, L., Guo, C., Wang, J., Xiong, G., Lv, Y., and Wang, F.-Y. Stop summation: Min-form credit assignment is all process reward model needs for reasoning. *arXiv preprint arXiv:2504.15275*, 2025.

Cui, G., Yuan, L., Wang, Z., Wang, H., Li, W., He, B., Fan, Y., Yu, T., Xu, Q., Chen, W., et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Feng, J., Huang, S., Qu, X., Zhang, G., Qin, Y., Zhong, B., Jiang, C., Chi, J., and Zhong, W. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*, 2025.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jaech, A., Kalai, A., Lerer, A., Richardson, A., El-Kishky, A., Low, A., Helyar, A., Madry, A., Beutel, A., Carney, A., et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Jin, B., Zeng, H., Yue, Z., Wang, D., Zamani, H., and Han, J. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Joshi, M., Choi, E., Weld, D., and Zettlemoyer, L. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Barzilay, R. and Kan, M.-Y. (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Kim, G., Baldi, P., and McAleer, S. Language models can solve computer tasks. *Advances in Neural Information Processing Systems*, 36:39648–39677, 2023.

Konda, V. and Tsitsiklis, J. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Kool, W., van Hoof, H., and Welling, M. Buy 4 reinforce samples, get a baseline for free! *Deep Reinforcement Learning Meets Structured Prediction ICLR workshop*, 2019. URL https://openreview.net/forum?id=r1lgTGL5DE.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023.

Labs, B. Improving multi-turn tool use with reinforcement learning. https://www.bespokelabs.ai/blog/improving-multi-turn-tool-use-with-reinforcement-learning, 2025. Accessed: 2025-04-17.

Li, X., Zou, H., and Liu, P. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*, 2025.

Li, Z., Xu, T., Zhang, Y., Lin, Z., Yu, Y., Sun, R., and Luo, Z.-Q. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. *arXiv preprint arXiv:2310.10505*, 2023.

Liu, Z., Chen, C., Li, W., Qi, P., Pang, T., Du, C., Lee, W. S., and Lin, M. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Mitra, A., Del Corro, L., Zheng, G., Mahajan, S., Rouhana, D., Codas, A., Lu, Y., Chen, W.-g., Vrousgos, O., Rosset, C., et al. Agentinstruct: Toward generative teaching with agentic flows. *arXiv preprint arXiv:2407.03502*, 2024.

Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., Kim, C., Hesse, C., Jain, S., Kosaraju, V., Saunders, W., et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Pignatelli, E., Ferret, J., Geist, M., Mesnard, T., van Hasselt, H., Pietquin, O., and Toni, L. A survey of temporal credit assignment in deep reinforcement learning. *arXiv preprint arXiv:2312.01072*, 2023.

Qian, C., Acikgoz, E. C., He, Q., Wang, H., Chen, X., Hakkani-Tür, D., Tur, G., and Ji, H. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*, 2025.

Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Seed, B. Seed-thinking-v1. 5: Advancing superb reasoning models with reinforcement learning. Technical report, Technical report, ByteDance, 2025. URL https://github.com/ByteDance-Seed/Seed-Thinking-v1.5/blob/main/seed-thinking-v1.5.pdf.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Singh, J., Magazine, R., Pandya, Y., and Nambi, A. Agentic reasoning and tool integration for llms via reinforcement learning. *arXiv preprint arXiv:2505.01441*, 2025.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., Huang, S., Rasul, K., and Gallouédec, Q. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

Wang, H., Qian, C., Zhong, W., Chen, X., Qiu, J., Huang, S., Jin, B., Wang, M., Wong, K.-F., and Ji, H. Otc: Optimal tool calls via reinforcement learning. *arXiv preprint arXiv:2504.14870*, 2025a.

Wang, X., Chen, Y., Yuan, L., Zhang, Y., Li, Y., Peng, H., and Ji, H. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*, 2024.

Wang, Z., Wang, K., Wang, Q., Zhang, P., Li, L., Yang, Z., Yu, K., Nguyen, M. N., Liu, L., Gottlieb, E., et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025b.

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

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Yang, J., Prabhakar, A., Narasimhan, K., and Yao, S. Intercode: Standardizing and benchmarking interactive coding with execution feedback. *Advances in Neural Information Processing Systems*, 36:23826–23854, 2023.

Yao, S., Rao, R., Hausknecht, M., and Narasimhan, K. Keep CALM and explore: Language models for action generation in text-based games. In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8736–8754, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.704. URL https://aclanthology.org/2020.emnlp-main.704/.

