Tool-Integrated Reasoning via Hierarchical Multi-Agent Reinforcement Learning

Anonymous ACL submission

Abstract

Recent advancements in Tool-Integrated Reasoning (TIR) empower Large Language Models (LLMs) with external utilities to overcome intrinsic deficits in knowledge currency and numerical computation. However, existing methods often face a dilemma: single-agent models lack structural interpretability due to entangled planning and execution, while multi-agent systems suffer from unstable tool usage caused by misaligned optimization between high-level planners and low-level executors. To address these challenges, we propose a **T**ree-structured **M**ulti-**A**gent **T**ool **R**easoning framework optimized via **H**ierarchical **R**einforcement **L**earning (TMATR-HRL). Specifically, we introduce the TMATR system, which structures reasoning into a hierarchical tree of atomic decisions, explicitly decoupling strategic planning from step-wise execution to enhance interpretability. To ensure stability and coordination, we employ a HRL scheme that enables the co-evolution of planning and execution policies through alternating on-policy updates. Experiments on mathematical reasoning and question-answering benchmarks show that TMATR-HRL consistently achieves performance improvements across multiple models, while exhibiting significant advantages in the controllability and interpretability of tool usage.

1 Introduction

Large language models (LLMs) (Yang et al., 2025; Liu et al., 2024; Abdin et al., 2024) have demonstrated remarkable capabilities in mathematical reasoning and commonsense question answering (QA), fueling applications in education (Cheng et al., 2025), healthcare (Luo et al., 2025), and law (Li et al., 2025a). Despite these successes, LLMs face intrinsic limitations regarding knowledge currency and symbolic computation. First, parametric knowledge is static, inevitably becoming stale compared to evolving real-world facts. Second, the next-token prediction objective is often misaligned with the rigorous logic required for precise numerical calculation (Jia et al., 2025). To mitigate these deficits, Tool-Integrated Reasoning (TIR) has emerged as a promising paradigm, enabling LLMs to act as agents that invoke external tools (retrievers, calculators, and code interpreters) to ground their reasoning in accurate execution (Jia et al., 2025; Dong et al., 2025a; Li et al., 2025d)

Despite progress, existing TIR approaches (Li et al., 2025d; Hu et al., 2025; Li et al., 2025c; Qian et al., 2025; Feng et al., 2025) face a dilemma in balancing structural interpretability with tool-use stability. One line of work decomposes tasks into specialized multi-agent systems (Li et al., 2025d; Hu et al., 2025), assigning distinct roles for planning and execution to enhance clarity. However, these methods often suffer from a coordination deficit during optimization. Typically, they focus on training the high-level planner while keeping the executor fixed or optimizing them in isolation via static Supervised Fine-Tuning (SFT). This disjoint optimization creates a competency mismatch: the planner may generate sophisticated strategies that the executor cannot reliably fulfill, or the executor may fail to provide the feedback the planner expects. Consequently, without joint adaptation through dynamic interaction, tool usage becomes unstable and prone to progressive degradation across long reasoning horizons.

Conversely, single-agent paradigms (Li et al., 2025c; Qian et al., 2025; Feng et al., 2025) handle both planning and tool invocation within a unified model. While this enables joint optimization of all capabilities, it sacrifices structural constraints. Lacking explicit role boundaries, these models tend to be myopic and entangled, often reacting to immediate tool feedback without a coherent global strategy. This unstructured approach not only obscures the decision-making process, making it difficult to interpret why a tool is invoked, but also

leads to reactive, redundant, or erratic tool usage patterns. Thus, a core challenge remains: *how to impose a structured, interpretable reasoning hierarchy while ensuring the coordinated evolution of all system components?*

To address these challenges, we propose a **T**ree-structured **M**ulti-**A**gent **T**ool **R**easoning framework optimized via **H**ierarchical **R**einforcement **L**earning (TMATR-HRL). First, to ensure interpretability, we introduce the Tree-structured Multi-Agent Reasoning Tool (TMATR) system. TMATR organizes the problem-solving process into a hierarchical tree of atomic decisions, explicitly decoupling high-level strategic planning from low-level execution and exposing the rationale behind every tool invocation. Second, to resolve the coordination deficit, we propose a hierarchical reinforcement learning scheme (HRL) that treats planning and execution as separate but co-evolving policies. Instead of static supervision, we employ alternating on-policy updates: fixing one agent while optimizing the other against the system-level reward. This approach allows the planner to adapt to the executor's boundaries and the executor to align with the planner's intent, ensuring robust and stable performance across multi-turn interactions. Experiments on math reasoning and QA benchmarks show TMATR-HRL is effective and generalizes across multiple backbone models.

Our contributions are summarized as follows:

- We introduce a **tree-structured multi-agent tool-integrated reasoning system** that decouples high-level planning from step-wise execution, organizing tool use into interpretable hierarchical units.
- We propose a **hierarchical reinforcement learning** scheme that jointly optimizes planning and execution policies via alternating updates, ensuring coordinated improvement across abstraction levels.
- Extensive experiments on mathematical reasoning and QA benchmarks validate the effectiveness of TMATR-HRL, which produces more controllable and robust tool-use.

2 Preliminary

2.1 Problem Setup

Let $\Pi = \{\pi_i\}_{i=1}^{N}$ denote a multi-agent system (MAS), where each π_i is an instruction-following large language model that generates actions based on the evolving dialogue context. The MAS interacts with a tool buffer $\mathcal{T}$, where each tool $t \in \mathcal{T}$ takes an input argument u, queries external information sources, and returns a textual observation o. Given an input $x \in \mathcal{X}$, the MAS produces an output $\hat{y} \in \mathcal{Y}$ through multi-turn interaction under a maximum turn budget H. We denote the induced decision process by $\mathcal{F}(\cdot\,; \Pi, \mathcal{T}, H)$. We formulate the task as a MAS, such that

$$\mathcal{F}(x; \Pi, \mathcal{T}, H) \to \hat{y}, \tag{1}$$

which produces the system output for a given x.

2.2 Multi-Agent POMDP Formulation

We formalize the tool-integrated multi-agent reasoning process as a finite-horizon partially observable Markov decision process (POMDP):

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \mathcal{O}, \mathcal{R}, \Pi, H). \tag{2}$$

Here, $\mathcal{S}$ denotes the latent reasoning state space and $s_t \in \mathcal{S}$ is the state at turn t. The action space $\mathcal{A}$ contains internal reasoning actions and optional tool-invocation actions, and the transition kernel P captures state updates conditioned on these actions. The observation space $\mathcal{O}$ includes textual observations accessible to the agents, such as dialogue context and tool feedback. The agent set $\Pi = \{\pi_i\}_{i=1}^{N}$ consists of N cooperative language-model agents, and H is the maximum episode length.

At turn t, each agent π_i receives a partial observation $\tau_{i,0:t}$ summarizing all information available to that agent up to turn t. The joint policy factorizes in a decentralized manner:

$$\pi(\mathbf{a}_{t+1} \mid \boldsymbol{\tau}_{0:t+1}) = \prod_{i=1}^{N} \pi_i(a_{i,t} \mid \tau_{i,0:t}), \tag{3}$$

where $\mathbf{a}_{t+1} = (a_{1,t+1}, \ldots, a_{N,t+1})$ is the joint action. $\boldsymbol{\tau}_{0:t+1} = (\tau_{1,0:t+1}, \ldots, \tau_{N,0:t+1})$ is the collection of agent trajectories.

The reward is computed for each agent based on its own trajectory and the reference answer y:

$$r(\tau_{i,0:t_{\text{end}}}, y) \in \mathcal{R}. \tag{4}$$

The learning objective is to maximize the expected terminal reward across agents:

$$\max_{\Pi} \; \mathbb{E}_{\Pi,P}\left[\frac{1}{N}\sum_{i=1}^{N} r(\tau_{i,0:t_{\text{end}}}, y)\right]. \tag{5}$$

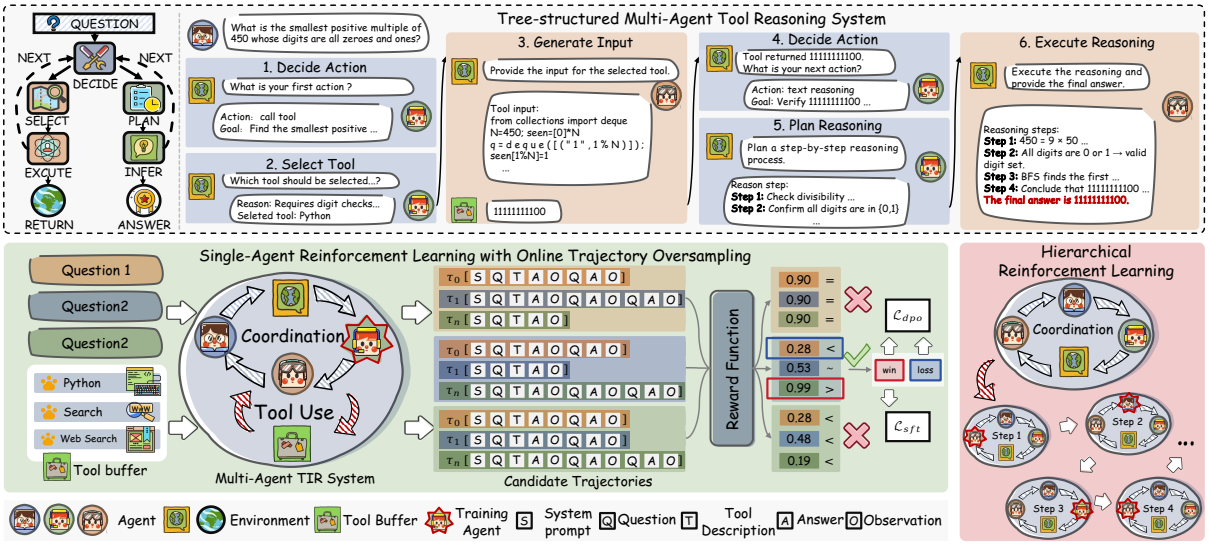


Figure 1: Overview of **TMATR-HRL**: A tree-structured multi-agent reasoning system separates planning and execution for TIR. An online trajectory-based preference learning module optimizes the agent via on-policy reinforcement learning. A hierarchical reinforcement learning framework alternates updates agents.

2.3 Direct Preference Optimization

Direct Preference Optimization (DPO) (Rafailov et al., 2023) is an offline preference-based fine-tuning method that directly updates a target policy π_θ using pairwise comparisons. DPO constructs a closed-form objective based on likelihood ratios between the target policy and a fixed reference policy π_{ref}, eliminating the need for reward models or value estimation.

Given a preference triplet (x, z^+, z^-), where x is the prompt and z^+ and z^- denote the win and loss responses, respectively, the DPO objective is

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,z^+,z^-)\sim\mathcal{D}}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(z^+|x)}{\pi_\theta(z^-|x)} - \beta \log \frac{\pi_{\text{ref}}(z^+|x)}{\pi_{\text{ref}}(z^-|x)}\right)\right], \tag{6}$$

where $\sigma(\cdot)$ is the sigmoid function and $\beta > 0$ controls the preference margin. The objective increases the likelihood of preferred responses while regularizing updates through the reference model, providing a stable and efficient formulation for offline preference learning.

3 Method

As illustrated in Figure 1, TMATR-HRL comprises three tightly coupled components. A tree-structured multi-agent tool reasoning system separates high-level planning from low-level execution via structured interactions. A single-agent reinforcement learning component with online trajectory oversampling improves long-horizon tool-integrated reasoning through online DPO and trajectory filtering. A hierarchical reinforcement learning framework coordinates agent training via alternating single-agent updates.

3.1 Tree-structured Multi-Agent Tool Reasoning System

TIR requires high-level planning before invoking external tools. Existing systems (Li et al., 2025c; Qian et al., 2025; Feng et al., 2025) often interleave planning and execution within a single decoding sequence, which entangles structural decisions with low-level reasoning and leads to reactive tool use. TMATR addresses this limitation by structuring each tool-use turn as a tree-guided multi-agent process that separates high-level control from step-wise execution.

System Overview. TMATR decomposes problem solving into a sequence of turns indexed by t. Each turn expands a directed tree

$$\mathcal{B}^{(t)} = \left(\mathcal{V}^{(t)}, \mathcal{E}^{(t)}\right), \tag{7}$$

whose nodes represent atomic actions and whose edges encode precedence relations. This structure exposes planning decisions before execution and forms a hierarchical scaffold aligned with the POMDP formulation.

Atomic Action Space. Atomic actions are partitioned by the component that produces them:

$$\begin{aligned} \mathcal{A}_{\text{pla}} &= \{\texttt{DECIDE}, \texttt{SELECT}, \texttt{PLAN}\}, \\ \mathcal{A}_{\text{inf}} &= \{\texttt{EXECUTE}, \texttt{INFER}, \texttt{ANSWER}\}, \\ \mathcal{A}_{\text{env}} &= \{\texttt{RETURN}\}, \\ \mathcal{A}_{\text{atom}} &= \mathcal{A}_{\text{pla}} \cup \mathcal{A}_{\text{inf}} \cup \mathcal{A}_{\text{env}}. \end{aligned} \tag{8}$$

Planner actions define the high-level structure, inferencer actions realize execution, and RETURN is generated by the environment.

Tool Buffer. TMATR maintains a tool buffer

$$\mathcal{T} = \{t_j\}_{j=1}^{K}$$

where each tool t_j is identified by a name d_{name} and is associated with a description d_{des}, an input specification d_{in}, an output specification d_{out} and an example d_{ex}. These descriptors define how the tool receives arguments, how its output is interpreted and how the tool is used in typical scenarios.

A deterministic function retrieves the descriptor set given a tool name

$$f(d_{\text{name}}) = (d_{\text{name}}, d_{\text{des}}, d_{\text{in}}, d_{\text{out}}, d_{\text{ex}}). \quad (9)$$

During a `SELECT` action the planner π_{pla} outputs a tool name, and the inferencer π_{inf} applies the function f to obtain the corresponding input and output specifications and example which guide the construction of valid tool parameters.

Multi-Agent Roles under Partial Observability. Two cooperative agents operate under partial observability. At the beginning of turn t, the planner receives its private history $\tau_{0:t-1}^{\text{pla}}$ and performs the atomic action `DECIDE`:

$$\big(g_t, a_t^{\text{pla}}\big) \sim \pi_{\text{pla}}\big(\cdot \mid \tau_{0:t-1}^{\text{pla}}, \texttt{DECIDE}\big), \quad (10)$$

where g_t is the subgoal and $a_t^{\text{pla}} \in \{\texttt{SELECT}, \texttt{PLAN}\}$ chooses the branch. The textual output of `DECIDE` is appended to the dialogue, producing updated histories $\tau_{0:t}^{\text{pla},+}$.

Tool Branch. If $a_t^{\text{pla}} = \texttt{SELECT}$, the planner chooses a tool identifier

$$t_j \sim \pi_{\text{pla}}\big(\cdot \mid \tau_{0:t}^{\text{pla},+}, g_t, \texttt{SELECT}\big), \quad (11)$$

and retrieves descriptors $f(t_j)$. Next, the inferencer binds executable parameters

$$u_t \sim \pi_{\text{inf}}\big(\cdot \mid \tau_{0:t-1}^{\text{inf}}, g_t, f(t_j), \texttt{EXECUTE}\big). \quad (12)$$

The environment executes the tool call and returns

$$o_t = t_j(u_t), \quad (13)$$

which is appended to the dialogue and incorporated into $\tau_{0:t+1}^{\text{pla}}$. This design allows observations to be naturally integrated into dialogue history, avoiding the need for explicit loss masking during training.