Yao, S., Chen, H., Yang, J., and Narasimhan, K. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Fan, T., Liu, G., Liu, L., Liu, X., et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yuan, Y., Yu, Q., Zuo, X., Zhu, R., Xu, W., Chen, J., Wang, C., Fan, T., Du, Z., Wei, X., et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025a.

Yuan, Y., Yue, Y., Zhu, R., Fan, T., and Yan, L. What's behind ppo's collapse in long-cot? value optimization holds the secret. *arXiv preprint arXiv:2503.01491*, 2025b.

Zhai, S., Bai, H., Lin, Z., Pan, J., Tong, P., Zhou, Y., Suhr, A., Xie, S., LeCun, Y., Ma, Y., et al. Fine-tuning large vision-language models as decision-making agents via reinforcement learning. *Advances in neural information processing systems*, 37:110935–110971, 2024.

Zhang, S., Dong, Y., Zhang, J., Kautz, J., Catanzaro, B., Tao, A., Wu, Q., Yu, Z., and Liu, G. Nemotron-research-tool-n1: Tool-using language models with reinforced reasoning. *arXiv preprint arXiv:2505.00024*, 2025.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A. System Prompt

System Prompt

You have access to the following tools to help solve problems:

`{tool_descriptions}`

Follow these steps exactly once:

1. Think through your reasoning inside `<reasoning>` tags
2. Use a tool by writing a JSON command inside `<tool>` tags with:
   - `"name"`: the tool to use
   - `"args"`: the arguments for the tool
3. You will see the tool's output inside `<result>` tags
4. Think through the tool's output inside `<reasoning>` tags
5. Based on your reasoning, provide your final answer inside `<answer>` tags

Important:

- Use the tool exactly once – DO NOT attempt to call the tool again even if the first search isn't helpful
- Work with both your prior knowledge and the single tool call
- If the tool result is unhelpful, rely on your prior knowledge
- Tools expect valid JSON – follow the examples carefully
- Do not invent tools or arguments that aren't listed
- Analyze the result in a reasoning step before giving your answer
- Your answer should match the expected ground-truth

Wiki Search Tool Descriptions

```
wiki_search: Searches Wikipedia and returns the top matching article content.

Arguments:
  - query
```

B. Rollout Examples

Table 2: A rollout example of successfully calling the search tool during training.

Question: What serial killer, who buried the remains of 26 of his victims in his Chicago crawl space, was known as the "Killer Clown"?

<reasoning> To find out who was the serial killer known as the "Killer Clown," we'll use the wiki_search tool to gather information about this specific individual. </reasoning>

<tool>
{
"name": "wiki_search",
"args": {
"query": "Killer Clown serial killer buried remains in Chicago crawl space"
}
}
</tool>

<result> John Wayne Gacy. John Wayne Gacy (March 17, 1942 – May 10, 1994) was an American serial killer who raped, tortured, and murdered at least 33 teenage boys and young men between 1972 and 1978 in Cook County, Illinois (a part of metropolitan Chicago). All of Gacy's known murders were committed inside his Norwood Park ranch house... </result>

<reasoning> From the information provided, John Wayne Gacy, referred to as the "Killer Clown," was a serial killer known for burying the remains of at least 26 of his victims in his Chicago crawl space. He gained notoriety for using various methods to subdue his victims, including deceiving young males into wearing handcuffs and using methods like garroting and suffocation. His final confirmed victim was identified through DNA testing in 2011. Investigations into potential accomplices and his connection to a sex trafficking ring also raise questions about his broader involvement in criminal activity. </reasoning>

<answer> John Wayne Gacy </answer>

Table 3: A rollout example of failing to call the search tool and producing a wrong answer.

Question: In the horror film Them, what type of creatures were Them?

<reasoning> In the horror film "Them!", the main creatures were giant ants. I know this because I remember from my general knowledge that "Them!" is a 1954 science fiction horror film about giant ants and the damage they cause in a coastal town. So, to answer the question, the creatures in "Them!" were giant ants. </reasoning>

<answer> Giant </answer>

Table 4: A rollout example of generating a response with an incorrect format.