Internal Reasoning Branch. If $a_t^{\text{pla}} = \texttt{PLAN}$, the planner generates a meta plan

$$p_t \sim \pi_{\text{pla}}\big(\cdot \mid \tau_{0:t}^{\text{pla},+}, g_t, \texttt{PLAN}\big), \quad (14)$$

and the inferencer π_{inf} produces an output

$$o_t \sim \pi_{\text{inf}}\big(\cdot \mid \tau_{0:t}^{\text{inf}}, g_t, p_t, \texttt{INFER}\big). \quad (15)$$

The output o_t is always appended to the dialogue and incorporated into the planner's observation for turn $t+1$. If the inferencer π_{inf} emits a final answer

$$\hat{y} \sim \pi_{\text{inf}}\big(\cdot \mid \tau_{0:t}^{\text{inf}}, g_t, \texttt{ANSWER}\big), \quad (16)$$

the answer is appended in the same manner, and the episode terminates without initiating a new turn.

3.2 Single-Agent Reinforcement Learning with Online Trajectory Oversampling

Reinforcement learning is used to improve policy behavior in long-horizon tool-integrated reasoning. Prior works (Kang et al., 2023; Rafailov et al., 2023) typically apply offline preference optimization over static trajectories, which prevents the policy from adapting to the evolving interaction patterns between the agents in TMATR. Moreover, naively applying online RL to long-horizon TIR leads to degraded trajectory quality, as accumulated tool noise introduces instability in long rollouts. We address these issues by adopting an online variant of DPO in which trajectories are generated in real time under the current policies of both agents, and a trajectory-oversampling procedure selectively retains high-margin rollouts to guide preference learning.

Reward Design. The reward is computed at the completion of each rollout and evaluates both the final prediction and the behavioral properties of the generated trajectory. Given the produced answer y and ground truth $y^\star$, correctness is defined as

$$R_{\text{corr}}(y, y^\star) \in [0, 1], \quad (17)$$

which combines symbolic matching and numerical agreement. Long-horizon tool-integrated reasoning introduces additional variability, so the reward also accounts for tool efficiency, output repetition and trajectory length through, $R_{\text{tool}}(\tau)$, $R_{\text{rep}}(\tau)$, $R_{\text{len}}(\tau)$, where R_{tool} reflects an adaptive preference over the frequency and success rate of tool calls, R_{rep} measures lexical redundancy and R_{len} imposes a nonlinear penalty

as the output approaches a predefined maximum length. These components are aggregated as

$$R_{\text{aux}} = \alpha\,(2R_{\text{tool}} - 1) - \beta\,R_{\text{rep}} + \gamma\,R_{\text{len}}, \quad (18)$$

and are normalized to obtain $\widetilde{R}_{\text{aux}} \in [0, 1]$. The final reward interpolates between correctness and auxiliary feedback with an adaptive mixing coefficient ϵ that increases when correctness is low:

$$R(y, \tau) = (1 - \epsilon)\,R_{\text{corr}}(y, y^\star) + \epsilon\,\widetilde{R}_{\text{aux}}. \quad (19)$$

This reward formulation captures the structural characteristics of TMATR and provides informative preference signals for online DPO under long-horizon, tool-dependent rollouts.

Preference Construction. For each input x, the agent generates n trajectories under the current policy, denoted by $\mathcal{T}(x) = \{\tau_1, \ldots, \tau_n\}$, where each trajectory τ_i receives a scalar reward $R(\tau_i)$. The win and loss trajectories are identified as

$$\tau^+ = \arg\max_{\tau_i \in \mathcal{T}(x)} R(\tau_i), \quad (20)$$

$$\tau^- = \arg\min_{\tau_i \in \mathcal{T}(x)} R(\tau_i). \quad (21)$$

To avoid degenerate preference signals, we apply a filtering operator $\Phi(\cdot)$ that discards uninformative pairs. Specifically, a preference pair (τ^+, τ^-) is retained by Φ only if it satisfies

$$R(\tau^+) > \eta \quad \text{and} \quad R(\tau^+) \neq R(\tau^-), \quad (22)$$

where η is a reward threshold. The resulting preference set is defined as

$$\mathcal{P}(x) = \Phi\big(\{(\tau^+, \tau^-)\}\big). \quad (23)$$

Mini-batches $\mathcal{B}$ sampled from $\bigcup_x \mathcal{P}(x)$ are used to perform online DPO updates.

Training Objective. Training jointly optimizes Direct Preference Optimization and supervised fine-tuning on high-reward trajectories. The DPO loss encourages the policy to assign higher likelihood to preferred trajectories τ^+ than to non-preferred ones τ^- constructed from online rollouts, while the SFT loss anchors the policy by imitating the corresponding preferred trajectories. The combined objective is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{DPO}} + \mathcal{L}_{\text{SFT}}. \quad (24)$$

This joint formulation allows the policy to leverage relative preference signals from online trajectory comparisons while retaining strong supervision from high-quality rollouts.

3.3 Hierarchical Reinforcement Learning

This stage is designed to enhance the reasoning capability of the multi-agent system using hierarchical reinforcement learning. Existing approaches (Li et al., 2025d; Hu et al., 2025) often optimize only a single policy or update all components uniformly. Such strategies ignore the asymmetry between planning and execution, which can lead to unstable tool usage or biased planning behavior. In particular, an under-trained inferencer may produce noisy tool feedback, causing the planner to avoid tools altogether. To address this issue, we adopt a hierarchical reinforcement learning scheme with alternating optimization over a set of agents. Training proceeds in iterations indexed by k, where online trajectories are generated under the current agents.

At each iteration, only one agent is updated while the others are kept fixed, following a predefined order in the hierarchy. Let $\{\pi_i^{(k)}\}_{i=0}^{L}$ denote the policies of all agents at iteration k. The update at iteration k takes the form

$$\pi_i^{(k+1)} \leftarrow \text{Update}\big(\pi_i^{(k)} \mid \{\pi_j^{(k)}\}_{j \neq i}\big), \quad (25)$$

where the index i is selected according to the hierarchical training schedule. This alternating update strategy stabilizes training by preventing uncontrolled co-adaptation across agents. Here $\text{Update}(\cdot)$ denotes the single-agent optimization procedure defined in the previous subsection.

4 Experiment

Our experiments are designed to answer the following questions: **Q1.** Does the proposed method achieve consistent performance improvements across math and QA benchmarks? **Q2.** Is the hierarchical reinforcement learning framework effective in jointly optimizing planning and inference? **Q3.** How does the model leverage external tools during reasoning, and how do tool usage patterns evolve across different training stages? **Q4.** How does the training dynamics evolve under hierarchical reinforcement learning?

4.1 Experiment setting

Data Construction. We construct our experimental data from several publicly available math and question answering datasets, including **NuminaMath-TIR** (Li et al., 2024), **HotpotQA** (Yang et al., 2018),

2WikiMultiHopQA (Ho et al., 2020), **SimpleDeepSearch** (Sun et al., 2025), and **WebWalker** (Wu et al., 2025). To obtain data suitable for reinforcement learning, we first evaluate all instances using Qwen2.5-7B-Instruct (Bai et al., 2025) under a pass@1 setting and filter out instances that can be solved correctly in a single attempt. We then apply DeepSeek-V3 (Liu et al., 2024) with the proposed TMATR pipeline, discarding instances that cannot be solved within five turns. From the remaining pool, we randomly sample 3,000 instances for training, maintaining a 2:1 ratio between reasoning and question answering examples. Details are in Appendix A.

Tool Settings. We equip the model with three external tools (**Search**, **Python**, **WebSearch**). The tool configuration follows Tool-Star (Dong et al., 2025a). Search uses the Bing Search API to retrieve relevant documents from the internet. Python supports code execution for numerical computation and symbolic reasoning. WebSearch retrieves the top ten relevant web pages through Bing search and produces a concise summary using a locally deployed Qwen2.5-72B-Instruct (Bai et al., 2025).

Benchmarks. We conduct evaluations on a broad range of benchmarks covering math and QA. The mathematical benchmarks span competition-style and curriculum-based settings, including AIME24 (Zhang and Math-AI, 2024), AIME25 (Zhang and Math-AI, 2025), MATH500 (Lightman et al., 2023), GSM8K (Cobbe et al., 2021), AMC23[1], Gaokao Math Cloze (Zhong et al., 2023), Gaokao Math QA (Zhong et al., 2023), Gaokao2023-EN[2], Gaokao2024-I/II [3], Gaokao2024-Mix [4] and OlympiadBench [5]. For question answering, we consider datasets that require multi-hop reasoning and information aggregation, including WebWalker (Wu et al., 2025), HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQ (Trivedi et al., 2022) and Bamb (Press et al., 2023). Evaluation across all benchmarks is performed using pass@1 accuracy, with DeepSeek-v3.1[6] employed as the LLM judge to assess answer correctness.

[1] https://huggingface.co/datasets/math-ai/amc23

[2] https://huggingface.co/datasets/MARIO-Math-Reasoning/Gaokao2023-Math-En?utm_source=chatgpt.com

[3] https://github.com/llmeval/Llmeval-Gaokao2024-Math

[4] https://github.com/QwenLM/Qwen2.5-Math

[5] https://huggingface.co/datasets/math-ai/olympiadbench

[6] https://huggingface.co/deepseek-ai/DeepSeek-V3.1

Models. TMATR-HRL adopts a decoupled architecture. The planner is fixed to **Qwen3-4B-Instruct-2507** (Yang et al., 2025) across all experiments. For the inferencer, we consider multiple backbone models, including **Qwen3-4B-Instruct-2507** (Yang et al., 2025), **DeepSeek-R1-Distill-Qwen-1.5B** (Guo et al., 2025), and **Phi-4-mini-instruct** (Abdin et al., 2024), in order to evaluate performance under different inference configurations. The official thinking mode of all models is disabled. Additional details are in Appendix B.

Baselines. We primarily compare our method against two widely used prompt-based reasoning approaches, with detailed prompt designs provided in Appendix C. Training-based methods (Li et al., 2025d,c; Qian et al., 2025; Feng et al., 2025) optimize with PPO or GRPO and are not directly comparable due to differing objectives, tool setups, and base models. In addition, as a supplementary comparison, we evaluate several larger open-source models, including Qwen3-14/32B (Yang et al., 2025), Ministral-8B-Instruct-2410[7], and DeepSeek-R1-0528-Qwen3-8B (Guo et al., 2025), under two prompt settings and report the best performance.

Experiment Details. All training runs are conducted with 8 A800 GPUs. To improve training efficiency for DeepSeek-R1-Distill-Qwen-1.5B (Guo et al., 2025) and Phi-4-mini-instruct (Abdin et al., 2024), we additionally construct an 8.5K tool-integrated reasoning dataset using DeepSeek-V3 (Liu et al., 2024) for supervised fine-tuning. Auxiliary 6 GPUs are provisioned to host frozen models and tool backends such as WebSearch execution environments to support model interaction. Detailed settings are provided in the Appendix D.

4.2 Main results (Q1)

Math Results. As shown in Table 1, first, TMATR-HRL consistently improves performance across all math benchmarks and backbone models, and uniformly outperforms the non fine tuned TMATR. The gains are more pronounced on harder tasks, suggesting that HRL brings robust and largely model agnostic improvements. Second, TMATR already outperforms pure reasoning baselines, indicating that structured multi stage reasoning provides a useful inductive bias. However, its performance remains consistently below that of TMATR-HRL, which suggests that structure alone

[7] https://huggingface.co/mistralai/Ministral-8B-Instruct-2410

Method	AIME24	AIME25	MATH500	GSM8K	AMC23	Gk Cloze	Gk QA	Gk23 En	Gk24 I	Gk24 II	Gk24 Mix	Olymp.	Avg.
Qwen3-14B	23.33	23.33	87.80	**96.20**	72.50	78.81	81.30	79.48	78.57	50.00	73.63	56.20	66.76
Qwen3-32B	36.67	26.67	86.60	95.40	57.50	75.42	85.19	78.44	78.57	64.29	70.88	53.60	67.44
Ministral-3-8B	3.33	0.00	50.40	83.80	25.00	45.76	53.85	50.65	42.86	35.71	40.66	23.40	37.95
DeepSeek-R1-8B	33.33	33.33	88.80	78.00	70.00	**90.68**	87.75	81.30	85.71	71.43	72.53	52.40	70.45
Qwen3-4B-Instruct-2507													
+ boxed	36.67	33.33	88.40	95.00	80.00	83.05	83.19	80.78	85.71	**78.57**	69.23	54.00	72.33
+ cot	36.67	20.00	88.00	95.20	75.00	82.20	82.22	82.60	78.57	71.43	70.33	53.00	69.60
+ TMATR	43.33	**50.00**	89.80	90.80	**85.00**	80.51	88.60	81.82	85.71	71.43	**82.42**	54.80	75.35
+ TMATR-HRL	**60.00**	46.67	**90.80**	91.80	**85.00**	81.36	**90.60**	**83.64**	**100.00**	**78.57**	79.12	**59.00**	**78.88**
DeepSeek-R1-Distill-Qwen-1.5B													
+ boxed	23.33	20.00	81.20	85.20	**65.00**	75.42	80.34	74.81	64.29	50.00	63.74	39.60	60.24
+ cot	30.00	25.00	**82.80**	77.60	**65.00**	**78.81**	79.20	**76.36**	**71.43**	50.00	61.54	41.60	61.61
+ TMATR	16.67	**30.00**	76.20	**86.60**	50.00	76.27	77.49	71.17	64.29	**57.14**	63.74	41.80	59.28
+ TMATR-HRL	**33.33**	20.00	77.20	86.20	**65.00**	73.73	**81.97**	70.13	**71.43**	**57.14**	**64.84**	**44.44**	**62.12**
Phi-4-mini-instruct													
+ box	0.00	3.33	34.80	68.80	20.00	34.75	44.16	37.92	28.57	28.57	28.57	8.80	28.19
+ cot	3.33	0.00	34.00	58.20	15.00	29.66	37.32	34.03	35.71	50.00	24.18	8.80	27.52
+ TMATR	13.33	13.33	68.20	84.60	40.00	56.78	61.54	64.42	57.14	50.00	57.14	**39.00**	50.46
+ TMATR-HRL	**30.00**	**20.00**	**73.80**	**86.00**	**67.50**	**69.49**	**71.95**	**65.19**	**64.29**	**57.14**	**58.70**	37.60	**58.47**

Table 1: Pass@1 accuracy of TMATR-HRL across multiple mathematical datasets and model variants. The best results are highlighted in bold, and the second-best results are underlined.