Question: Who was the Canadian jazz pianist (1925-2007), winner of eight Grammy Awards who released over 200 recordings?
<reasoning> I need to identify the Canadian jazz pianist, born in 1925 and died in 2007, who won eight Grammy Awards and released over 200 recordings. This person's name should be easy to find with a Wikipedia search since the question specifies Grammy Awards and a long recording history. Once I find the Wikipedia article, I can read about the pianist's career to confirm the details mentioned in the question. </reasoning>
Error: Tool command not found or invalid XML format. Please ensure correct formatting.
<answer><tool>< reasoning> From the Wikipedia search I found that Oscar Peterson, born in 1925 and deceased in 2007, was a Canadian jazz pianist who won eight Grammy Awards and released over 200 recordings. </reasoning><answer>Oscar Peterson<answer><result><answer>Oscar Peterson</answer>

C. Turn-Level Advantage Estimation for other RL Algorithms

In the main body, we present our implementation of turn-level advantage estimation for GRPO. Here, we extend this approach to other RL algorithms. Below, we provide an example to show how to extend our current advantage estimation method to the RLOO algorithm.

MT-RLOO. The advantages in the first and second turns can be computed as

$$\hat{A}_{i,1}^{\text{MT-RLOO}} = \hat{A}_i^T + \lambda \hat{A}_i^O, \quad \hat{A}_{i,2}^{\text{MT-RLOO}} = \hat{A}_i^O. \tag{12}$$

where λ is the turn-level advantage coefficient, $\hat{A}_i^T$ and $\hat{A}_i^O$ are computed:

$$\hat{A}_i^T = \frac{G}{G-1}\left(R_i^T - \text{mean}(\{R_i^T\}_{i=1}^G)\right), \quad \hat{A}_i^O = \frac{G}{G-1}\left(R_i^O - \text{mean}(\{R_i^O\}_{i=1}^G)\right). \tag{13}$$