Method	WebWalker	HQA	2Wiki	MusiQ	Bamb	Avg.
Qwen3-14B	4.50	27.50	32.50	**12.50**	48.80	25.16
Qwen3-32B	3.00	30.50	31.50	12.00	**53.60**	26.12
Ministral-3-8B	2.50	23.00	22.50	8.50	32.80	17.86
DeepSeek-R1-8B	0.00	4.50	10.50	3.50	13.60	6.42
Qwen3-4B-Instruct-2507						
+ box	2.50	24.50	27.00	9.50	34.40	19.58
+ cot	2.50	25.00	26.50	8.50	31.20	18.74
+ TMATR	9.00	42.00	**42.50**	10.50	49.60	30.72
+ TMATR-HRL	**11.00**	**43.50**	41.50	12.00	48.80	**31.36**
DeepSeek-R1-Distill-Qwen-1.5B						
+ box	0.00	0.50	6.25	0.00	0.80	1.51
+ cot	0.50	0.00	0.00	0.50	0.00	0.20
+ TMATR	2.34	21.00	20.50	6.50	24.80	15.03
+ TMATR-HRL	**2.50**	**28.00**	**26.00**	**7.81**	**30.40**	**18.94**
Phi-4-mini-instruct						
+ box	1.00	13.50	17.00	4.00	12.00	5.60
+ cot	1.50	14.00	9.00	2.00	16.80	5.53
+ TMATR	7.50	31.50	29.50	8.00	41.60	12.70
+ TMATR-HRL	**11.50**	**45.00**	**33.50**	**10.50**	**42.40**	**28.58**

Table 2: Pass@1 accuracy of TMATR-HRL across QA datasets and models. The best results are highlighted in bold, and the second-best results are underlined.

is insufficient without task specific optimization. Third, pure reasoning methods perform competitively on simpler benchmarks but show larger performance drops as task difficulty increases. This reflects the limitations of static prompting in long horizon mathematical reasoning. Fourth, TMATR-HRL consistently improves upon TMATR across benchmarks and model settings. This supports the effectiveness of HRL in jointly optimizing planning and inference for TIR.

QA Results. First, as shown in Table 2, TMATR-HRL achieves the best average performance across all backbone models and consistently outperforms TMATR and pure reasoning methods, with clearer gains on QA tasks requiring multi-step reasoning and frequent tool use. This indicates that jointly optimizing planning and execution is especially effective for long-horizon QA. Second, compared with pure reasoning methods, TMATR and TMATR-HRL exhibit more consistent performance across QA tasks, while pure reasoning approaches show larger performance variation. This can be attributed to the structured multi-agent reasoning process, which enables clearer decision decomposition.

4.3 Ablation Study (Q2)

To address Q2, we report intermediate pass@1 accuracy during training in Figure 2. The inference model is trained first, followed by planner training and full system optimization. First, fine tuning the inference model alone brings limited gains, while adding planner training and system level optimization yields clear improvements, supporting the need to stabilize inference before joint optimization. Second, on DeepSeek-R1-Distill-Qwen-1.5B, performing an additional round of inference model training further improves performance, suggesting that repeated refinement of the inference component can be beneficial. Third, accuracy is not monotonic during training, which we attribute to temporary capability mismatches between agents, highlighting the importance of coordinated optimization.

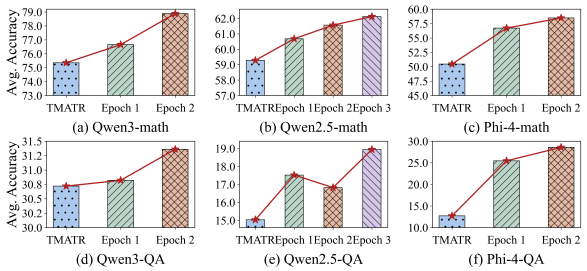
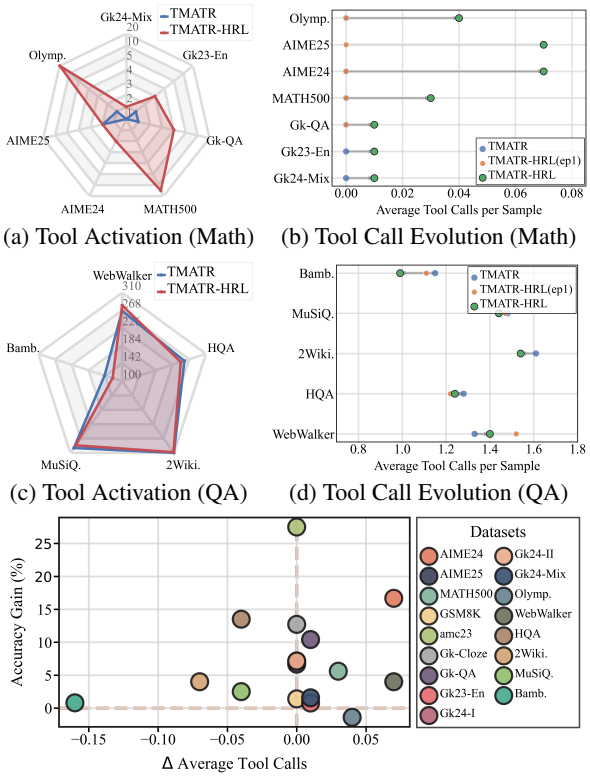


Figure 2: Model analysis of HRL framework.

4.4 Tool Usage Analysis (Q3)

To investigate Q3, we analyze behavioral shifts using Delta Average Tool Calls (Δ Avg. Tool

Calls) and Accuracy Gain. First, in mathematical domains, the agent adaptively expands tool usage. Figures 3a and 3c show that tool activation on AIME24 and MATH500 significantly exceeds the baseline, indicating that training identifies reasoning bottlenecks and invokes tools as compensation. Second, in retrieval settings, trajectories shift from exploration to on-demand precision. Dumbbell plots in Figures 3b and 3d show that although tool calls initially increase, they are pruned in the final stage for QA tasks, reflecting an internalized cost–benefit trade-off. Third, these patterns suggest that performance gains arise from decision quality rather than call frequency. Figure 3e shows multiple datasets in the upper-left quadrant (negative Δ Tool Calls with positive Gain), highlighting a policy that optimizes when and whether to invoke tools for better cost–accuracy balance.



(a) Tool Activation (Math) (b) Tool Call Evolution (Math)

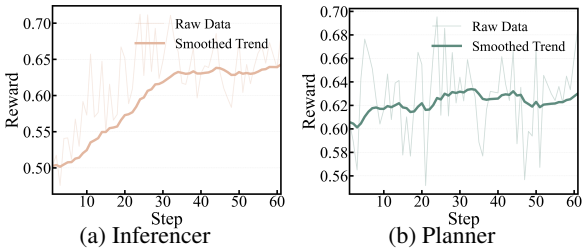
(c) Tool Activation (QA) (d) Tool Call Evolution (QA)

(e) Accuracy Gain

Figure 3: Tool-use Analysis on **Phi-4**.

4.5 Training Dynamics (Q4)

To address Q4, Figure 4 shows TMATR-HRL training rewards on Qwen3, with the Inferencer trained before the Planner. The Inferencer converges rapidly around 0.65, while the Planner improves more gradually with higher variance. Their eventual convergence indicates effective coordination between planning and execution.



(a) Inferencer (b) Planner

Figure 4: Training reward curves on **Qwen3**.

5 Related Works

Tool-Integrated Reasoning (TIR) enables language models to interact with external tools (Jia et al., 2025). We categorize prior work into three groups. (1) Supervised fine-tuning (SFT) trains TIR systems to imitate tool-use trajectories from manual or model-generated demonstrations (Shim et al., 2025; Wang et al., 2025b; Li et al., 2025b), but limits the exploration to these behaviors, restricting discovery of alternative strategies. (2) Single-agent RL methods use a unified model for reasoning and execution, guiding tool invocation via reward design, staged optimization (Li et al., 2025c; Feng et al., 2025; Qian et al., 2025), step-level feedback (Yu et al., 2025), efficiency improvements (Wang et al., 2025a; Song et al., 2025), retrieval-augmented reasoning (Chen et al., 2025; Jin et al., 2025; Zhang et al., 2025), or multi-tool support (Dong et al., 2025a). Unstable trajectory quality in online RL motivates entropy-based dynamic sampling in ARPO (Dong et al., 2025b) and episode filtering in SimpleTIR (Xue et al., 2025). Our approach leverages the role-playing ability of large models to explicitly separate high-level planning from execution. (3) Multi-agent TIR systems implement specialized agents for search or reasoning (Zheng et al., 2025; Lu et al., 2025), but roles are usually fixed and optimization is offline, limiting coordination. OWL (Hu et al., 2025) learns a shared planner from feedback but does not support online multi-agent training, while AgentFlow (Li et al., 2025d) improves multi-turn tool use by fine-tuning the planner while keeping other modules fixed. Our approach trains agents with hierarchical RL for coordinated TIR.

6 Conclusion

We propose TMATR-HRL, a hierarchical multi-agent framework that establishes clear role boundaries through tree-structured reasoning and synchronizes agent co-evolution via hierarchical reinforcement learning. Extensive experiments confirm its effectiveness in reasoning and tool usage.

Limitations

While TMATR-HRL yields promising results, the hierarchical reinforcement learning process involves a complex optimization space that requires careful hyperparameter tuning. Furthermore, our current evaluation primarily focuses on math and QA tasks. Further exploration is needed to assess the framework's scalability in open-domain scenarios with a larger and more dynamic set of tools.

Ethics Statement

This research seeks to address the challenges of role ambiguity and decision-making noise in multi-agent tool-integrated reasoning through hierarchical fine-tuning. Our work is confined to the development of technical methodologies for optimizing agent coordination and is evaluated exclusively on standardized, publicly available datasets. The system does not involve the collection of personal identifiers or sensitive information. Furthermore, the tools integrated within our framework are restricted to neutral computational and retrieval APIs, posing no foreseeable risks of generating biased or harmful content in real-world applications.

References

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, and 1 others. 2024. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, and 1 others. 2025. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*.

Cheng Cheng, Zhenya Huang, GuanHao Zhao, Yuxiang Guo, Xin Lin, Jinze Wu, Xin Li, and Shijin Wang. 2025. From objectives to questions: A planning-based framework for educational mathematical question generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12836–12856, Vienna, Austria. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. 2025a. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning. *arXiv preprint arXiv:2505.16410*.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, and 1 others. 2025b. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.

Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Bowei Xia, Tao Sun, Ziyu Ye, Zhaoxuan Jin, Yingru Li, Qiguang Chen, and 1 others. 2025. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation. *arXiv preprint arXiv:2505.23885*.

Xinda Jia, Jinpeng Li, Zezhong Wang, Jingjing Li, Xingshan Zeng, Yasheng Wang, Weinan Zhang, Yong Yu, and Weiwen Liu. 2025. Fast, slow, and tool-augmented thinking for llms: A review. *arXiv preprint arXiv:2508.12265*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Yachen Kang, Diyuan Shi, Jinxin Liu, Li He, and Donglin Wang. 2023. Beyond reward: offline preference-guided policy optimization. In *Proceedings of the 40th International Conference on Machine Learning*, pages 15753–15768.

Ang Li, Yiquan Wu, Yifei Liu, Ming Cai, Lizhi Qing, Shihang Wang, Yangyang Kang, Chengyuan Liu, Fei Wu, and Kun Kuang. 2025a. UniLR: Unleashing

the power of LLMs on multiple legal tasks with a unified legal retriever. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11953–11967, Vienna, Austria. Association for Computational Linguistics.

Chengpeng Li, Mingfeng Xue, Zhenru Zhang, Jiaxi Yang, Beichen Zhang, Bowen Yu, Binyuan Hui, Junyang Lin, Xiang Wang, and Dayiheng Liu. 2025b. START: Self-taught reasoner with tools. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13523–13564, Suzhou, China. Association for Computational Linguistics.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024. Numinamath tir. [https://huggingface.co/AI-MO/NuminaMath-TIR](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf).

Xuefeng Li, Haoyang Zou, and Pengfei Liu. 2025c. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. 2025d. In-the-flow agentic system optimization for effective planning and tool use. In *NeurIPS 2025 Workshop on Efficient Reasoning*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Rui Lu, Zhenyu Hou, Zihan Wang, Hanchen Zhang, Xiao Liu, Yujiang Li, Shi Feng, Jie Tang, and Yuxiao Dong. 2025. Deepdive: Advancing deep search agents with knowledge graphs and multi-turn rl. *arXiv preprint arXiv:2509.10446*.

Zheheng Luo, Chenhan Yuan, Qianqian Xie, and Sophia Ananiadou. 2025. EMPEC: A comprehensive benchmark for evaluating large language models across diverse healthcare professions. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 9945–9958, Vienna, Austria. Association for Computational Linguistics.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

Jeonghoon Shim, Gyuhyeon Seo, Cheongsu Lim, and Yohan Jo. 2025. Tooldial: Multi-turn dialogue generation method for tool-augmented language models. In *The Thirteenth International Conference on Learning Representations*.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*.

Shuang Sun, Huatong Song, Yuhao Wang, Ruiyang Ren, Jinhao Jiang, Junjie Zhang, Fei Bai, Jia Deng, Wayne Xin Zhao, Zheng Liu, and 1 others. 2025. Simpledeepsearcher: Deep information seeking via web-powered reasoning trajectory synthesis. *arXiv preprint arXiv:2505.16834*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. 2025a. Acting less is reasoning more! teaching model to act efficiently. *arXiv preprint arXiv:2504.14870*.

Pei Wang, Yanan Wu, Noah Wang, Jiaheng Liu, Xiaoshuai Song, ZY Peng, Ken Deng, Chenchen Zhang, Junran Peng, Ge Zhang, and 1 others. 2025b. Mtu-bench: A multi-granularity tool-use benchmark for large language models. In *The Thirteenth International Conference on Learning Representations*.

Jialong Wu, Wenbiao Yin, Yong Jiang, Zhenglin Wang, Zekun Xi, Runnan Fang, Linhai Zhang, Yulan He, Deyu Zhou, Pengjun Xie, and 1 others. 2025. Webwalker: Benchmarking llms in web traversal. In *Workshop on Reasoning and Planning for Large Language Models*.

Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun MA, and Bo An. 2025. Simpletir: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. In *NeurIPS 2025 Fourth Workshop on Deep Learning for Code*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.

Yuanqing Yu, Zhefan Wang, Weizhi Ma, Shuai Wang, Chuhan Wu, Zhiqiang Guo, and Min Zhang. 2025. Steptool: Enhancing multi-step tool usage in llms via step-grained reinforcement learning. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 3952–3962.

Shaokun Zhang, Yi Dong, Jieyu Zhang, Jan Kautz, Bryan Catanzaro, Andrew Tao, Qingyun Wu, Zhiding Yu, and Guilin Liu. 2025. Nemotron-research-tool-n1: Tool-using language models with reinforced reasoning. *arXiv preprint arXiv:2505.00024*.

Yifan Zhang and Team Math-AI. 2024. American invitational mathematics examination (aime) 2024.

Yifan Zhang and Team Math-AI. 2025. American invitational mathematics examination (aime) 2025.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. 2025. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *arXiv preprint arXiv:2504.03160*.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models. *Preprint*, arXiv:2304.06364.

A Data Statistics

Table 3 reports the overall dataset sizes together with the number of instances incorrectly solved by Qwen-7B-Instruct (Bai et al., 2025) under the pass@1 setting, covering NuminaMath-TIR (Li et al., 2024), HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), SimpleDeepSearch (Sun et al., 2025), and WebWalker (Wu et al., 2025). The *Counts* column shows the total number of instances in each dataset, while *Incorrect* indicates cases that cannot be solved correctly in a single attempt by the base model. Across these datasets, incorrectly solved instances are generally more challenging and often require multi-step reasoning or effective tool usage. We therefore use these error cases as training data for reinforcement learning, as learning from difficult instances is essential for improving decision-making and tool-integrated reasoning performance.

Dataset	Task Type	Counts	Incorrect
NuminaMath-TIR	Text	72.4K	17.53K
HotpotQA	Text	8.4K	6.84K
2WikiMultiHopQA	Text	0.87K	0.66K
SimpleDeepSearch	TIR	0.87K	0.67K
WebWalker	Text	0.68K	0.67K
Total	–	82.35K	25.7K

Table 3: Dataset statistics and the number of instances incorrectly solved by Qwen-7B-Instruct under the pass@1 setting.