D. Additional Experiment Results

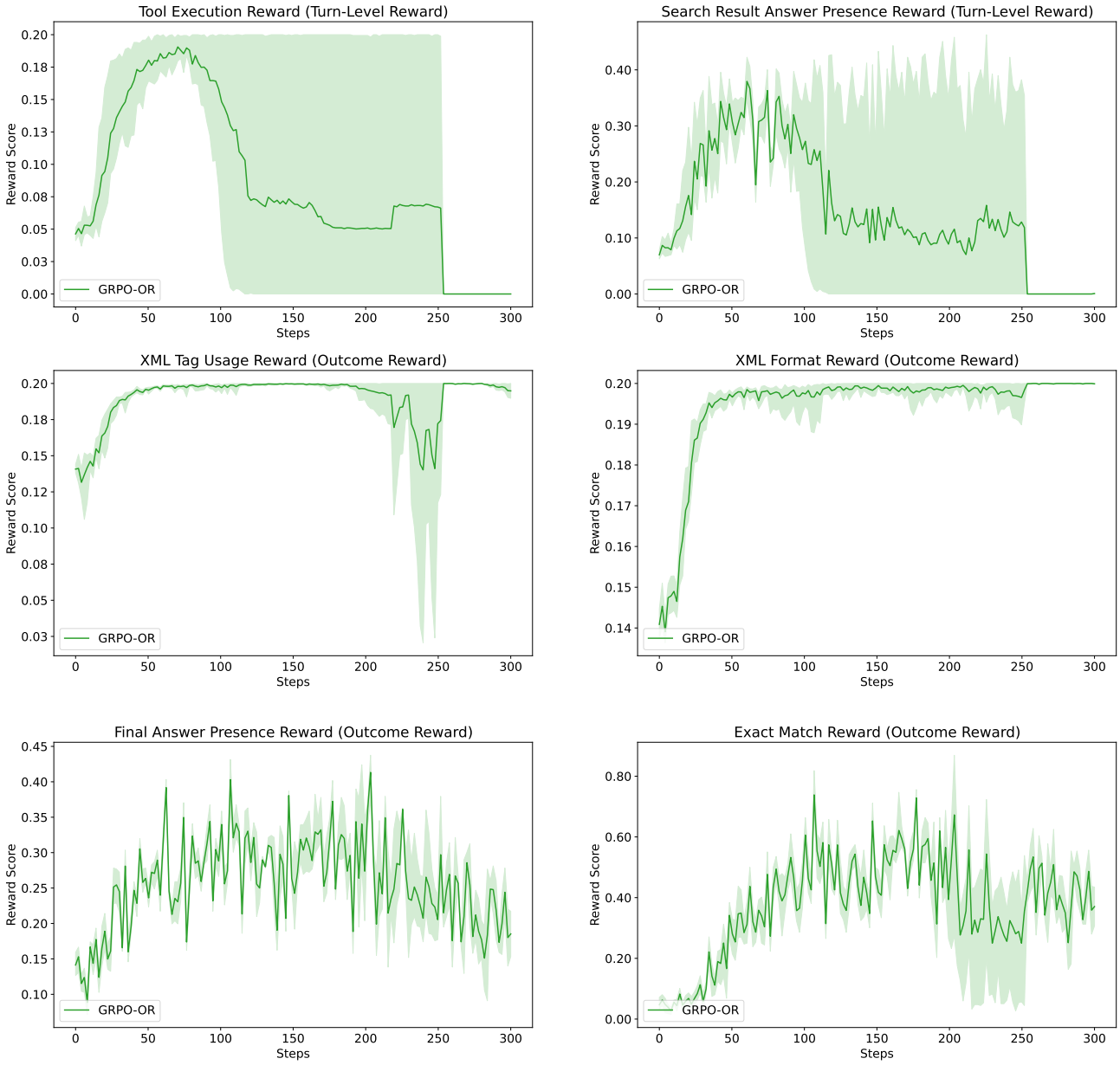

Figure 3: Curves for different training reward components during training using GRPO-OR, where shaded regions represent the range between the maximum and minimum values across 10 runs.

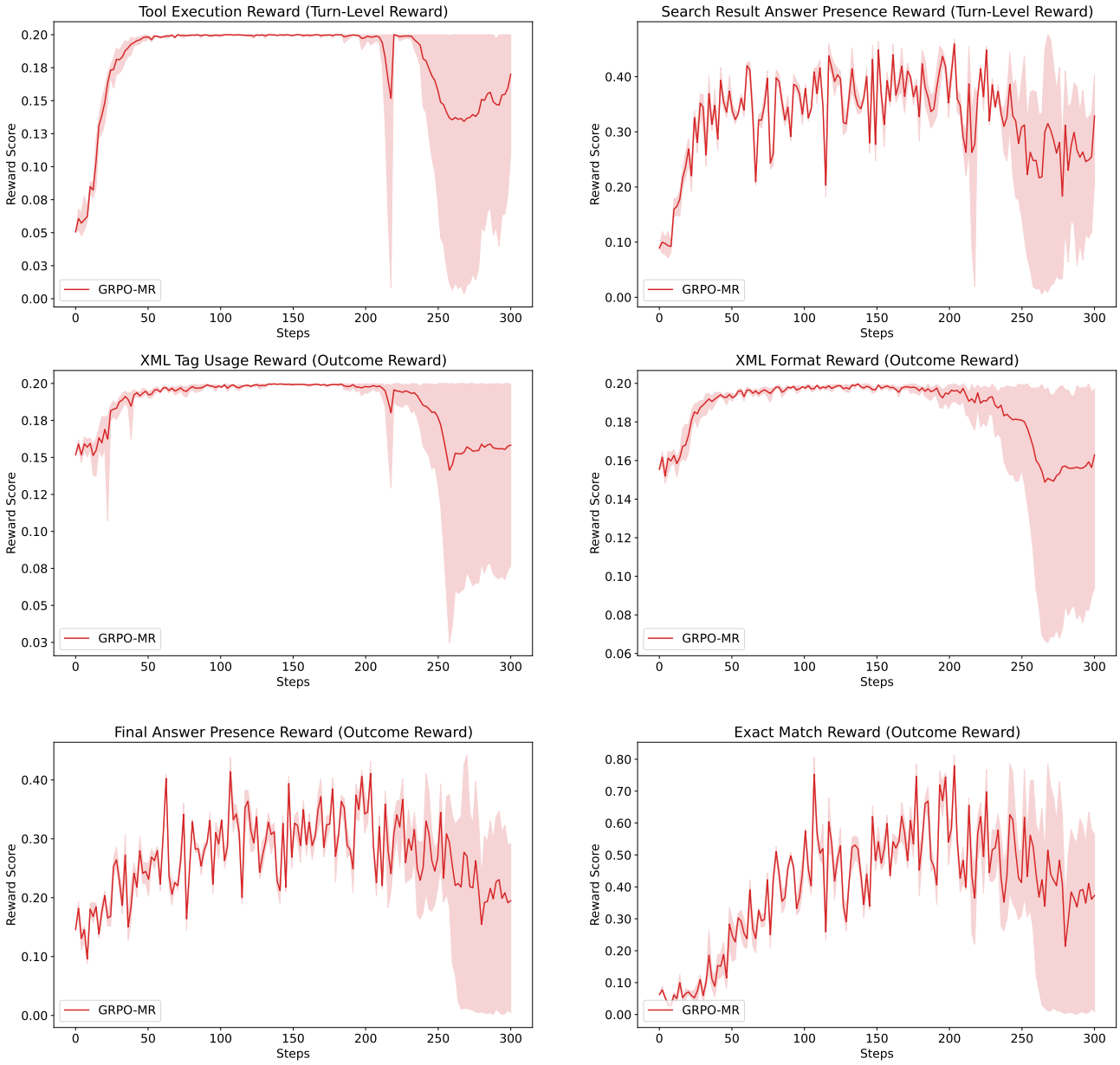

Figure 4: Curves for different training reward components during training using GRPO-MR, where shaded regions represent the range between the maximum and minimum values across 10 runs.