B Benchmark and Evaluation Details

For benchmarks with large test sets, we follow a fixed subsampling protocol during evaluation. For mathematical reasoning benchmarks with more than 500 test instances, we evaluate models on a fixed subset of 500 problems sampled from the official test split. For question answering benchmarks with more than 200 test instances, we evaluate models on a fixed subset of 200 examples sampled from the official test split. Subsampling is performed once prior to evaluation, and the same subsets are used for all models and methods.

We additionally specify the context length configurations used for different models. For **Qwen3-4B-Instruct-2507** (Yang et al., 2025) and **Phi-4-mini-instruct** (Abdin et al., 2024), the maximum context length for a single dialogue round is set to 4,096 tokens, while the total context length budget across multi-turn interactions is capped at 40000 tokens. For **DeepSeek-R1-Distill-Qwen-1.5B** (Guo et al., 2025), we allow a maximum of 10,000 tokens per single dialogue round and a total context length of up to 50000 tokens for multi-turn conversations. Each example is evaluated with at most 5 turns.

C Baseline Details

This appendix describes the two prompt designs used for the baseline methods. Both prompts are instruction-based and contain no in-context examples. They differ only in the placement of the reasoning and output-format instructions.

- CoT-style Prompt.

```
SYSTEM: Please reason step by step, and
put your final answer within \boxed{}.
USER: {question}
```

- Box-style Prompt.

```
SYSTEM: You are a helpful assistant.
USER: {question}
Please reason step by step, and put your
final answer within \boxed{}.
```

D Training Details

For all experiments, we use fixed training hyperparameters with α set to 0.22, β set to 0.1, γ set to 0.1, and η is set to 0.50. The training batch size is 48, and the DPO mini-batch size $\mathcal{B}$ is 16. We set the rollout number to 6, restrict the maximum number of rounds to 3, and use a learning rate of 1×10^{-6}. When **Qwen3-4B-Instruct-2507** (Yang et al., 2025) is used as the inferencer, both the planner and the inferencer are trained with a maximum response length of 4096 tokens and a maximum prompt length of 9000 tokens. When **DeepSeek-R1-Distill-Qwen-1.5B** (Guo et al., 2025) is used as the inferencer, the maximum response length is 4096 tokens and the maximum prompt length is 12000 tokens. When **Phi-4-mini-instruct** (Abdin et al., 2024) is used as the inferencer, the maximum response length is 4096 tokens and the maximum prompt length is 8000 tokens. For all SFT warm-up runs, the maximum sequence length is set to 20444 tokens, and models are trained for 6 epochs with a learning rate of 5×10^{-6} and a weight decay of 0.1, selecting the checkpoint with the minimum validation loss.

E Fine-grained Training Dynamics of Hierarchical Reinforcement Learning

This appendix provides a fine-grained analysis of the training dynamics in hierarchical reinforcement learning (HRL), complementing the main results by reporting stage-wise performance changes across models and benchmarks. As shown in Figures 5, 6 and 7, fine-tuning the inference model alone yields limited performance gains on most tasks, suggesting that improving low-level execution in isolation is insufficient for effective tool-integrated reasoning. When planner training is introduced, performance variations become more pronounced, and temporary regressions are observed on several benchmarks, indicating potential coordination mismatch between high-level planning and low-level execution. After full system-level optimization, these fluctuations are largely mitigated and performance trends become more consistent across tasks. Overall, this stage-wise analysis supports our motivation that explicitly coordinating decisions across abstraction levels through staged optimization is critical for reliable and controllable tool usage.

F Instructions

Figures 8, 9, 10, 11, 12, 13 and 14 present the complete set of prompt templates used in our framework.

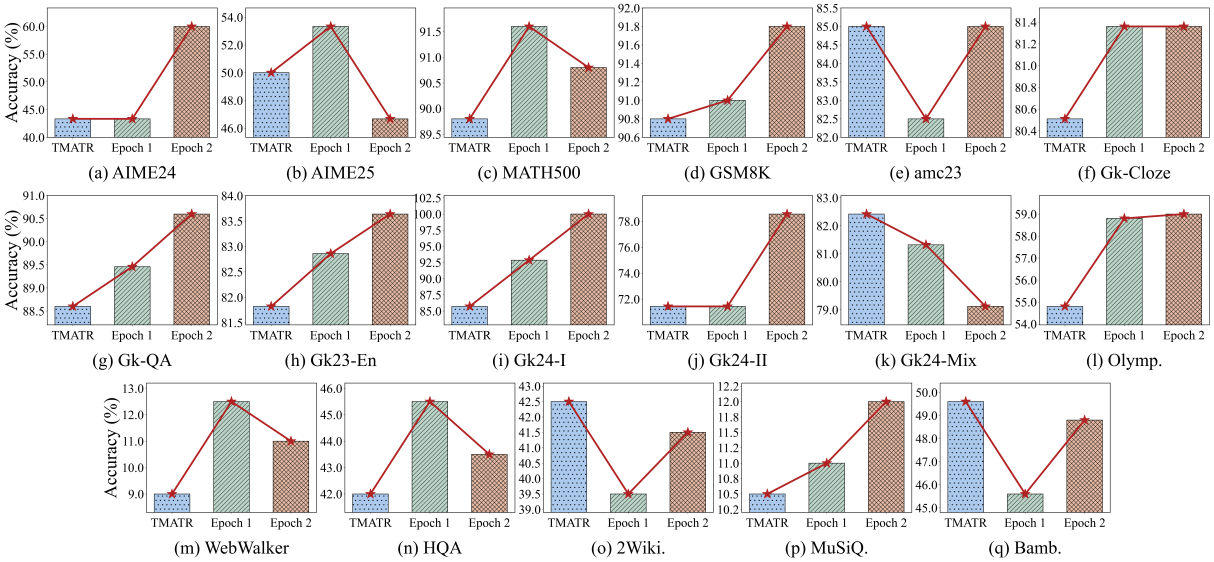


Figure 5: Model analysis of the Hierarchical Reinforcement Learning framework on **Qwen3**.

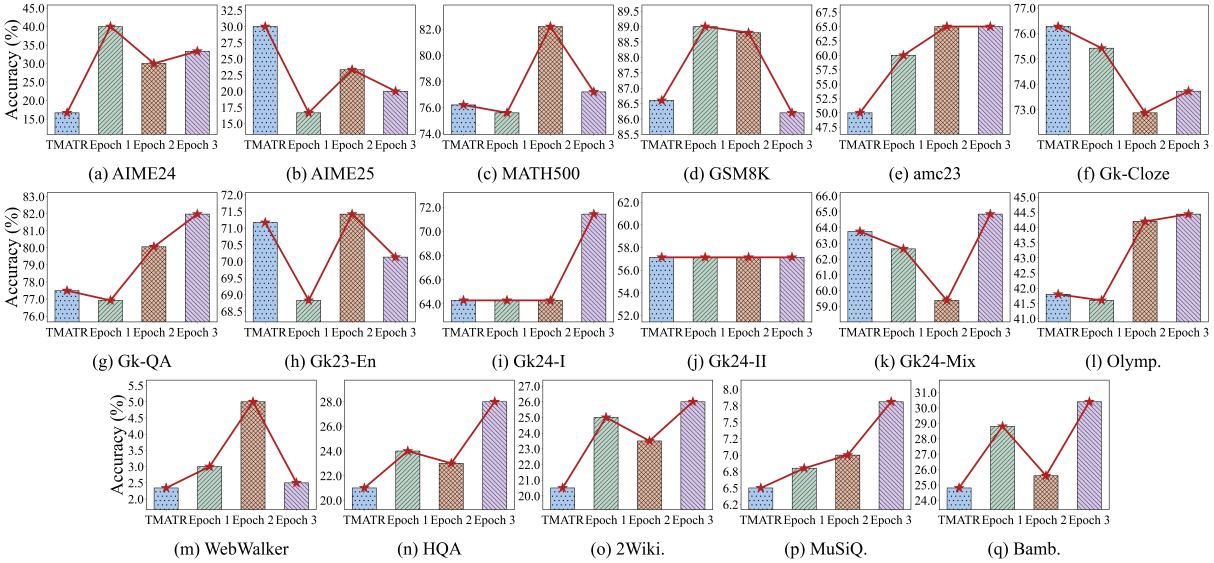


Figure 6: Model analysis of the Hierarchical Reinforcement Learning framework on **Qwen2.5**.

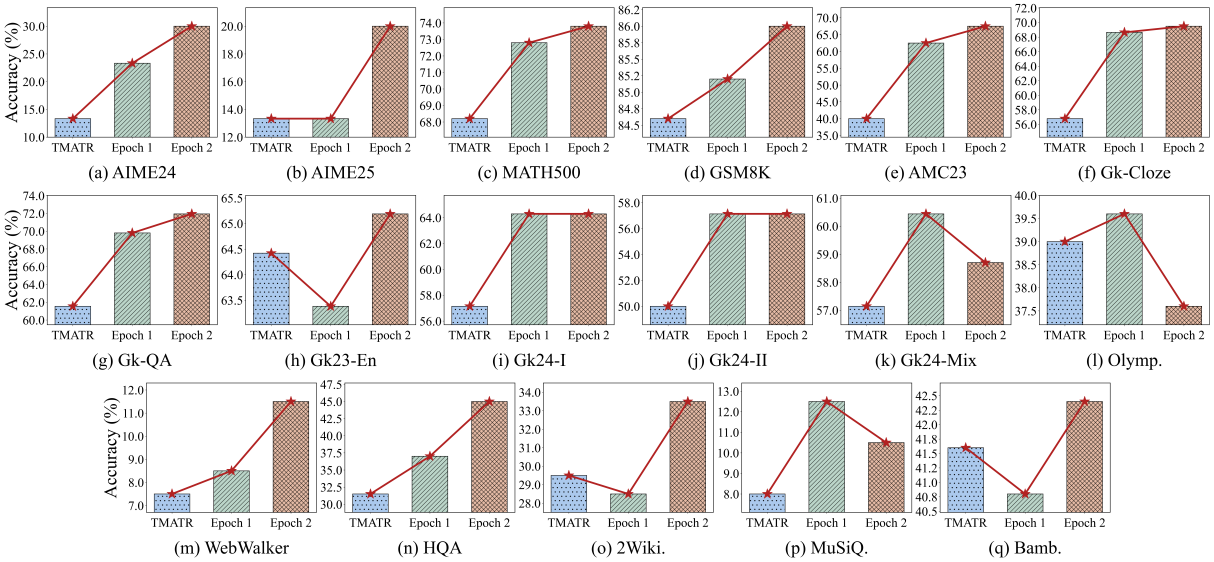


Figure 7: Model analysis of the Hierarchical Reinforcement Learning framework on **Phi-4**.

You are an intelligent planning assistant, responsible for breaking down complex problems into well-defined sub-tasks.
Your responsibilities are:
1. Determine whether an external tool needs to be invoked;
2. If so, clearly specify the tool usage objective and select the most appropriate tool;
3. If not, construct a clear and logical reasoning path in natural language;
4. Strictly follow formatting requirements – all decisions must be output in the required structured format in subsequent prompts.
You will collaborate with other modules to complete the task. Do not repeat the original question, and do not reduce the scope of the problem.

Figure 8: System prompt of the planner.

```
Please solve the following problem step by step:
{question}
You may use the following tools to assist you in this task:
[
  {
    "tool_name": "Search",
   "tool_description": "Used for retrieving factual knowledge (e.g., people, places, definitions), with an emphasis on accuracy and source reliability."
  },
  {
    "tool_name": "WebSearch",
     "tool_description": "Used for web scraping, list comparison, and organizing structured information."
  },
  {
    "tool_name": "Python",
    "tool_description": "Used for mathematical calculations, simulations, enumeration, logical verification, graph computation, and other complex reasoning tasks."
  }
]
You should decide whether to invoke any tools based on the problem, and perform reasoning or construct inputs in subsequent steps.
```

Figure 9: Question Prompt of Planner

Please determine whether the next step requires tool usage or can be completed via language reasoning.
- If tool usage is needed, output: <ACTION>call_tool</ACTION>
- If it can be solved directly, output: <ACTION>text_reasoning</ACTION>
- Then, use <GOAL>...</GOAL> to clearly describe the analysis or computation objective for this step.
Your output must strictly follow the XML tag format below:
<ACTION>...</ACTION>
<GOAL>...</GOAL>
Requirements for GOAL
- The description must be clear and concise, yet fully informative;
- Do not omit key conditions, variable definitions, or task context;
- Avoid vague phrases such as "enumerate" or "judge whether" without further explanation;
- The GOAL must be self-contained and executable without relying on prior context;
- Encourage tool use where helpful.
Examples
<ACTION>call_tool</ACTION>
<GOAL>Enumerate all bijections f from set $A = \{1, 2, 3, 4, 5, 6\}$ to itself such that $f(f(f(x))) = x$</GOAL>
<ACTION>text_reasoning</ACTION>
<GOAL>Analyze all positive integers $a = 2^n \cdot 3^m$ such that a^6 does not divide 6^a</GOAL>
Do not output anything other than the specified tags above.

Figure 10: DECIDE Prompt of Planner.

Based on the tool usage objective proposed in the previous step, select the most appropriate tool from the available list and briefly explain your reasoning.
Tool Usage Objective:
{goal}
The tool name must be selected from the provided list. Use the following output format:
<REASON>(your explanation) </REASON>
<SELECTED_TOOL>(tool name) </SELECTED_TOOL>

Figure 11: SELECT Prompt of Planner.

Based on the current reasoning objective, plan a structured, step-by-step reasoning process. Each step should clearly specify the operation to perform and the underlying logic, progressively guiding the task toward the final answer.
Reasoning Objective:
{goal}
Please output strictly in the following format:
<REASONING_STEPS>
Step 1: ...
Step 2: ...
...
Step N: ...
</REASONING_STEPS>
Note: The steps should describe only the reasoning path. Do not include or infer the final answer at this stage.

Figure 12: PLAN Prompt of Planner.

Please execute the following original question, reasoning objective, and reasoning plan step by step. For each step, provide both the operation performed and its result.
If the question involves multiple candidate values (e.g., numbers, options, or combinations), explicitly list and analyze all of them. Skipping relevant data or cases is strictly prohibited.
–
Original Question
{question}
Current Reasoning Objective
{goal}
Reasoning Plan
{plan}
–
Output Format
<THINK>
Step 1: Perform xxx, the result is xxx.
Step 2: ...
...
</THINK>
<ANSWER>\[\\boxed{{final_answer}} \]</ANSWER>
(Only fill this in after the full reasoning is complete)
Instructions
- You must follow the reasoning plan **exactly as given**, step by step – do not rewrite, skip, merge, or alter any steps;
- Each step must include both the action performed and its result;
- If reasoning is still in progress, output only the <THINK>block without the <ANSWER>tag;
- In the <ANSWER>block, provide the **complete final answer**, not just a label;
- Use \\boxed{{. . . }} to wrap all answers, whether mathematical, symbolic, or textual;
- The boxed content must be sufficient to match or evaluate against the ground truth;
- **Only output the <ANSWER>tag if the final answer contains sufficient information to reconstruct the Original Question** (key entities, quantities, units, and task intent). Otherwise, output only the <THINK>block and omit <ANSWER>.

Figure 13: INFER Prompt of Inferencer.

Based on the following original question, tool usage objective and tool description, generate a valid input for the selected tool.

—

Original Question

{question}

Tool Usage Objective

{goal}

Tool Description

{tool_information}

—

Output Format

<TOOL_INPUT>
(Fill in the tool input content here)
</TOOL_INPUT>
Only output the <TOOL_INPUT>block. Do not include any extra statements, quotation marks, comments, explanations, or markdown.

Figure 14: EXCUTE Prompt of Inferencer.