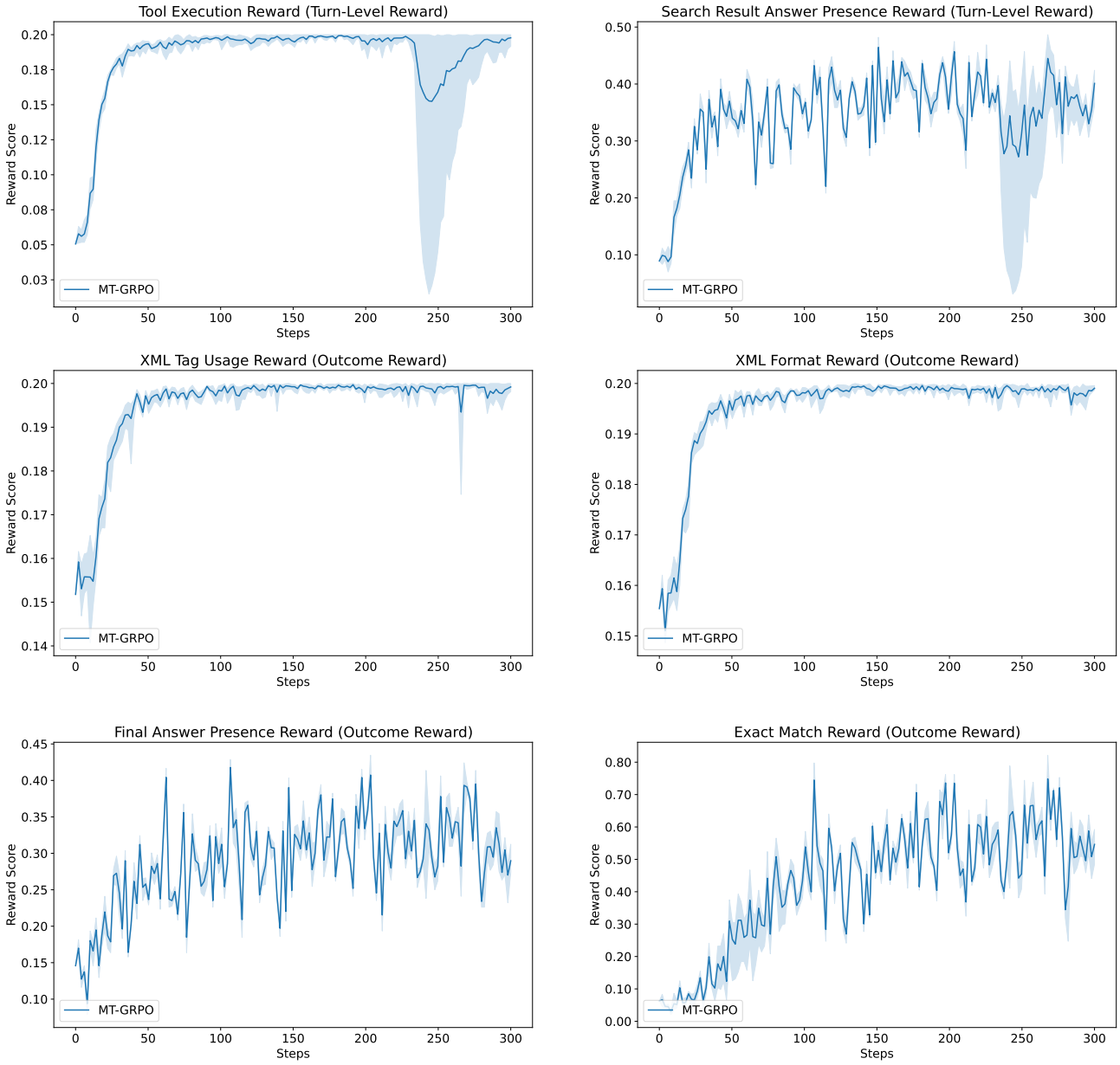

Figure 5: Curves for different training reward components during training using MT-GRPO, where shaded regions represent the range between the maximum and minimum values across 10 runs.